

Федеральное государственное бюджетное образовательное учреждение высшего образования «Тамбовский государственный университет имени Г.Р. Державина»
Институт экономики, информационных технологий и креативных индустрий
Кафедра математического моделирования и информационных технологий

УТВЕРЖДАЮ:
Директор института



Т. М. Кожевникова
«17» июня 2026 г.

РАБОЧАЯ ПРОГРАММА

по дисциплине Б1.О.36 Программирование на Python

Направление подготовки/специальность: 10.03.01 - Информационная безопасность

Профиль/направленность/специализация: Безопасность компьютерных систем

Уровень высшего образования: бакалавриат

Квалификация: Бакалавр

год набора: 2026

Тамбов, 2026

Автор программы:

Кандидат педагогических наук, доцент Скворцов Александр Александрович

Рабочая программа составлена в соответствии с ФГОС ВО по направлению подготовки 10.03.01 - Информационная безопасность (уровень бакалавриата) (приказ Министерства науки и высшего образования РФ от «17» ноября 2020 г. № 1427).

Рабочая программа принята на заседании Кафедры математического моделирования и информационных технологий «16» июня 2026 г. Протокол № 11

Рассмотрена и одобрена на заседании Ученого совета Института экономики, информационных технологий и креативных индустрий, Протокол от «17» июня 2026 г. № 10.

СОДЕРЖАНИЕ

1. Цели и задачи дисциплины.....	4
2. Место дисциплины в структуре ОП бакалавриата.....	5
3. Объем и содержание дисциплины.....	5
4. Контроль знаний обучающихся и типовые оценочные средства.....	24
5. Методические указания для обучающихся по освоению дисциплины (модуля).....	49
6. Учебно-методическое и информационное обеспечение дисциплины.....	50
7. Материально-техническое обеспечение дисциплины, программное обеспечение, профессиональные базы данных и информационные справочные системы.....	51

1. Цели и задачи дисциплины

1.1 Цель дисциплины – формирование компетенций:

ОПК-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности

1.2 Типы задач профессиональной деятельности, к которым готовятся обучающиеся в рамках освоения дисциплины:

- организационно-управленческий

1.3 Дисциплина ориентирована на подготовку обучающихся к профессиональной деятельности в сфере: 06 Связь, информационные и коммуникационные технологии (в сфере техники и технологии, охватывающей совокупность проблем, связанных с обеспечением защищенности объектов информатизации в условиях существования угроз в информационной сфере)

1.4 В результате освоения дисциплины у обучающихся должны быть сформированы:

Обобщенные трудовые функции / трудовые функции / трудовые или профессиональные действия (при наличии профстандарта)	Код и наименование компетенции ФГОС ВО, необходимой для формирования трудового или профессионального действия	Индикаторы достижения компетенций
	ОПК-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности	Использует язык программирования Python и технологии разработки программных средств для решения задач профессиональной деятельности

1.5 Согласование междисциплинарных связей дисциплин, обеспечивающих освоение компетенций:

ОПК-7 Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности

№ п/п	Наименование дисциплин, определяющих междисциплинарные связи	Форма обучения						
		Очная (семестр)						
		1	2	3	4	5	6	8
1	Алгоритмизация и программирование		+					
2	Основы программирования в корпоративных информационных системах			+	+	+		
3	Преддипломная практика							+
4	Программирование на C++	+	+					
5	Разработка алгоритмов и программных решений	+						

6	Языки программирования					+	+	
---	---------------------------	--	--	--	--	---	---	--

2. Место дисциплины в структуре ОП бакалавриата:

Дисциплина «Программирование на Python» относится к обязательной части учебного плана ОП по направлению подготовки 10.03.01 - Информационная безопасность.

Дисциплина «Программирование на Python» изучается в 1 семестре.

3. Объем и содержание дисциплины

3.1. Объем дисциплины: 3 з.е.

Очная: 3 з.е.

Вид учебной работы	Очная (всего часов)
Общая трудоёмкость дисциплины	108
Контактная работа	48
Лекции (Лекции)	32
Лабораторные (Лаб. раб.)	16
Самостоятельная работа (СР)	60
Зачет	-

3.2. Содержание курса:

№ темы	Название раздела/темы	Вид учебной работы, час.			Формы текущего контроля
		Лек ции	Лаб · раб.	СР	
		О	О	О	
1 семестр					
1	Переменные. Типы данных. Преобразование типов данных	1	1	2	Собеседование; Тестирование
2	Условные операторы и циклы	1	1	6	Собеседование; Тестирование
3	Строки и двоичные данные	1	1	2	Собеседование; Тестирование
4	Функции и методы для работы со строками	1	1	4	Собеседование; Тестирование
5	Списки. Операции над списками	2	3	6	Собеседование; Тестирование
6	Кортежи, множества и диапазоны.	2	1	4	Собеседование; Тестирование
7	Словари. Операции и методы для работы со словарями	2	1	4	Собеседование; Тестирование
8	Работа с датой и временем	2	3	4	Собеседование; Тестирование

9	Пользовательские функции	2	2	4	Собеседование; Тестирование
10	Обработка исключений	2	2	4	Собеседование; Тестирование

Тема 1. Переменные. Типы данных. Преобразование типов данных (ОПК-7)

Лекция.

Python представляет популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение питон получил в области машинного обучения и исследований искусственного интеллекта. Впервые язык Python был анонсирован в 1991 году голландским разработчиком Гвидо Ван Россумом. С тех пор данный язык проделал большой путь развития. В 2000 году была издана версия 2.0, а в 2008 году - версия 3.0. Несмотря на вроде такие большие промежутки между версиями постоянно выходят подверсии. Так, текущей актуальной версией на момент написания данного материала является 3.10, которая вышла в октябре 2021 года. Основные особенности языка программирования Python: Скриптовый язык. Код программ определяется в виде скриптов. Поддержка самых различных парадигм программирования, в том числе объектно-ориентированной и функциональной парадигм. Интерпретация программ. Для работы со скриптами необходим интерпретатор, который запускает и выполняет скрипт. Выполнение программы на Python выглядит следующим образом. Сначала мы пишем в текстовом редакторе скрипт с набором выражений на данном языке программирования. Передаем этот скрипт на выполнение интерпретатору. Интерпретатор транслирует код в промежуточный байткод, а затем виртуальная машина переводит полученный байткод в набор инструкций, которые выполняются операционной системой. Здесь стоит отметить, что хотя формально трансляция интерпретатором исходного кода в байткод и перевод байткода виртуальной машиной в набор машинных команд представляют два разных процесса, но фактически они объединены в самом интерпретаторе. Выполнение программы на Python Портативность и платформенезависимость. Не имеет значения, какая у нас операционная система - Windows, Mac OS, Linux, нам достаточно написать скрипт, который будет запускаться на всех этих ОС при наличии интерпретатора Автоматическое управление памяти Динамическая типизация Python - очень простой язык программирования, он имеет лаконичный и в то же время довольно простой и понятный синтаксис. Соответственно его легко изучать, и собственно это одна из причин, по которой он является одним из самых популярных языков программирования именно для обучения. В частности, в 2014 году он был признан самым популярным языком программирования для обучения в США. Python также популярен не только в сфере обучения, но в написании конкретных программ в том числе коммерческого характера. В немалой степени поэтому для этого языка написано множество библиотек, которые мы можем использовать. Кроме того, у данного языка программирования очень большое комьюнити, в интернете можно найти по данному языку множество полезных материалов, примеров, получить квалифицированную помощь специалистов.

Лабораторные работы.

1. Произвести сложение и вычитание *a* и *b*. Результат вычислений сохранить в переменных *c* и *d*.
2. Исправить, приведенный ниже, блок кода.

```
a:=2
```

```
b-a=c
```

```
a+b:=c
```

3. Отредактируйте код, что бы он выводил заданный текст.

```
# данный код
```

```
a, b = 45, 54
```

```
c = a + 1
```

```
d = c +
```

```
print(d)
```

```
# требуемый вывод:
```

56

4. Допишите код, чтобы получить требуемый вывод.

данный код

a, b = 45, 54

c = a + 1

d = c +

print(d)

требуемый вывод:

56

5. Создайте переменные, чтобы вывести требуемый текст.

данный код

print("Мой стек:", programming_lang_1, programming_lang_2, programming_lang_3)

требуемый вывод:

Мой стек: python, javascript, php

Задания для самостоятельной работы.

1. Дано целое десятичное число. Выведите его последнюю цифру.

2. Дано целое десятичное число. Найдите число десятков в его десятичной записи.

3. Дано трехзначное число. Найдите сумму его цифр.

4. Пирожок в столовой стоит a рублей и b копеек. Определите, сколько рублей и копеек нужно заплатить за n пирожков.

5. В школе решили набрать три новых математических класса.

Так как занятия по математике у них проходят в одно и то же время,

было решено выделить кабинет для каждого класса и купить в них новые парты.

За каждой партией может сидеть не больше двух учеников.

Известно количество учащихся в каждом из трёх классов.

Сколько всего нужно закупить парт чтобы их хватило на всех учеников? Программа получает на вход три целых десятичных числа:

количество учащихся в каждом из трех классов.

6. Доработайте код задачи № 3 таким образом, чтобы он запрашивал время начала занятий (минуты и часы отдельно) и номер урока,

а далее также рассчитывал время окончания уроков.

7. Пользователь вводит число и систему счисления этого числа. Программа переводит число в десятичную, двоичную, восьмеричную и шестнадцатеричную системы счисления с использованием стандартных функций.

Тема 2. Условные операторы и циклы (ОПК-7)

Лекция.

Говоря простым языком, в выражении $2 + 3$, числа "2" и "3" называются операндами, знак "+" оператором. В языке программирования Python существуют следующие типы операторов:

Арифметические операторы
 Операторы сравнения (реляционные)
 Операторы присваивания
 Побитовые операторы
 Логические операторы
 Операторы членства (Membership operators)
 Операторы тождественности (Identity operators)

Рассмотрим их по порядку.

Арифметические операторы в Python:

Оператор + Описание: Сложение - Суммирует значения слева и справа от оператора. Примеры: $15 + 5$ в результате будет 20, $20 + -3$ в результате будет 17, $13.4 + 7$ в результате будет 20.4.

Оператор - Описание: Вычитание - Вычитает правый операнд из левого. Примеры: $15 - 5$ в результате будет 10, $20 - -3$ в результате будет 23, $13.4 - 7$ в результате будет 6.4.

Оператор * Описание: Умножение - Перемножает операнды. Примеры: $5 * 5$ в результате будет 25, $7 * 3.2$ в результате будет 22.4, $-3 * 12$ в результате будет -36.

Оператор / Описание: Деление - Делит левый операнд на правый. Примеры: $15 / 5$ в результате будет 3, $5 / 2$ в результате будет 2.5 (В Python 2.x версии при делении двух целых чисел результат будет целое число). $5.0 / 2$ в результате будет 2.5 (Чтобы получить "правильный" результат хотя бы один операнд должен быть float).

Оператор % Описание: Деление по модулю - Делит левый операнд на правый и возвращает остаток. Примеры: $6 \% 2$ в результате будет 0, $7 \% 2$ в результате будет 1, $13.2 \% 5$ в результате будет 3.2.

Оператор ** Описание: Возведение в степень - возводит левый операнд в степень правого. Примеры: $5 ** 2$ в результате будет 25, $2 ** 3$ в результате будет 8, $-3 ** 2$ в результате будет -9.

Оператор // Описание: Целочисленное деление - Деление в котором возвращается только целая часть результата. Часть после запятой отбрасывается. Примеры: $12 // 5$ в результате будет 2, $4 // 3$ в результате будет 1, $25 // 6$ в результате будет 4.

Операторы сравнения в Python:

Оператор == Описание: Проверять равны ли оба операнда. Если да, то условие становится истинным. Примеры: $5 == 5$ в результате будет True, $True == False$ в результате будет False, $"hello" == "hello"$ в результате будет True.

Оператор != Описание: Проверять равны ли оба операнда. Если нет, то условие становится истинным. Примеры: $12 != 5$ в результате будет True, $False != False$ в результате будет False, $"hi" != "Hi"$ в результате будет True.

