ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ

дисциплины «Основы программирования»

Челябинск, 2023

Оглавление

1 . Введение в Python 3
Задание 1. Среда программирования PyCharm 3
Задание 2. Переменные. Типы 8
Задание 3. Строки 12
Задание 4. Структуры данных 18
2 . Решение задач 23
Задание 5. Условный оператор. Подпрограммы 23
Задание 6. Структуры данных 30
Задание 7. Структуры данных ..... **Ошибка! Закладка не определена.**
Задание 8. Структуры данных ..... **Ошибка! Закладка не определена.**
3 . Решение задач 40
Задание 9. Структуры данных 40
Задание 10. Структуры данных 41

1. Введение в Python

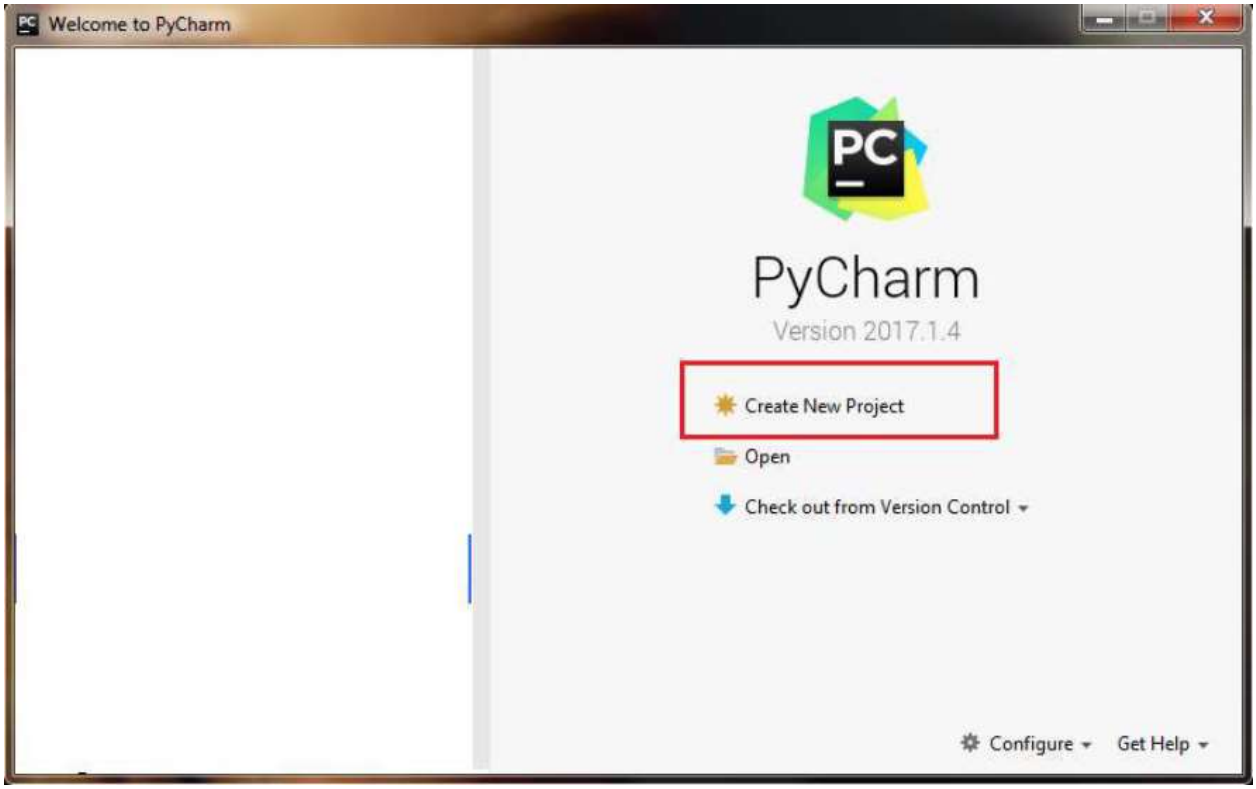
Задание 1. Среда программирования PyCharm

Выполните следующие