Оператор <> Описание: Проверять равны ли оба операнда. Если нет, то условие становится истинным. Примеры: $12 <> 5$ в результате будет True.

Оператор > Описание: Проверять больше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным. Примеры: $5 > 2$ в результате будет True, $True > False$ в результате будет True, $"A" > "B"$ в результате будет False.

Оператор < Описание: Проверять меньше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным. Примеры: $3 < 5$ в результате будет True, $True < False$ в результате будет False, $"A" < "B"$ в результате будет True.

Оператор >= Описание: Проверять больше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным. Примеры: $1 >= 1$ в результате будет True, $23 >= 3.2$ в результате будет True, $"C" >= "D"$ в результате будет False.

Оператор <= Описание: Проверять меньше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным. Примеры: $4 <= 5$ в результате будет True, $0 <= 0.0$ в результате будет True, $-0.001 <= -36$ в результате будет False.

Операторы присваивания в Python:

Оператор = Описание: Присваивает значение правого операнда левому. Примеры: $c = 23$ присвоит переменной c значение 23.

Оператор += Описание: Прибавит значение правого операнда к левому и присвоит эту сумму левому операнду. Примеры: $c = 5$, $a = 2$, $c += a$ равносильно: $c = c + a$, c будет равно 7.

Оператор -= Описание: Отнимает значение правого операнда от левого и присваивает результат левому операнду. Примеры: $c = 5$, $a = 2$, $c -= a$ равносильно: $c = c - a$, c будет равно 3.

Оператор *= Описание: Умножает правый операнд с левым и присваивает результат левому операнду. Примеры: $c = 5$, $a = 2$, $c *= a$ равносильно: $c = c * a$, c будет равно 10.

Оператор /= Описание: Делит левый операнд на правый и присваивает результат левому операнду. Примеры: $c = 10$, $a = 2$, $c /= a$ равносильно: $c = c / a$, c будет равно 5.

Оператор %= Описание: Делит по модулю операнды и присваивает результат левому. Примеры: $c = 5$, $a = 2$, $c %= a$ равносильно: $c = c \% a$, c будет равно 1.

Оператор **= Описание: Возводит в левый операнд в степень правого и присваивает результат левому операнду. Примеры: $c = 3$, $a = 2$, $c **= a$ равносильно: $c = c ** a$, c будет равно 9.

Оператор //= Описание: Производит целочисленное деление левого операнда на правый и присваивает результат левому операнду. Примеры: $c = 11$, $a = 2$, $c //= a$ равносильно: $c = c // a$, c будет равно 5.

Лабораторные работы.

1. Занятия в школе начинаются в 8-30. Урок длится 45 минут, перерывы между уроками – 10 минут. Ввести номер урока и вывести время его окончания.
2. Ввести число, обозначающее размер одной фотографии в Мбайтах. Определить, сколько фотографий поместится на флэш-карту объёмом 2 Гбайта.
3. Разведчики-математики для того, чтобы опознать своих, используют числовые пароли. Услышав число-пароль, разведчик должен возвести его в квадрат и сказать в ответ первую цифры дробной части полученного числа.

Напишите программу, которая по полученному паролю (вещественному числу) вычисляет число-ответ.

4. В игре «Русское лото» из мешка случайным образом выбираются бочонки, на каждом из которых написано число от 1 до 90. Напишите программу,

которая выводит наугад первые 5 выигрышных номеров.

5. Напишите программу, которая получает возраст человека (целое число, не превышающее 120) и выводит этот возраст со словом «год»,

«года» или «лет». Например, «21 год», «22 года», «25 лет».

6. Напишите программу, которая получает с клавиатуры натуральное число и определяет, сколько раз в его десятичной записи встречается цифра 1

7. Напишите программу, которая получает с клавиатуры натуральное число и определяет, есть ли в его десятичной записи одинаковые цифры, стоящие рядом.

Задания для самостоятельной работы.

1. Напишите программу, которая получает с клавиатуры два целых числа и вычисляет их произведение,

используя только операции сложения.

2. Напишите программу, которая получает с клавиатуры натуральное число и вычисляет целый квадратный корень из него – наибольшее число,

квадрат которого не больше данного числа.

3. Натуральное число называется числом Армстронга, если сумма цифр числа, возведенных в N-ную степень (где N – количество цифр в числе) равна самому числу.

.

4. Вывести на экран циклом пять строк из нулей, причем каждая строка должна быть пронумерована;

5. Найти сумму ряда чисел от 1 до 100. Полученный результат вывести на экран;

6. Даны три числа. Вывести на экран «yes», если среди них есть одинаковые, иначе вывести «ERROR»;

7. Даны три числа. Вывести на экран «yes», если можно взять какие-то два из них и в сумме получить третье;

8. Дано три числа. Найти количество положительных чисел среди них;

9. Вывести на экран все чётные значения в диапазоне от 1 до 497;

10. Написать программу, которая будет складывать, вычитать, умножать или делить два числа.

Числа и знак операции вводятся пользователем. После выполнения вычисления

Тема 3. Строки и двоичные данные (ОПК-7)

Лекция.

В уроке по присвоению типа переменной в Python вы могли узнать, как определять строки: объекты, состоящие из последовательности символьных данных. Обработка строк неотъемлемая часть программирования на python. Крайне редко приложение, не использует строковые типы данных. Из этого урока вы узнаете: Python предоставляет большую коллекцию операторов, функций и методов для работы со строками. Когда вы закончите изучение этой документации, узнаете, как получить доступ и извлечь часть строки, а также познакомитесь с методами, которые доступны для манипулирования и изменения строковых данных. Ниже рассмотрим операторы, методы и функции, доступные для работы с текстом. Строковые операторы Вы уже видели операторы + и * в применении их к числовым значениям в уроке по операторам в Python . Эти два оператора применяются и к строкам. Оператор сложения строк + — оператор конкатенации строк. Он возвращает строку, состоящую из других строк, как показано здесь: `>>> s = 'py' >>> t = 'th' >>> u = 'on' >>> s + t 'pyth' >>> s + t + u 'python' >>> print('Привет, ' + 'Мир!') Go team!!!` Оператор умножения строк * — оператор создает несколько копий строки. Если s это строка, а n целое число, любое из следующих выражений возвращает строку, состоящую из n объединенных копий s: `s * n` или `n * s` Вот примеры умножения строк: `>>> s = 'py.' >>> s * 4 'py.py.py.py.' >>> 4 * s 'py.py.py.py.'` Значение множителя n должно быть целым положительным числом. Оно может быть нулем или отрицательным, но в этом случае результатом будет пустая строка: `>>> 'py' * -6 ''` Если вы создадите строковую переменную и превратите ее в пустую строку, с помощью `'py' * -6`, кто-нибудь будет справедливо считать вас немного глупым. Но это работает.

Лабораторные работы.

1. Дана строка, состоящая из слов, разделенных пробелами. Определите, сколько в ней слов. Используйте для решения задачи метод `count`.
2. Дана строка. Разрежьте ее на две равные части (если длина строки — четная, а если длина строки нечетная, то длина первой части должна быть на один символ больше). Переставьте эти две части местами, результат запишите в новую строку и выведите на экран.
3. Дана строка, состоящая ровно из двух слов, разделенных пробелом. Переставьте эти слова местами. Результат запишите в строку и выведите получившуюся строку.
4. Дана строка, в которой буква `h` встречается как минимум два раза. Разверните последовательность символов, заключенную между первым и последним появлением буквы `h`, в противоположном порядке.
5. Дана строка. Замените в этой строке все цифры `1` на слово `one`.

Задания для самостоятельной работы.

1. Строки в python обозначаются кавычками. Приведите все способы.
2. Какие типы данных можно преобразовать в строку?
3. Опишите синтаксис срезов строк при помощи квадратных скобок.
4. Как применяют операции сложения и умножения к строкам?
5. Что такое двоичные данные?

Тема 4. Функции и методы для работы со строками (ОПК-7)

Лекция.

Для определения функции нужно всего лишь написать ключевое слово `def` перед ее именем, а после — поставить двоеточие. Следом идет блок инструкций. Последняя строка в блоке инструкций может начинаться с `return`, если нужно вернуть какое-то значение. Если инструкции `return` нет, тогда по умолчанию функция будет возвращать объект `None`. Как в этом примере: `i = 0` `def increment():`
`global i` `i += 1` Функция инкрементирует глобальную переменную `i` и возвращает `None` (по умолчанию). Вызовы Для вызова функции, которая возвращает переменную, нужно ввести: `surface = compute_surface(1.)` Для вызова функции, которая ничего не возвращает: `increment()` Еще Функцию можно записать в одну строку, если блок инструкций представляет собой простое выражение: `def sum(a, b): return a + b` Функции могут быть вложенными: `def func1(a, b):` `def inner_func(x):`
`return x*x*x` `return inner_func(a) + inner_func(b)` Функции — это объекты, поэтому их можно присваивать переменным. Инструкция `return` Возврат простого значения Аргументы можно использовать для изменения ввода и таким образом получать вывод функции. Но куда удобнее использовать инструкцию `return`, примеры которой уже встречались ранее. Если ее не написать, функция вернет значение `None`. Возврат нескольких значений Получи бесплатный пробный урок английского Пока что функция возвращала только одно значение или не возвращала ничего (объект `None`). А как насчет нескольких значений? Этого можно добиться с помощью массива. Технически, это все еще один объект. Например: `def stats(data):` `"""данные должны быть списком"""` `_sum =`
`sum(data)` `# обратите внимание на подчеркивание, чтобы избежать переименования встроенной`
`функции sum` `mean = _sum / float(len(data))` `# обратите внимание на использование функции float,`
`чтобы избежать деления на целое число` `variance = sum([(x-mean)**2/len(data) for x in data])`
`return mean, variance` `# возвращаем x,y — кортеж!` `m, v = stats([1, 2, 1])` Аргументы и параметры В функции можно использовать неограниченное количество параметров, но число аргументов должно точно соответствовать параметрам. Эти параметры представляют собой позиционные аргументы. Также Python предоставляет возможность определять значения по умолчанию, которые можно задавать с помощью аргументов-ключевых слов. Параметр — это имя в списке параметров в первой строке определения функции. Он получает свое значение при вызове. Аргумент — это реальное значение или ссылка на него, переданное функции при вызове. В этой функции: `def sum(x, y):`
`return x + y` `x` и `y` — это параметры, а в этой: `sum(1, 2)` `1` и `2` — аргументы. При определении функции параметры со значениями по умолчанию нужно указывать до позиционных аргументов: `def compute_surface(radius, pi=3.14159):` `return pi * radius * radius` Если использовать необязательный параметр, тогда все, что указаны справа, должны быть параметрами по умолчанию. Выходит, что в следующем примере допущена ошибка: `def compute_surface(radius=1, pi):` `return pi * radius * radius` Выберите востребованную профессию Для вызовов это работает похожим образом. Сначала нужно указывать все позиционные аргументы, а только потом необязательные: `S = compute_surface(10,`
`pi=3.14)` На самом деле, следующий вызов корректен (можно конкретно указывать имя позиционного аргумента), но этот способ не пользуется популярностью: `S = compute_surface(radius=10, pi=3.14)` А этот вызов некорректен: `S = compute_surface(pi=3.14, 10)` При вызове функции с аргументами по умолчанию можно указать один или несколько, и порядок не будет иметь значения: `def`
`compute_surface2(radius=1, pi=3.14159):` `return pi * radius * radius` `S = compute_surface2(radius=1,`
`pi=3.14)` `S = compute_surface2(pi=3.14, radius=10.)` `S = compute_surface2(radius=10.)` Можно не указывать ключевые слова, но тогда порядок имеет значение. Он должен соответствовать порядку параметров в определении: `S = compute_surface2(10., 3.14)` `S = compute_surface2(10.)` Если ключевые слова не используются, тогда нужно указывать все аргументы: `def f(a=1, b=2, c=3):` `return a + b + c` Второй аргумент можно пропустить: `f(1,,3)` Чтобы обойти эту проблему, можно использовать словарь: `params = {'a':10, 'b':20}` `S = f(**params)`

Лабораторные работы.

1) Что будет выведено в результате выполнения следующих строк кода?

```
String = "hello" + ","
```

```
String = string + "World!"
```

```
Print(string)
```

```
1)string
```

```
2)Hello, World!
```

```
3)Hello+, +World
```

```
4)World
```

Ответ:

2) Что возвращает функция `len()` при передаче в неё строки?

- 1) Размер строки в байтах
- 2) Количество символов в строке
- 3) Количество слов
- 4) Данная функция не предназначена для строк

Ответ:

3) Что делает следующая программа?

```
string = "explain your opinion"
```

```
var= string.split()
```

- 1) Разделяет предложение на символы
- 2) Удаляет пробелы из предложения
- 3) Делает все заглавные символы прописными
- 4) Разделяет предложение на слова

Ответ:

4) Сколько символов будет содержать строка `str1` после запуска программы?

```
Str1 = "abc bc bcaa"
```

```
Str1 = Str1.replace("a", "bc")
```

```
Str1 = Str1.replace("bc", "1")
```

- 1) 1
- 2) 7
- 3) 8
- 4) 11

Ответ:

5) Сколько символов будет содержать строка `str1` после выполнения следующих строк кода?

```
Str1 = "good morning"
```

```
Str1 = str1[2:6] + str1[1:3]
```

- 1) 5
- 2) 6
- 3) 8
- 4) 2

Ответ:

Задания для самостоятельной работы.

1. Даны четыре действительных числа: `x1`, `y1`, `x2`, `y2`. Напишите функцию `distance(x1, y1, x2, y2)`, вычисляющая расстояние между точкой `(x1, y1)` и `(x2, y2)`.

Считайте четыре действительных числа и выведите результат работы этой функции.

2. Дано действительное положительное число `a` и целое число `n`. Вычислите `an`.

3. Напишите функцию `search_substr(subst, st)`, которая принимает 2 строки и определяет, имеется ли подстрока `subst` в строке `st`. В случае нахождения подстроки, возвращается фраза «Есть контакт!», а иначе «Мимо!».

Должно быть найдено совпадение независимо от регистра обеих строк.

Тема 5. Списки. Операции над списками (ОПК-7)

Лекция.

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться). Чтобы использовать списки, их нужно создать. Создать список можно несколькими способами. Например, можно обработать любой итерируемый объект (например, строку) встроенной функцией `list`: `>>> list('список') ['с', 'п', 'и', 'с', 'о', 'к']` Список можно создать и при помощи литерала: `>>> s = [] # Пустой список >>> l = ['s', 'p', ['isok'], 2] >>> s [] >>> l ['s', 'p', ['isok'], 2]` Как видно из примера, список может содержать любое количество любых объектов (в том числе и вложенные списки), или не содержать ничего. И еще один способ создать список - это генераторы списков. Генератор списков - способ построить новый список, применяя выражение к каждому элементу последовательности. Генераторы списков очень похожи на цикл `for`. `>>> c = [c * 3 for c in 'list'] >>> c ['lll', 'iii', 'sss', 'ttt']` Возможна и более сложная конструкция генератора списков: `>>> c = [c * 3 for c in 'list' if c != 'i'] >>> c ['lll', 'sss', 'ttt'] >>> c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a'] >>> c ['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']` Но в сложных случаях лучше пользоваться обычным циклом `for` для генерации списков. Функции и методы списков Создать создали, теперь нужно со списком что-то делать. Для списков доступны основные встроенные функции, а также методы списков. Таблица "методы списков" Метод Что делает `list.append(x)` Добавляет элемент в конец списка `list.extend(L)` Расширяет список `list`, добавляя в конец все элементы списка `L` `list.insert(i, x)` Вставляет на *i*-ый элемент значение `x` `list.remove(x)` Удаляет первый элемент в списке, имеющий значение `x`. `ValueError`, если такого элемента не существует `list.pop([i])` Удаляет *i*-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент `list.index(x, [start [, end]])` Возвращает положение первого элемента со значением `x` (при этом поиск ведется от `start` до `end`) `list.count(x)` Возвращает количество элементов со значением `x` `list.sort([key=функция])` Сортирует список на основе функции `list.reverse()` Разворачивает список `list.copy()` Поверхностная копия списка `list.clear()` Очищает список Нужно отметить, что методы списков, в отличие от строковых методов, изменяют сам список, а потому результат выполнения не нужно записывать в эту переменную. `>>> l = [1, 2, 3, 5, 7] >>> l.sort() >>> l [1, 2, 3, 5, 7] >>> l = l.sort() >>> print(l) None` И, напоследок, примеры работы со списками: `>>> a = [66.25, 333, 333, 1, 1234.5] >>> print(a.count(333), a.count(66.25), a.count('x')) 2 1 0 >>> a.insert(2, -1) >>> a.append(333) >>> a [66.25, 333, -1, 333, 1, 1234.5, 333] >>> a.index(333) 1 >>> a.remove(333) >>> a [66.25, -1, 333, 1, 1234.5, 333] >>> a.reverse() >>> a [333, 1234.5, 1, 333, -1, 66.25] >>> a.sort() >>> a [-1, 1, 66.25, 333, 333, 1234.5]` Изредка, для увеличения производительности, списки заменяют гораздо менее гибкими массивами (хотя в таких случаях обычно используют сторонние библиотеки, например `NumPy`).

Лабораторные работы.

1. Выведите все элементы списка с четными индексами (то есть `A[0]`, `A[2]`, `A[4]`, ...).
2. Дан список чисел. Выведите все элементы списка, которые больше предыдущего элемента.
3. Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух своих соседей, и выведите количество таких элементов. Крайние элементы списка никогда не учитываются, поскольку у них недостаточно соседей.
4. Дан список, упорядоченный по неубыванию элементов в нем. Определите, сколько в нем различных элементов.
5. В списке все элементы различны. Поменяйте местами минимальный и максимальный элемент этого списка.

Задания для самостоятельной работы.

- 1) Как создать список?
- 1) Все варианты верны
- 2) `L = list(1, 2, 3)`
- 3) `l = [1, 2, 3]`
- 4) `l = list[1, 2, 3]`

Ответ:

- 2) Что выведет этот код:

```
a = [ 1, 342, 223, 'Африка', 'Очки']
print(a[-3])
```

1)223

2)'Африка'

3)342

4)Ошибку

Ответ:

3) Что выведет этот код:

```
sample = [10, 20, 30]
```

```
sample.append(60)
```

```
sample.insert(3, 40)
```

```
print(sample)
```

1)[10, 20, 30, 40]

2)[10, 20, 30, 40, 60]

3)[10, 20, 30, 60, 40]

4)[60, 10, 20, 30, 40]

Ответ:

4) Что из перечисленного правда?

1)Элементы списка не могут повторяться

2)Все элементы списка должны быть одного типа

3)Мы можем вставить элемент на любую позицию в списке

4)Список не может содержать вложенных списков

Ответ:

5) Как получить ['bar', 'baz'] из списка

```
a = ['foo', 'bar', 'baz', 'qux', 'quux']
```

?

1)print(a[2:4])

2)print(a[1], a[2])

3)print(a[1:-2])

4)print(a[-4:-3])

5)print(a[2:3])

Ответ:

Тема 6. Кортежи, множества и диапазоны. (ОПК-7)

Лекция.

Зачем нужны кортежи, если есть списки? Защита от дурака. То есть кортеж защищен от изменений, как намеренных (что плохо), так и случайных (что хорошо). Меньший размер. Дабы не быть голословным: >>> >>> a = (1, 2, 3, 4, 5, 6) >>> b = [1, 2, 3, 4, 5, 6] >>> a.__sizeof__() 36 >>> b.__sizeof__() 44

Возможность использовать кортежи в качестве ключей словаря: >>> >>> d = {(1, 1, 1): 1} >>> d {(1, 1, 1): 1} >>> d = {[1, 1, 1]: 1} Traceback (most recent call last): File "", line 1, in d = {[1, 1, 1]: 1} TypeError: unhashable type: 'list'

Как работать с кортежами? С преимуществами кортежей разобрались, теперь встает вопрос - а как с ними работать. Примерно так же, как и со списками. Создаем пустой кортеж: >>> >>> a = tuple() # С помощью встроенной функции tuple() >>> a () >>> a = () # С помощью литерала кортежа >>> a () >>> Создаем кортеж из одного элемента: >>> >>> a = ('s') >>> a 's' Стоп. Получилась строка. Но как же так? Мы же кортеж хотели! Как же нам кортеж получить? >>> >>> a = ('s',) >>> a ('s',) Ура! Заработало! Все дело - в запятой. Сами по себе скобки ничего не значат, точнее, значат то, что внутри них находится одна инструкция, которая может быть отделена пробелами, переносом строк и прочим мусором. Кстати, кортеж можно создать и так: >>> >>> a = 's', >>> a ('s',) Но все же не увлекайтесь, и ставьте скобки, тем более, что бывают случаи, когда скобки необходимы. Ну и создать кортеж из итерируемого объекта можно с помощью все той же пресловутой функции tuple() >>> >>> a = tuple('hello, world!') >>> a ('h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd', '!')

Операции с кортежами Все операции над списками, не изменяющие список (сложение, умножение на число, методы index() и count() и некоторые другие операции). Можно также по-разному менять элементы местами и так далее. Например, гордость программистов на python - поменять местами значения двух переменных: a, b = b, a

Лабораторные работы.

1. Напишите функцию tpl_sort(), которая сортирует кортеж, состоящий из целых чисел по возрастанию и возвращает его.
Если хотя бы один элемент не является целым числом, то функция возвращает исходный кортеж.
2. Перед студентом стоит задача: на вход функции sieve() поступает список целых чисел.
В результате выполнения этой функции будет получен кортеж уникальных элементов списка в обратном порядке.
3. Дан текст: в первой строке записано число строк, далее идут сами строки. Определите, сколько различных слов содержится в этом тексте.
4. Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово YES (в отдельной строке), если это число ранее встречалось в последовательности или NO, если не встречалось.
5. Даны два списка чисел. Найдите все числа, которые входят как в первый, так и во второй список и выведите их в порядке возрастания.

Задания для самостоятельной работы.

- 1) Что такое множество в Python?
- 1) Это любая коллекция элементов
- 2) Это список, содержащий в себе только функции
- 3) Это контейнер, значения в котором не повторяются
- 4) Это список, содержащий вложенные списки в себе

Ответ:

- 2) Каким образом правильно объявляется множество?

- 1) a = {}
- 2) a = []
- 3) a = set()
- 4) a = set

Ответ:

- 3) Чем отличаются методы remove() и discard(), применяемые к множеству?

- 1) Ничем
- 2) remove() удаляет элемент если он есть, но бросает ошибку если элемента нет. discard() просто удаляет элемент если он есть

3)discard() удаляет элемент если он есть, но бросает ошибку если элемента нет. remove() просто удаляет элемент если он есть

4)Метода discard() для множеств не существует

Ответ:

4) Что такое frozenset?

1)Множество, которое нельзя изменять

2)Множество, которое хранит в себе только неизменяемые объекты

3)Множество, которое используется для хранения констант

4)Выдумка нашей редакции

Ответ:

5) Что это за метод такой, set.difference(another_set)

1)Возвращает True, если два множества одинаковые, False если хотя бы один элемент не совпадает

2)Возвращает True, если два множества разные, False если хотя бы один элемент совпадает

3)Возвращает множество из элементов, которые встречаются только в множестве set

4)Возвращает множество из элементов, которые встречаются только в множестве another_set

Ответ:

Тема 7. Словари. Операции и методы для работы со словарями (ОПК-7)

Лекция.

Сегодня я расскажу о таком типе данных, как словари, о работе со словарями, операциях над ними, методах, о генераторах словарей. Словари в Python - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами. Чтобы работать со словарём, его нужно создать. Сделать это можно несколькими способами. Во-первых, с помощью литерала: `>>> >>> d = {} >>> d {} >>> d = {'dict': 1, 'dictionary': 2} >>> d {'dict': 1, 'dictionary': 2}` Во-вторых, с помощью функции dict: `>>> >>> d = dict(short='dict', long='dictionary') >>> d {'short': 'dict', 'long': 'dictionary'} >>> d = dict([(1, 1), (2, 4)]) >>> d {1: 1, 2: 4}` В-третьих, с помощью метода fromkeys: `>>> >>> d = dict.fromkeys(['a', 'b']) >>> d {'a': None, 'b': None} >>> d = dict.fromkeys(['a', 'b'], 100) >>> d {'a': 100, 'b': 100}` В-четвертых, с помощью генераторов словарей, которые очень похожи на генераторы списков. `>>> >>> d = {a: a ** 2 for a in range(7)} >>> d {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}` Теперь попробуем добавить записей в словарь и извлечь значения ключей: `>>> >>> d = {1: 2, 2: 4, 3: 9} >>> d[1] 2 >>> d[4] = 4 ** 2 >>> d {1: 2, 2: 4, 3: 9, 4: 16} >>> d['1'] Traceback (most recent call last): File "", line 1, in d['1'] KeyError: '1'` Как видно из примера, присвоение по новому ключу расширяет словарь, присвоение по существующему ключу перезаписывает его, а попытка извлечения несуществующего ключа порождает исключение. Для избежания исключения есть специальный метод (см. ниже), или можно перехватывать исключение. Что же можно ещё делать со словарями? Да то же самое, что и с другими объектами: встроенные функции, ключевые слова (например, циклы for и while), а также специальные методы словарей. Методы словарей dict.clear() - очищает словарь. dict.copy() - возвращает копию словаря. classmethod dict.fromkeys(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None). dict.get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None). dict.items() - возвращает пары (ключ, значение). dict.keys() - возвращает ключи в словаре. dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение). dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены. dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None). dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!). dict.values() - возвращает значения в словаре.

Лабораторные работы.

1. В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее.

2. Дан текст: в первой строке задано число строк, далее идут сами строки. Выведите слово, которое в этом тексте чаще встречается.

Если таких слов несколько, выведите то, которое меньше в лексикографическом порядке.

3. Дан список стран и городов каждой страны. Затем даны названия городов. Для каждого города укажите, в какой стране он находится.

4. Во входных данных записано дерево в том же формате, что и в предыдущей задаче. Далее идет число запросов K. В каждой из следующих K строк, содержатся имена двух элементов дерева.

Для каждого такого запроса выведите одно из трех чисел: 1, если первый элемент является предком второго, 2, если второй является предком первого или 0,

если ни один из них не является предком другого.

5. В генеалогическом древе определите для двух элементов их наименьшего общего предка (Lowest Common Ancestor). Наименьшим общим предком элементов A и B является такой элемент C,

что C является предком A, C является предком B, при этом глубина C является наибольшей из возможных. При этом элемент считается своим собственным предком. Для каждого запроса выведите

наименьшего общего предка данных элементов.

Задания для самостоятельной работы.

1) Что означает ошибка `TypeError: unhashable type?`

1) Неверно задано значение

2) Тип данных нельзя использовать в роли ключа

3) Слишком большое значение

4) Ошибка синтаксиса

Ответ:

2) Какие типы данных нельзя использовать в роли ключа?

1) Список, словарь

2) Словарь, кортеж

3) Кортеж, число

4) Число, булево значение

Ответ:

3) Что выдаст этот код?

```
Another_dict = {'a': {'a': ['a']}}
```

```
Print(another_dict.pop('a') == another_dict.clear())
```

1) True

2) False

3) Ошибка

Ответ:

4) Каков будет результат кода?

```
Dict_1 = {'a': 10, 'b': 20}
```

```
Dict_2 = {'b': 20, 'c': 30}
```

```
Dict_1.update(dict_2)
```

```
Print(dict_1)
```

1) {'a': 10}

2) {'a': 10, 'b': 20}

3) {'a': 10, 'b': 20, 'c': 30}

4) {'b': 20, 'c': 30}

Ответ:

5) Что выведет этот код?

```
Old_dict = {'a': 10, 'b': 10}
```

```
New_dict = {}
```

```
For i, j in old_dict.items():
```

```
New_dict[j] = i
Print(new_dict)
1){'a': 10, 'b': 10}
2){10: 'a', 10: 'b'}
3){'a': 10}
4){10: 'b'}
```

Ответ:

Тема 8. Работа с датой и временем (ОПК-7)

Лекция.

Модуль `datetime` предоставляет классы для обработки времени и даты разными способами. Поддерживается и стандартный способ представления времени, однако больший упор сделан на простоту манипулирования датой, временем и их частями. Классы, предоставляемые модулем `datetime`: Класс `datetime.date(year, month, day)` - стандартная дата. Атрибуты: `year`, `month`, `day`. Неизменяемый объект. Класс `datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None)` - стандартное время, не зависит от даты. Атрибуты: `hour`, `minute`, `second`, `microsecond`, `tzinfo`. Класс `datetime.timedelta` - разница между двумя моментами времени, с точностью до микросекунд. Класс `datetime.tzinfo` - абстрактный базовый класс для информации о временной зоне (например, для учета часового пояса и / или летнего времени). Класс `datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)` - комбинация даты и времени. Обязательные аргументы: `datetime.MINYEAR (1) ≤ year ≤ datetime.MAXYEAR (9999)` `1 ≤ month ≤ 12` `1 ≤ day ≤ количество дней в данном месяце и году` Необязательные: `0 ≤ minute < 60` `0 ≤ second < 60` `0 ≤ microsecond < 1000000` Методы класса `datetime`: `datetime.today()` - объект `datetime` из текущей даты и времени. Работает также, как и `datetime.now()` со значением `tz=None`. `datetime.fromtimestamp(timestamp)` - дата из стандартного представления времени. `datetime.fromordinal(ordinal)` - дата из числа, представляющего собой количество дней, прошедших с 01.01.1970. `datetime.now(tz=None)` - объект `datetime` из текущей даты и времени. `datetime.combine(date, time)` - объект `datetime` из комбинации объектов `date` и `time`. `datetime.strptime(date_string, format)` - преобразует строку в `datetime` (так же, как и функция `strptime` из модуля `time`). `datetime.strftime(format)` - см. функцию `strftime` из модуля `time`. `datetime.date()` - объект даты (с отсечением времени). `datetime.time()` - объект времени (с отсечением даты). `datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]])` - возвращает новый объект `datetime` с изменёнными атрибутами. `datetime.timetuple()` - возвращает `struct_time` из `datetime`. `datetime.toordinal()` - количество дней, прошедших с 01.01.1970. `datetime.timestamp()` - возвращает время в секундах с начала эпохи. `datetime.weekday()` - день недели в виде числа, понедельник - 0, воскресенье - 6. `datetime.isoweekday()` - день недели в виде числа, понедельник - 1, воскресенье - 7. `datetime.isocalendar()` - кортеж (год в формате ISO, ISO номер недели, ISO день недели). `datetime.isoformat(sep='T')` - красивая строка вида "YYYY-MM-DDTHH:MM:SS.mmmmmmm" или, если `microsecond == 0`, "YYYY-MM-DDTHH:MM:SS" `datetime.ctime()` - см. `ctime()` из модуля `time`. Пример работы с классом `datetime`:

```
>>> >>> from datetime import datetime, date, time >>> # Using datetime.combine() >>> d = date(2005, 7, 14) >>> t = time(12, 30) >>> datetime.combine(d, t) datetime.datetime(2005, 7, 14, 12, 30) >>> # Using datetime.now() or datetime.utcnow() >>> datetime.now() datetime.datetime(2007, 12, 6, 16, 29, 43, 79043) # GMT +1 >>> datetime.utcnow() datetime.datetime(2007, 12, 6, 15, 29, 43, 79060) >>> # Using datetime.strptime() >>> dt = datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M") >>> dt datetime.datetime(2006, 11, 21, 16, 30) >>> # Using datetime.timetuple() to get tuple of all attributes >>> tt = dt.timetuple() >>> for it in tt: ... print(it) ... 2006 # year 11 # month 21 # day 16 # hour 30 # minute 0 # second 1 # weekday (0 = Monday) 325 # number of days since 1st January -1 # dst - method tzinfo.dst() returned None >>> # Date in ISO format >>> ic = dt.isocalendar() >>> for it in ic: ... print(it) ... 2006 # ISO year 47 # ISO week 2 # ISO weekday >>> # Formatting datetime >>> dt.strftime("%A, %d. %B %Y %I:%M%p") 'Tuesday, 21. November 2006 04:30PM'
```

Лабораторные работы.

1. По заданному числу `n` от 1 до 365 определите, на какое число какого месяца приходится день невисокосного года с номером `n`.

Программа получает на вход целое число n и должна вывести два числа: число месяца (от 1 до 31) и номер месяца (от 1 до 12), на которое приходится данный день.

2. Часы показывают время в формате hh:mm:ss. Определите количество секунд, которое прошло с начала суток.

Программа не должна содержать циклов для решения этой задачи.

3. Часы показывают время в формате hh:mm:ss. На этих часах запустили таймер, который прозвенит через n секунд. Определите время, которое будет на часах, когда прозвенит таймер.

n может принимать значения от 0 до 109. Решение задачи не должно содержать циклов. Постарайтесь также не использовать условную инструкцию.

4. В часах села батарейка и они стали идти вдвое медленней. Когда на часах было время $h1:m1:s1$, точное время было $h2:m2:s2$. Определите,

точное время, когда часы в следующий раз покажут время $h3:m3:s3$.

Решение задачи не должно использовать циклы.

Задания для самостоятельной работы.

1. Как получить текущую дату и время в Python?
2. Как получить текущую дату?
3. Что внутри datetime?
4. Как вывести час, минуту, секунду и микросекунду?

Тема 9. Пользовательские функции (ОПК-7)

Лекция.

Функции Python: 7 примеров. Базовые, встроенные и пользовательские функции Python → Полезные материалы по Python Теги: python, параметры, пользовательские функции, встроенные функции, базовые функции В этой статье мы просто приведём практические примеры работы функций в Python. Рассмотрим базовые, встроенные и пользовательские функции, а также функции с параметрами, возвращаемым значением и типом данных. Функции в Python представляют собой фрагменты кода в блоке, который имеет назначенное имя. Функции принимают ввод, осуществляют вычисления либо какое-нибудь действие и возвращают вывод. И, разумеется, функции упрощают работу с кодом, делая возможным его повторное использование. Базовые функции Python Давайте рассмотрим пример функции Python, принимающей 2 параметра, а также вычисляющей сумму и возвращающей вычисленное значение: `#определяем и объявляем функцию def calculate_sum(a,b):`
`sum = a+b return sum` #инструкция, приведённая ниже, называется вызовом функции
`print(calculate_sum(2,3))` # 5 Кроме того, в Python есть встроенные и пользовательские функции. Пользовательские функции Python Объявление пользовательской функции осуществляется с применением ключевого слова `def`. При этом оно должно сопровождаться именем пользовательской функции: `def calculate_si_amount(principal, rate, time): interest = (principal*rate*time)/100 return principal+interest` В данной функции окончательная сумма может быть рассчитана посредством использования простого процента к основной сумме. Именем функции является `Calculate_si_amount`. Что касается `principal`, `time` и `rate` — то это параметры, а функция возвращает рассчитанные данные. Для пользовательской функции можно не принимать возвращаемые значения и параметры. На нижеследующем примере мы видим пользовательскую функцию, не принимающую никаких параметров, зато возвращающую данные. `from random import seed, random from random import random def generate_random_number(): seed(10) return random()` Встроенные функции Python В Python существует много встроенных функций. Одна из наиболее часто используемых — `print()`. Её работа чрезвычайно проста: `print("Всем привет") print(len("Меня зовут Андрей"))` Ещё популярны такие функции, как `len()`, `abs()`, `sum()`, `str()`, `int()` и другие. Параметры функции в Python В языке программирования Python функция может иметь параметры по умолчанию: `def multiply(a, b=10): return a*b multiply(12) # 120 multiply(2, 3) # 6 multiply(b=9) # Ошибка: None*9 недопустимо` В вышеописанной функции, когда пользователь не задает 2-й параметр `b`, он предполагает, что параметр равен 10, однако при этом нужно предоставить 1-й параметр. Неизвестное количество параметров в функции Python Когда в функции, допустим, четыре параметра, а для второго параметра определено значение по умолчанию, то третьему и четвёртому параметрам тоже необходимо присвоить значение по умолчанию. Когда число параметров неизвестно, тогда в определение функции в качестве одного из параметров добавляется `*args`. Данный параметр ожидает кортеж. В нашем случае звёздочка (*) очень важна, т. к. название `args` просто является соглашением, то есть можно дать любое другое имя. `def calculate_sum(a, *args): sum = a for i in args: sum += i return sum calculate_sum(10) # 10 calculate_sum(10, 11, 12) # 33 calculate_sum(1, 2, 94, 6, 2, 8, 9, 20, 43, 2) # 187` Так же `**kwargs` ожидает словарь в качестве параметра. `def print_names(fl, ll, **kwargs): print(fl, ll, end=' ') for key in kwargs: print(key, kwargs[key], end=' ') print_names("andrey", "master") print_names("andrey", "master", alex="john", leon="elene") # andrey master andrey master alex john leon elene` Обратите внимание, что фрагмент выше имеет ссылку на цикл `for`. Тип данных для возвращаемого значения и параметров в Python Определение типов данных для параметров функции в Python может быть полезным: `def prime_numbers(x:int) -> (int, list): l=[] for i in range(x+1): if checkPrime(i): l.append(i) return len(l), l` В нашем примере определение функции указывает, что нужен 1 параметр типа `int` и вернёт два значения типа `list` и `int` соответственно. Возвращаемое значение функции в Python Язык программирования Python даёт возможность функции возвращать несколько значений. `def prime_numbers(x): l=[] for i in range(x+1): if checkPrime(i): l.append(i) return len(l), l no_of_primes, primes_list = prime_numbers(100)` В нашем случае возвращаются 2 значения. Если данная функция вызывается, то возвращаемые значения сохраняются одновременно в 2-х переменных. Если же функция не возвращает ничего, то она неявно возвращает `None`.

Лабораторные работы.

- 1.Реализовать функцию `is_sorted`. Функция принимает на вход список и возвращает `True`, если элементы в нем упорядочены по возрастанию.
- 2.Реализовать функцию `find_longest`. Функция принимает на вход список строк и возвращает строку с максимальной длиной. Если таких строк несколько, то возвращает первую из них.

3.Реализовать функцию `min_max`. Функция принимает на вход список чисел и возвращает пару: минимальное и максимальное число. Алгоритм должен выполнять не более одного прохода по списку.

4.Реализовать функцию `zip_longest_sum`. Функция принимает на вход два списка и возвращает новый список, полученный сложением соответствующих элементов входных списков.

5.Реализовать функцию `is_prime`. Функция принимает на вход целое число и возвращает `True`, если число простое. Простым называется число, которое делится без остатка только на 1 и на себя.

6.Реализовать функцию `join`. Функция принимает разделитель и список строк, возвращает строку, в которой элементы списка чередуются с разделителем.

Задания для самостоятельной работы.

1) Как можно вызвать метод `func` у следующего класса (выберите все подходящие варианты):

```
Class myclass:
```

```
Def func(self):
```

```
Print('hello')
```

- 1) `myClass.func()`
- 2) `obj = myClass() obj.func()`
- 3) `obj = myClass() myClass.func(obj)`
- 4) `obj = myClass() obj.func`
- 5) ни один из перечисленных

Ответ:

2) Что напечатает следующий код:

```
def func(n):
```

```
    n = n + 1
```

```
print(func(0))
```

- 1)0
- 2)1
- 3)func(0)
- 4)None
- 5)возникнет ошибка

Ответ:

3) Что выведет следующий код:

```
a = 3
```

```
a = "foo" if a / 2 == 1 else 2
```

```
a = a + a
```

```
print (a)
```

- 1)6
- 2)Возникнет ошибка
- 3)2
- 4)4
- 5)foofoo

Ответ:

4) Что необходимо добавить на место пропущенной строки?

```
def find_max(nums):
```

```
    max_num = float("-inf")
```

```
    for num in nums:
```

```
        if num > max_num:
```

```
            # пропущенная строка
```

```
    return max_num
```

- 1) `max_num = num`

2) num = max_num
3) max_num += 1
4) max_num += num

Ответ:

5) Каким будет результат выполнения кода:

```
a = [1, 2, 3]
```

```
if a[2] < 3:
```

```
    print(a[a[1]])
```

```
else:
```

```
    print(a[1])
```

1) 1

2) 2

3) 3

4) возникнет ошибка

Ответ:

Тема 10. Обработка исключений (ОПК-7)

Лекция.

Исключения (exceptions) - ещё один тип данных в python. Исключения необходимы для того, чтобы сообщать программисту об ошибках. Самый простейший пример исключения - деление на ноль:

```
>>> >>> 100 / 0
Traceback (most recent call last): File "", line 1, in 100 / 0
ZeroDivisionError: division by zero
```

Разберём это сообщение подробнее: интерпретатор нам сообщает о том, что он поймал исключение и напечатал информацию (Traceback (most recent call last)). Далее имя файла (File ""). Имя пустое, потому что мы находимся в интерактивном режиме, строка в файле (line 1); Выражение, в котором произошла ошибка (100 / 0). Название исключения (ZeroDivisionError) и краткое описание исключения (division by zero). Разумеется, возможны и другие исключения: >>> >>> 2 + '1'

```
Traceback (most recent call last): File "", line 1, in 2 + '1'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> int('qwerty')
Traceback (most recent call last): File "", line 1, in int('qwerty')
ValueError: invalid literal for int() with base 10: 'qwerty'
```

В этих двух примерах генерируются исключения TypeError и ValueError соответственно. Подсказки дают нам полную информацию о том, где порождено исключение, и с чем оно связано. Рассмотрим иерархию встроенных в python исключений, хотя иногда вам могут встретиться и другие, так как программисты могут создавать собственные исключения. Данный список актуален для python 3.3, в более ранних версиях есть незначительные изменения. BaseException - базовое исключение, от которого берут начало все остальные. SystemExit - исключение, порождается функцией sys.exit при выходе из программы. KeyboardInterrupt - порождается при прерывании программы пользователем (обычно сочетанием клавиш Ctrl+C). GeneratorExit - порождается при вызове метода close объекта generator. Exception - а вот тут уже заканчиваются полностью системные исключения (которые лучше не трогать) и начинаются обыкновенные, с которыми можно работать. StopIteration - порождается встроенной функцией next, если в итераторе больше нет элементов. ArithmeticError - арифметическая ошибка. FloatingPointError - порождается при неудачном выполнении операции с плавающей запятой. На практике встречается нечасто. OverflowError - возникает, когда результат арифметической операции слишком велик для представления. Не появляется при обычной работе с целыми числами (так как python поддерживает длинные числа), но может возникать в некоторых других случаях. ZeroDivisionError - деление на ноль. AssertionError - выражение в функции assert ложно. AttributeError - объект не имеет данного атрибута (значения или метода). BufferError - операция, связанная с буфером, не может быть выполнена. EOFError - функция наткнулась на конец файла и не смогла прочитать то, что хотела. ImportError - не удалось импортировать модуль или его атрибута. LookupError - некорректный индекс или ключ. IndexError - индекс не входит в диапазон элементов. KeyError - несуществующий ключ (в словаре, множестве или другом объекте). MemoryError - недостаточно памяти. NameError - не найдено переменной с таким именем. UnboundLocalError - сделана ссылка на локальную переменную в функции, но переменная не определена ранее. OSError - ошибка, связанная с системой. BlockingIOError ChildProcessError - неудача при операции с дочерним процессом. ConnectionError - базовый класс для исключений, связанных с подключениями. BrokenPipeError ConnectionAbortedError ConnectionRefusedError ConnectionResetError FileExistsError - попытка создания файла или директории, которая уже существует. FileNotFoundError - файл или директория не существует. InterruptedError - системный вызов прерван входящим сигналом. IsADirectoryError - ожидался файл, но это директория. NotADirectoryError - ожидалась директория, но это файл. PermissionError - не хватает прав доступа. ProcessLookupError - указанного процесса не существует. TimeoutError - закончилось время ожидания. ReferenceError - попытка доступа к атрибуту со слабой ссылкой. RuntimeError - возникает, когда исключение не попадает ни под одну из других категорий. NotImplementedError - возникает, когда абстрактные методы класса требуют переопределения в дочерних классах. SyntaxError - синтаксическая ошибка. IndentationError - неправильные отступы. TabError - смешивание в отступах табуляции и пробелов. SystemError - внутренняя ошибка. TypeError - операция применена к объекту несоответствующего типа. ValueError - функция получает аргумент правильного типа, но некорректного значения. UnicodeError - ошибка, связанная с кодированием / раскодированием unicode в строках. UnicodeEncodeError - исключение, связанное с кодированием unicode. UnicodeDecodeError - исключение, связанное с декодированием unicode. UnicodeTranslateError - исключение, связанное с переводом unicode. Warning - предупреждение. Теперь, зная, когда и при каких обстоятельствах могут возникнуть исключения, мы можем их обрабатывать. Для обработки исключений используется конструкция try - except. Первый пример применения этой конструкции: >>> >>> try: ... k = 1 / 0 ... except ZeroDivisionError: ... k = 0 ... >>> print(k) 0 В блоке try мы выполняем инструкцию, которая может породить исключение, а в блоке except мы перехватываем их. При этом перехватываются как само исключение, так и его потомки. Например, перехватывая ArithmeticError, мы также перехватываем FloatingPointError, OverflowError, ZeroDivisionError. >>> >>> try: ... 1 / 0 ... except (ArithmeticError, ZeroDivisionError): ... pass

Лабораторные работы.

1. Написать программу, которая будет писать в консоль названия нажатых клавиш. Реализовать поддержку enter, space, w, a, s, d, esc.
2. С помощью циклов, используя квадрат 10x10 пикселей и его след, нарисовать рамку 100x100 пикселей.
3. Написать программу, в которой по нажатию клавиш будет двигаться квадрат размером 20x20 пикселей. Учесть, что квадрат не должен выходить за границы экрана.
4. Доработать программу, чтобы квадрат при каждом движении менял свой цвет на случайный.

Задания для самостоятельной работы.

1. Что означает исключение в python?
2. Какие стандартные исключения в Python вы знаете?
3. Блок множественного исключения

4. Контроль знаний обучающихся и типовые оценочные средства

4.1. Распределение баллов:

1 семестр

- посещаемость – 10 баллов
- текущий контроль – 81 балл
- контрольные срезы – 2 среза: 4 балла, 5 баллов
- премиальные баллы – 20 баллов

Распределение баллов по заданиям:

№ темы	Название темы / вид учебной работы	Формы текущего контроля / срезы	Мак. кол-во баллов	Методика проведения занятия и оценки

1.	Переменные. Типы данных. Преобразование типов данных	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

2.	Условные операторы и циклы	Собеседование(контрольный срез)	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 3 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 2 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

3.	Строки и двоичные данные	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.
4.	Функции и методы для работы со строками	Собеседование	4	- рациональность использованных приемов и способов решения поставленной учебной задачи;
		Тестирование(контрольный срез)	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

5.	Списки. Операции над списками	Собеседо вание	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестиров ание	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

6.	Кортежи, множества и диапазоны.	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

7.	Словари. Операции и методы для работы со словарями	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

8.	Работа с датой и временем	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

9.	Пользовательские функции	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 4 баллов – студент умеет сопоставить полученную при подготовке к практическому занятию информацию, сравнивать разные точки зрения на анализируемую проблему, уметь четко формулировать свои вопросы и отвечать на задаваемые ему вопросы 3 баллов - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балла – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.

10.	Обработка исключений	Собеседование	4	Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д. Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке: - правильность ответа по содержанию; - полнота и глубина ответа; - сознательность ответа; - логика изложения материала; - рациональность использованных приемов и способов решения поставленной учебной задачи; - своевременность и эффективность использования наглядных пособий и технических средств при ответе; - использование дополнительного материала; - рациональность использования времени, отведенного на задание. 2 балла - студент умеет применять полученную при подготовке к практическому занятию информацию, отвечать на большинство вопросов, вести дискуссию . 1 балл – студент владеет теоретическим материалом по теме практического занятия, иногда затрудняется при ответе на вопросы, не умеет сформулировать свою точку зрения на обсуждаемую проблему Если студент не владеет проблематикой практического занятия, не может отвечать на вопросы, зачитывает ответ по напечатанному тексту – ответ баллами не оценивается.
		Тестирование	5	Оценка теста по текущему разделу или теме дисциплины 5 баллов – студент правильно отвечает на 50-100% вопросов в тесте. 2 балла - студент правильно отвечает на 25-50% вопросов в тесте.
11.	Посещаемость		10	10 баллов – студент посетил все 100% занятий 7-9 баллов – студент посетил не менее 80% занятий 4-6 баллов – студент посетил не менее 50% занятий 1-3 балла – студент посетил не менее 25% занятий Если студент посетил менее 25% занятий, баллы не начисляются
12.	Премияльные баллы		20	Дополнительные премияльные баллы могут быть начислены: - за проект, выполненный по заказу работодателя и реализованный на практике – 20 баллов; - постоянная активность во время практических занятий – 10 баллов; - полностью подготовленная к публикации статья по тематике в рамках дисциплины – 10 баллов; - участие с докладом во всероссийской олимпиаде по тематике изучаемой дисциплины – 20 баллов; - участие в выставке по тематике изучаемой дисциплины – 20 баллов; - публикация статьи по тематике изучаемой дисциплины в сборнике студенческих работ / материалах всероссийской конференции / журнале из перечня ВАК – 10 / 15 / 20
13.	Индивидуальные задания, с помощью которых можно набрать дополнительные баллы		20	Решение кейса (10 баллов) Прохождение тестирования (30 вопросов) по всему курсу дисциплины (10 баллов)
14.	Итого за семестр		100	

Итоговая оценка по зачету выставляется в 100-балльной шкале и в традиционной четырехбалльной шкале. Перевод 100-балльной рейтинговой оценки по дисциплине в традиционную четырехбалльную осуществляется следующим образом:

100-балльная система	Традиционная система
50 - 100 баллов	Зачтено
0 - 49 баллов	Не зачтено

4.2 Типовые оценочные средства текущего контроля

Собеседование

Тема 1. Переменные. Типы данных. Преобразование типов данных

- 1)Что такое переменная в Python?
- 2)Как указать значение переменной в Python?
- 3)Какие типы данных вы знаете?
- 4)Какие виды числового типа данных вы знаете?
- 5)Как преобразовывать типы данных?

Тема 2. Условные операторы и циклы

1. Какие условные операторы и циклы вы знаете?
2. Что делает оператор цикла выполняет?
3. Для чего предназначены операторы break и continue?
4. Для чего используется конструкция if – elif – else?

Тема 3. Строки и двоичные данные

1. Строки в python обозначаются кавычками. Приведите все способы.
2. Какие типы данных можно преобразовать в строку?
3. Опишите синтаксис срезов строк при помощи квадратных скобок.
4. Как применяют операции сложения и умножения к строкам?
5. Что такое двоичные данные?

Тема 4. Функции и методы для работы со строками

- 4.1.Арифметические выражения.
- 4.2. Логические выражения.
- 4.3.Строковые выражения.

Тема 5. Списки. Операции над списками

1. Перечислите характеристики типа данных «список», которые вы знаете.
2. Как объединить списки?
3. Как умножать списки?
4. Как перевернуть список?
5. В чём разница между append и extend?

Тема 6. Кортежи, множества и диапазоны.

1. Чем список отличается от других структур?
2. Как объединить два списка в список кортежей?
3. Как работает функция range?
4. В каких ситуациях лучше использовать списки, а в каких кортежи, словари или множества?

Тема 7. Словари. Операции и методы для работы со словарями

1. Какими способами можно получить доступ к значению ключа, не изменяя при этом словарь?
2. Что может служить ключом словаря? Перечислите структуры данных и общие требования к именованию.
3. Для чего нужен метод `pop()`? Какие параметры он может принимать? Приведите пример использования.
4. Охарактеризуйте методы `keys()`, `items()`, `values()`. Что они возвращают, какова специфика результирующих объектов?
5. В чем опасность копирования словаря? Для чего нужно глубокое копирование?

Тема 8. Работа с датой и временем

1. Как получить текущую дату и время в Python?
2. Как получить текущую дату?
3. Что внутри `datetime`?
4. Как вывести час, минуту, секунду и микросекунду?

Тема 9. Пользовательские функции

1. Что такое пользовательская функция?
2. Чем служит ключевое слово `def`?
3. Для чего используется слово `return`?
4. Приведите пример функции Python, которая принимает два параметра, вычисляет сумму и возвращает вычисленное значение.

Тема 10. Обработка исключений

1. Что означает исключение в python?
2. Какие стандартные исключения в Python вы знаете?
3. Блок множественного исключения

Тестирование

Тема 1. Переменные. Типы данных. Преобразование типов данных

1) Чему равно значение `z`:

```
x, y, z = 32 43 11
```

- 1) 32
- 2) `z`
- 3) 11
- 4) Код не выполнится из-за ошибки

Ответ:

2) Какой тип данных вернет этот код `print(type(88))`

- 1) `integer`
- 2) `str`
- 3) `int`
- 4) `float`

Ответ:

3) Что выведет этот код:

```
x = 685.0
print(x)
```

- 1) Ошибку, переменная должна быть длиннее 3х символов
- 2) 685
- 3) Ошибку, десятичную часть нужно отделять запятой
- 4) 685.0
- 5) "685.0"
- 4) Как создать переменную с типом float?
 - 1) var = "9846"
 - 2) var = "9846.499"
 - 3) var = 9846
 - 4) var = 9846.495
- 5) Какие типы данных из урока могут содержать цифры?
 - 1) Все
 - 2) bool, str
 - 3) int, float, bool
 - 4) int, float
 - 5) int, float, str

Тема 2. Условные операторы и циклы

- 1) Что выведет программа?

```
if 4>8:
    print(6)
else:
    print(3)
```

Ответ:

- 2) Что выведет программа?

```
a = -10
b = -1
if a<b:
    print(a)
else:
    print(b)
```

Ответ:

- 3) Что выведет программа?

```
a = False
b = False
if a or b:
    print(1)
else:
    print(2)
```

Ответ:

- 4) Что выведет программа?

```
if True:
    print(1)
else:
    print(0)
```

Ответ:

- 5) Что выведет программа?

```
a = False
b = True
```

```

if a and b:
    print(1)
else:
    print(2)

```

Ответ:

Тема 3. Строки и двоичные данные

1) Что будет выведено в результате выполнения данной программы:

```

def string_length(str1):
    count = 0
    for char in str1:
        count += 1
    return count
print(string_length('w3resource.com'))

```

- 1) Длина строки прописью
- 2) Длина строки цифрой
- 3) Количество букв в строке
- 4) Количество цифр в строке

2) Что будет результатом выполнения данной программы:

```

def chars_mix_up(a, b):
    new_a = b[2:] + a[2:]
    new_b = a[2:] + b[2:]
    return new_a + ' ' + new_b
print(chars_mix_up('abc', 'xyz'))

```

- а) Первые буквы строк поменяются местами
- б) Последние буквы строк поменяются местами
- в) Поменяет строки местами
- г) Объединит строки в одну и перемешает

3) Что будет выведено в результате выполнения данной программы:

```

def find_longest_word(words_list):
    word_len = []
    for n in words_list:
        word_len.append((len(n), n))
    word_len.sort()
    return word_len[-1][0], word_len[-1][1]
result = find_longest_word(["PHP", "Exercises", "Backend"])
print("\nLongest word: ", result[1])
print("Length of the longest word: ", result[0])

```

- а) Длина строки цифрами и прописью
- б) Длина самого длинного слова строки и само слово
- в) Длина самого короткого слова и само слово
- г) Самое длинное слово и длина строки

Тема 4. Функции и методы для работы со строками

1. Компонент Combobox-это

- 1) упорядоченная совокупность взаимосвязанных элементов, являющихся текстовыми строками
- 2) контейнер, в котором можно размещать другие элементы управления

3) многострочный редактор

4) переключатель с зависимой фиксацией

2. Для работы с комбинированным списком в Delphi служит компонент

1) Combobox

2) ListBox

3) Radiogroup

4) Stringgid

5) Checklistbox

3. Свойство Style типа TComboboxstyle определяет

1) внешний вид и поведение комбинированного списка

2) свойства комбинированного списка

3) стиль работы с комбинированным списком

4) цвет комбинированного списка

5) число записей в комбинированном списке

4. Свойство DropDownCount типа Integer определяет

1) количество строк, одновременно отображающиеся в раскрывающемся списке

2) номер выделенного элемента

3) число выделенных компонентов на форме

4) размеры выделенного компонента

5) раскрыт ли список

5. Свойство DroppedDown типа Boolean позволяет определить

1) раскрыт ли список

2) выделен ли список

3) число выделенных компонентов на форме

4) размеры выделенного компонента

5) количество удаленных записей

6. Отсчет элементов списка начинается с

1) с нуля

2) с единицы

3) с минус единицы

4) с двух

5) с минус двух

7. Какое свойство представляет собой массив строк

и определяет количество элементов списка и их содержимое

1) items

2) count

3) index

4) font

5) hint

8. Свойство count типа integer

1) задает число элементов в списке

2) определяет номер выделенного элемента

3) определяет количество удаленных записей

4) задает число выделенных компонентов на форме

5) определяет внешний вид и поведение комбинированного списка

9. Процедура Insert(Index:Integer; const S:String)

1) вставляет строку S на позицию с номером, заданным параметром Index

2) добавляет в конец списка строку

3) удаляет элемент с номером, заданным параметром Index

4) нет верного ответа

5) очищает список, удаляя все его элементы

10.Функция Add (const S:string)

1) вставляет строку S на позицию с номером, заданным параметром Index

2) добавляет в конец списка строку

3) удаляет элемент с номером, заданным параметром Index

4) нет верного ответа

5) очищает список, удаляя все его элементы

11.Процедура Delete(Index:Integer; const S:String)

1) вставляет строку S на позицию с номером, заданным параметром Index

2) добавляет в конец списка строку

3) удаляет элемент с номером, заданным параметром Index

4) нет верного ответа

5) очищает список, удаляя все его элементы

12.Процедура Clear

1) вставляет строку S на позицию с номером, заданным параметром Index

2) добавляет в конец списка строку

3) удаляет элемент с номером, заданным параметром Index

4) нет верного ответа

5) очищает список, удаляя все его элементы

13. Процедура IndexOf(const S:string):integer

1) определяет, содержится ли в списке строка S

2) вставляет строку S на позицию с номером, заданным параметром Index

3) добавляет в конец списка строку

4) удаляет элемент с номером, заданным параметром Index

5) нет верного ответа

Тема 5. Списки. Операции над списками

1)Как создать список?

1)Все варианты верны

2)L = list(1, 2, 3)

3)l = [1, 2, 3]

4)l = list[1, 2, 3]

Ответ:

2) Что выведет этот код:

```
a = [ 1, 342, 223, 'Африка', 'Очки']
```

```
print(a[-3])
```

1)223

2)'Африка'

3)342

4)Ошибку

Ответ:

3) Что выведет этот код:

```
sample = [10, 20, 30]
```

```
sample.append(60)
```

```
sample.insert(3, 40)
```

```
print(sample)
```

1)[10, 20, 30, 40]

2)[10, 20, 30, 40, 60]

3)[10, 20, 30, 60, 40]

4)[60, 10, 20, 30, 40]

Ответ:

4) Что из перечисленного правда?

1)Элементы списка не могут повторяться

2)Все элементы списка должны быть одного типа

3)Мы можем вставить элемент на любую позицию в списке

4)Список не может содержать вложенных списков

Ответ:

5) Как получить ['bar', 'baz'] из списка

```
a = ['foo', 'bar', 'baz', 'qux', 'quux']
```

?

1)print(a[2:4])

2)print(a[1], a[2])

3)print(a[1:-2])

4)print(a[-4:-3])

5)print(a[2:3])

Ответ:

Тема 6. Кортежи, множества и диапазоны.

1)Что такое множество в Python?

1)Это любая коллекция элементов

- 2) Это список, содержащий в себе только функции
- 3) Это контейнер, значения в котором не повторяются
- 4) Это список, содержащий вложенные списки в себе

Ответ:

2) Каким образом правильно объявляется множество?

- 1) `a = {}`
- 2) `a = []`
- 3) `a = set()`
- 4) `a = set`

Ответ:

3) Чем отличаются методы `remove()` и `discard()`, применяемые к множеству?

- 1) Ничем
- 2) `remove()` удаляет элемент если он есть, но бросает ошибку если элемента нет. `discard()` просто удаляет элемент если он есть
- 3) `discard()` удаляет элемент если он есть, но бросает ошибку если элемента нет. `remove()` просто удаляет элемент если он есть
- 4) Метода `discard()` для множеств не существует

Ответ:

4) Что такое `frozenset`?

- 1) Множество, которое нельзя изменять
- 2) Множество, которое хранит в себе только неизменяемые объекты
- 3) Множество, которое используется для хранения констант
- 4) Выдумка нашей редакции

Ответ:

5) Что это за метод такой, `set.difference(another_set)`

- 1) Возвращает `True`, если два множества одинаковые, `False` если хотя бы один элемент не совпадает
- 2) Возвращает `True`, если два множества разные, `False` если хотя бы один элемент совпадает
- 3) Возвращает множество из элементов, которые встречаются только в множестве `set`
- 4) Возвращает множество из элементов, которые встречаются только в множестве `another_set`

Ответ:

Тема 7. Словари. Операции и методы для работы со словарями

1) Что означает ошибка `TypeError: unhashable type`?

- 1) Неверно задано значение
- 2) Тип данных нельзя использовать в роли ключа
- 3) Слишком большое значение
- 4) Ошибка синтаксиса

Ответ:

2) Какие типы данных нельзя использовать в роли ключа?

- 1) Список, словарь
- 2) Словарь, кортеж
- 3) Кортеж, число
- 4) Число, булево значение

Ответ:

3) Что выдаст этот код?

```
Another_dict = {'a': {'a': ['a']}}
```

```
Print(another_dict.pop('a') == another_dict.clear())
```

- 1) True
- 2) False
- 3) Ошибка

Ответ:

- 4) Каков будет результат кода?

```
Dict_1 = {'a': 10, 'b': 20}
```

```
Dict_2 = {'b': 20, 'c': 30}
```

```
Dict_1.update(dict_2)
```

```
Print(dict_1)
```

- 1) {'a': 10}
- 2) {'a': 10, 'b': 20}
- 3) {'a': 10, 'b': 20, 'c': 30}
- 4) {'b': 20, 'c': 30}

Ответ:

- 5) Что выведет этот код?

```
Old_dict = {'a': 10, 'b': 10}
```

```
New_dict = {}
```

```
For i, j in old_dict.items():
```

```
New_dict[j] = i
```

```
Print(new_dict)
```

- 1) {'a': 10, 'b': 10}
- 2) {10: 'a', 10: 'b'}
- 3) {'a': 10}
- 4) {10: 'b'}

Ответ:

Тема 8. Работа с датой и временем

- 1) Определите что будет выведено после выполнения данного кода:

```
from datetime import datetime, timedelta
```

```
given_date = datetime(2020, 2, 25)
```

```
print("Given date")
```

```
print(given_date)
```

```
days_to_subtract = 7
```

```
res_date = given_date - timedelta(days=days_to_subtract)
```

```
print("New Date")
```

```
print(res_date)
```

- а) Все даты прошлой недели
- б) Все дни прошлой недели
- в) Дата неделю назад
- г) День недели неделю назад

- 2) Определите что будет выведено после выполнения данного кода:

```
from datetime import datetime
```

```
given_date = datetime(2020, 7, 26)
```

```
print(given_date.today().weekday())
```

```
print(given_date.strftime('%A'))
```

- а) Проверка заданной даты на четность
- б) День недели заданного числа

в) Проверка является ли заданная дата выходным

г) Является ли четной неделя, в которой находится заданная дата

3) Определите что будет выведено после выполнения данного кода:

```
from datetime import datetime
given_date = datetime(2020, 2, 25)
string_date = given_date.strftime("%Y-%m-%d %H:%M:%S")
print(string_date)
```

а) Заданная дата в измененном формате (другой внешний вид)

б) Заданная дата в типе str

в) Заданная дата через неделю

г) Данная дата в типе int для проведения вычислений

4) Определите что будет выведено в результате выполнения данного кода:

```
from datetime import datetime
date_1 = datetime(2020, 2, 25).date()
date_2 = datetime(2020, 9, 17).date()
delta = None
```

```
if date_1 > date_2:
    print("date_1 is greater")
    delta = date_1 - date_2
```

```
else:
    print("date_2 is greater")
    delta = date_2 - date_1
```

```
print("Difference is", delta.days, "days")
```

а) Количество дней между сегодняшним днем и указанными датами

б) Количесво дней между указанными днями

в) Какая дата "старше" (какое число дальше от начала календаря)

г) Какая дата "младше" (какое число ближе к началу календаря)

****За начало календаря принимать Рождество Христово**

5) Определите что будет выведено в результате выполнения данного кода:

```
from datetime import datetime, timedelta
given_date = datetime(2020, 3, 22, 10, 00, 00)
print("Given date")
print(given_date)
days_to_add = 7
res_date = given_date + timedelta(days=days_to_add, hours=12)
print("New Date")
print(res_date)
```

а) Указанная дата с добавлением 1 недели и 12 часов

б) Указанная дата с добавлением 12 часов

в) Указанная дата с вычетом 1 недели и 12 часов

г) Указанная дата с вычетом 12 часов и добавлением одной недели

Тема 9. Пользовательские функции

1) Как можно вызвать метод func у следующего класса (выберите все подходящие варианты):

```
Class myclass:
```

```
Def func(self):
```

```
Print('hello')
```

1) myClass.func()

2) obj = myClass() obj.func()

3) `obj = myClass() myClass.func(obj)`

4) `obj = myClass() obj.func`

5) ни один из перечисленных

Ответ:

2) Что напечатает следующий код:

```
def func(n):
```

```
    n = n + 1
```

```
print(func(0))
```

1)0

2)1

3)func(0)

4)None

5)возникнет ошибка

Ответ:

3) Что выведет следующий код:

```
a = 3
```

```
a = "foo" if a / 2 == 1 else 2
```

```
a = a + a
```

```
print (a)
```

1)6

2)Возникнет ошибка

3)2

4)4

5)foofoo

Ответ:

4) Что необходимо добавить на место пропущенной строки?

```
def find_max(nums):
```

```
    max_num = float("-inf")
```

```
    for num in nums:
```

```
        if num > max_num:
```

```
            # пропущенная строка
```

```
    return max_num
```

1)max_num = num

2)num = max_num

3)max_num += 1

4)max_num += num

Ответ:

5) Каким будет результат выполнения кода:

```
a = [1, 2, 3]
```

```
if a[2] < 3:
```

```
    print(a[a[1]])
```

```
else:
```

```
    print(a[1])
```

1)1

2)2

3)3

4)возникнет ошибка

Ответ:

Тема 10. Обработка исключений

1. Какое исключение возникнет при вводе кода:

```
>>> 1/0
```

- 1) TypeError
- 2) ZeroDivisionError +
- 3) ValueError
- 4) NameError

2. Какое выражение позволяет получить исключение и повторно поднять его?

- 1) try
- 2) raise +
- 3) expect
- 4) traceback

3. Что выведет данный код?

```
my_dict = {"a":1, "b":2, "c":3}
```

```
try:
```

```
    value = my_dict["a"]
```

```
except KeyError:
```

```
    print("A KeyError occurred")
```

```
else:
```

```
    print("No error occurred")
```

```
finally:
```

```
    print("The finally statement ran")
```

- 1) No error occurred
The finally statement ran +
- 2) A KeyError occurred
- 3) The finally statement ran
- 4) ValueError

4. Что выведет данный код при a = 10, b = 5?

```
try:
```

```
    a = input("Введите число: ")
```

```
    b = input("Введите еще одно число: ")
```

```
    a = int(a)
```

```
    b = int(b)
```

```
    print (a / b)
```

```
except ZeroDivisionError:
```

```
    print("b не может быть нулем!")
```

```
except ValueError:
```

```
    print("Ошибка ввода числа")
```

- 1) b не может быть нулем
- 2) NameError
- 3) 2 +
- 4) Ошибка ввода числа

5. Что выведет данный код при a = 1, b = 0?

```
try:
```

```
    print(a/b)
```

```
    print("Это не будет напечатано")
```

```
    print('10'+10)
```

```
except TypeError:
```

```
print("Вы сложили значения несовместимых типов")
except ZeroDivisionError:
    print("Деление на 0")
1) Деление на 0 +
2) Вы сложили значения несовместимых типов
3) NameError
4) Это не будет напечатано
```

4.3 Промежуточная аттестация по дисциплине проводится в форме зачета

Типовые вопросы зачета (ОПК-7)

1. Основные характеристики и критерии оценки алгоритмов. Данные. Понятие типа данных.
2. Понятие типа данных. Константы. Переменные.
3. Основные характеристики и критерии оценки алгоритмов. Целочисленные типы данных.
4. Вещественные типы данных.
5. Основные характеристики и критерии оценки алгоритмов. Символьные типы данных.
6. Булевы типы данных.
7. Определение новых типов данных.
8. Основные характеристики и критерии оценки алгоритмов. Перечисляемые типы данных.
9. Интервальные типы данных.
10. Временной тип данных.
11. Операции. Выражения. Арифметические операции.
- 12 Операции. Операции отношения.
- 13 Операции. Булевы операции.
- 14 Основные характеристики и критерии оценки алгоритмов. Оператор присваивания.
- 15 Оператор ветвления if.
- 16 Оператор ветвления case.
- 17 Операторы повтора — циклы. Оператор повтора for.
- 18 Операторы повтора — циклы. Оператор повтора repeat
- 19 Операторы повтора — циклы. Оператор повтора while
20. Процедуры. Понятие. Свойства. Параметры.
21. Функции. Понятие. Свойства. Параметры.
22. Параметры процедур и функций.
23. Рекурсивные подпрограммы.
24. Строковые переменные.
25. Операции над строками.
26. Форматы кодирования символов.
27. Стандартные процедуры и функции для работы со строками.
28. Объявление массива.
29. Работа с массивами.
30. Понятие файла.
31. Работа с файлами.
32. Стандартные подпрограммы управления файлами.

Типовые задания для зачета (ОПК-7)

Каждому изучающему Python нужно писать код для закрепления. Вашему вниманию предлагаются несколько задач для реализации (не слишком простых (кроме первой) и не слишком сложных). Для выполнения заданий крайне рекомендуется пройти самоучитель. Также для этих задач есть репозиторий с тестами и моими решениями (чтобы проверить себя).

Для запуска тестов для вашей функции проще всего будет добавить код из папки с тестами в конец файла с функцией.

А теперь, собственно, задачи:

Простейшие арифметические операции (1)

Написать функцию `arithmetic`, принимающую 3 аргумента: первые 2 - числа, третий - операция, которая должна быть произведена над ними. Если третий аргумент `+`, сложить их; если `-`, то вычесть; `*` — умножить; `/` — разделить (первое на второе). В остальных случаях вернуть строку "Неизвестная операция".

Високосный год (2)

Написать функцию `is_year_leap`, принимающую 1 аргумент — год, и возвращающую `True`, если год високосный, и `False` иначе.

Квадрат (3)

Написать функцию `square`, принимающую 1 аргумент — сторону квадрата, и возвращающую 3 значения (с помощью кортежа): периметр квадрата, площадь квадрата и диагональ квадрата.

Времена года (4)

Написать функцию `season`, принимающую 1 аргумент — номер месяца (от 1 до 12), и возвращающую время года, которому этот месяц принадлежит (зима, весна, лето или осень).

Банковский вклад (5)

Пользователь делает вклад в размере `a` рублей сроком на `years` лет под 10% годовых (каждый год размер его вклада увеличивается на 10%. Эти деньги прибавляются к сумме вклада, и на них в следующем году тоже будут проценты).

Написать функцию `bank`, принимающая аргументы `a` и `years`, и возвращающую сумму, которая будет на счету пользователя.

Простые числа (6)

Написать функцию `is_prime`, принимающую 1 аргумент — число от 0 до 1000, и возвращающую `True`, если оно простое, и `False` - иначе.

Правильная дата (7)

Написать функцию `date`, принимающую 3 аргумента — день, месяц и год. Вернуть `True`, если такая дата есть в нашем календаре, и `False` иначе.

XOR-шифрование (8)

Написать функцию `XOR_cipher`, принимающая 2 аргумента: строку, которую нужно зашифровать, и ключ шифрования, которая возвращает строку, зашифрованную путем применения функции `XOR (^)` над символами строки с ключом. Написать также функцию `XOR_uncipher`, которая по зашифрованной строке и ключу восстанавливает исходную строку.

4.4. Шкала оценивания промежуточной аттестации

Оценка	Компетенции	Дескрипторы (уровни) – основные признаки освоения (показатели достижения результата)
«зачтено» (50 - 100 баллов)	ОПК-7	Демонстрирует высокий уровень знаний теоретических и практических знания в области программирования на языке Python. Способен применять методы программирования для решения задач профессиональной деятельности. Обладает навыком создания и разработки интерфейса и функциональности приложения.
«не зачтено» (0 - 49 баллов)	ОПК-7	Не имеет теоретических и практических знания в области программирования на языке Python. Не способен частично применять методы программирования для решения задач профессиональной деятельности. Не обладает навыком создания и разработки интерфейса и функциональности приложения.

5. Методические указания для обучающихся по освоению дисциплины (модуля)

5.1 Методические указания по организации самостоятельной работы обучающихся:

Приступая к изучению дисциплины, в первую очередь обучающимся необходимо ознакомиться содержанием рабочей программы дисциплины (РПД), которая определяет содержание, объем, а также порядок изучения и преподавания учебной дисциплины, ее раздела, части.

Для самостоятельной работы важное значение имеют разделы «Объем и содержание дисциплины», «Учебно-методическое и информационное обеспечение дисциплины» и «Материально-техническое обеспечение дисциплины, программное обеспечение, профессиональные базы данных и информационные справочные системы».

В разделе «Объем и содержание дисциплины» указываются все разделы и темы изучаемой дисциплины, а также виды занятий и планируемый объем в академических часах.

В разделе «Учебно-методическое и информационное обеспечение дисциплины» указана рекомендуемая основная и дополнительная литература.

В разделе «Материально-техническое обеспечение дисциплины, программное обеспечение, профессиональные базы данных и информационные справочные системы» содержится перечень профессиональных баз данных и информационных справочных систем, необходимых для освоения дисциплины.

5.2 Рекомендации обучающимся по работе с теоретическими материалами по дисциплине

При изучении и проработке теоретического материала необходимо:

- просмотреть еще раз презентацию лекции в системе MOODLe, повторить законспектированный на лекционном занятии материал и дополнить его с учетом рекомендованной дополнительной литературы;
- при самостоятельном изучении теоретической темы сделать конспект, используя рекомендованные в РПД источники, профессиональные базы данных и информационные справочные системы;
- ответить на вопросы для самостоятельной работы, по теме представленные в пункте 3.2 РПД.
- при подготовке к текущему контролю использовать материалы фонда оценочных средств (ФОС).

5.3 Рекомендации по работе с научной и учебной литературой

Работа с основной и дополнительной литературой является главной формой самостоятельной работы и необходима при подготовке к устному опросу на семинарских занятиях, к дебатам, тестированию, экзамену. Она включает проработку лекционного материала и рекомендованных источников и литературы по тематике лекций.

Конспект лекции должен содержать реферативную запись основных вопросов лекции, в том числе с опорой на размещенные в системе MOODLe презентации, основных источников и литературы по темам, выводы по каждому вопросу. Конспект может быть выполнен в рамках распечатки выдачи презентаций лекций или в отдельной тетради по предмету. Он должен быть аккуратным, хорошо читаемым, не содержать не относящуюся к теме информацию или рисунки.

Конспекты научной литературы при самостоятельной подготовке к занятиям должны содержать ответы на каждый поставленный в теме вопрос, иметь ссылку на источник информации с обязательным указанием автора, названия и года издания используемой научной литературы. Конспект может быть опорным (содержать лишь основные ключевые позиции), но при этом позволяющим дать полный ответ по вопросу, может быть подробным. Объем конспекта определяется самим студентом.

В процессе работы с основной и дополнительной литературой студент может:

- делать записи по ходу чтения в виде простого или развернутого плана (создавать перечень основных вопросов, рассмотренных в источнике);
- составлять тезисы (цитирование наиболее важных мест статьи или монографии, короткое изложение основных мыслей автора);
- готовить аннотации (краткое обобщение основных вопросов работы);
- создавать конспекты (развернутые тезисы).

5.4. Рекомендации по подготовке к отдельным заданиям текущего контроля

Собеседование предполагает организацию беседы преподавателя со студентами по вопросам практического занятия с целью более обстоятельного выявления их знаний по определенному разделу, теме, проблеме и т.п. Все члены группы могут участвовать в обсуждении, добавлять информацию, дискутировать, задавать вопросы и т.д.

Устный опрос может применяться в различных формах: фронтальный, индивидуальный, комбинированный. Основные качества устного ответа подлежащего оценке:

- правильность ответа по содержанию;
- полнота и глубина ответа;
- сознательность ответа;
- логика изложения материала;
- рациональность использованных приемов и способов решения поставленной учебной задачи;
- своевременность и эффективность использования наглядных пособий и технических средств при ответе;
- использование дополнительного материала;
- рациональность использования времени, отведенного на задание.

Устный опрос может сопровождаться презентацией, которая подготавливается по одному из вопросов практического занятия. При выступлении с презентацией необходимо обращать внимание на такие моменты как:

- содержание презентации: актуальность темы, полнота ее раскрытия, смысловое содержание, соответствие заявленной темы содержанию, соответствие методическим требованиям (цели, ссылки на ресурсы, соответствие содержания и литературы), практическая направленность, соответствие содержания заявленной форме, адекватность использования технических средств учебным задачам, последовательность и логичность презентуемого материала;
- оформление презентации: объем (оптимальное количество), дизайн (читаемость, наличие и соответствие графики и анимации, звуковое оформление, структурирование информации, соответствие заявленным требованиям), оригинальность оформления, эстетика, использование возможности программной среды, соответствие стандартам оформления;
- личностные качества: ораторские способности, соблюдение регламента, эмоциональность, умение ответить на вопросы, систематизированные, глубокие и полные знания по всем разделам программы;
- содержание выступления: логичность изложения материала, раскрытие темы, доступность изложения, эффективность применения средств ИКТ, способы и условия достижения результативности и эффективности для выполнения задач своей профессиональной или учебной деятельности, доказательность принимаемых решений, умение аргументировать свои заключения, выводы.

6. Учебно-методическое и информационное обеспечение дисциплины

6.1 Основная литература:

1. Лубашева, Т. В., Железко, Б. А. Основы алгоритмизации и программирования : учебное пособие. - 2022-08-04; Основы алгоритмизации и программирования. - Минск: Республиканский институт профессионального образования (РИПО), 2016. - 379 с. - Текст : электронный // IPR BOOKS [сайт]. - URL: <http://www.iprbookshop.ru/67689.html>
2. Агафонов Е. Д., Ващенко Г. В. Прикладное программирование : учебное пособие. - Красноярск: Сибирский федеральный университет, 2015. - 112 с. - Текст : электронный // ЭБС «Университетская библиотека онлайн» [сайт]. - URL: <http://biblioclub.ru/index.php?page=book&id=435640>
3. Белоцерковская И. Е., Галина Н. В., Катаева Л. Ю. Алгоритмизация. Введение в язык программирования C++. - 2-е изд., испр.. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 197 с. - Текст : электронный // ЭБС «Университетская библиотека онлайн» [сайт]. - URL: <http://biblioclub.ru/index.php?page=book&id=428935>
4. Колокольникова А. И., Таганов Л. С. Информатика: 630 тестов и теория : пособие. - Москва: Директ-Медиа, 2014. - 429 с. - Текст : электронный // ЭБС «Университетская библиотека онлайн» [сайт]. - URL: <http://biblioclub.ru/index.php?page=book&id=236489>

6.2 Дополнительная литература:

1. Седжвик Р. Алгоритмы на C++. - 2-е изд., испр.. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с. - Текст : электронный // ЭБС «Университетская библиотека онлайн» [сайт]. - URL: <http://biblioclub.ru/index.php?page=book&id=429164>
2. Сеницын, С. В., Хлытчиев, О. И. Основы разработки программного обеспечения на примере языка C. - 2021-01-23; Основы разработки программного обеспечения на примере языка C. - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 211 с. - Текст : электронный // IPR BOOKS [сайт]. - URL: <http://www.iprbookshop.ru/73700.html>
3. Алексеев Е. Р., Злобин Г. Г., Костюк Д. А., Чеснокова О. В., Чмыхало А. С. Программирование на языке C++ в среде Qt CreaTo. - 2-е изд., испр.. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 716 с. - Текст : электронный // ЭБС «Университетская библиотека онлайн» [сайт]. - URL: <http://biblioclub.ru/index.php?page=book&id=428929>

6.3 Иные источники:

1. Портал "Гуманитарное образование" - <http://www.humanities.edu.ru/>
2. Федеральный портал "Российское образование" - <http://www.edu.ru/>
3. Федеральное хранилище «Единая коллекция цифровых образовательных ресурсов» - <http://school-collection.edu.ru/>
4. Электронная библиотека социологического факультета МГУ имени М.В. Ломоносова - <http://lib.socio.msu.ru/l/library>
5. Электронная версия «Социологического журнала», издаваемого Российской академией наук Институтом социологии РАН - www.nir.ru/socio/scipubl/socjour.htm
6. Журнал «Социологические исследования» - <http://socis.isras.ru/>

7. Материально-техническое обеспечение дисциплины, программное обеспечение, профессиональные базы данных и информационные справочные системы

Для проведения занятий по дисциплине необходимо следующее материально-техническое обеспечение: учебные аудитории для проведения занятий лекционного и семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, помещения для самостоятельной работы.

Учебные аудитории и помещения для самостоятельной работы укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы укомплектованы компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду Университета.

Для проведения занятий лекционного типа используются наборы демонстрационного оборудования, обеспечивающие тематические иллюстрации (проектор, ноутбук, экран/ интерактивная доска).

Лицензионное и свободно распространяемое программное обеспечение:

Операционная система "Альт Образование"

Microsoft Windows 10

Профессиональные базы данных и информационные справочные системы:

1. Электронный каталог Фундаментальной библиотеки ТГУ. – URL: <https://www.tsutmb.ru/biblio/elektronnyij-katalog/>
2. Университетская библиотека онлайн: электронно-библиотечная система. – URL: <https://biblioclub.ru>
3. Консультант студента. Гуманитарные науки: электронно-библиотечная система. – URL: <https://www.studentlibrary.ru>

4. Научная электронная библиотека eLIBRARY.ru. – URL: <https://elibrary.ru>
5. Российская государственная библиотека: официальный сайт. – URL: <https://www.rsl.ru>
6. Российская национальная библиотека: официальный сайт. – URL: <http://nlr.ru>
7. Президентская библиотека имени Б.Н. Ельцина. – URL: <https://www.prlib.ru>
8. Научная электронная библиотека Российской академии естествознания. – URL: <https://www.monographies.ru>
9. Электронная библиотека РФФИ. – URL: <https://www.rfbr.ru/rffi/ru/library>

Электронная информационно-образовательная среда

https://auth.tsutmb.ru/authorize?response_type=code&client_id=moodle&state=xyz

Взаимодействие преподавателя и студента в процессе обучения осуществляется посредством мультимедийных, гипертекстовых, сетевых, телекоммуникационных технологий, используемых в электронной информационно-образовательной среде университета.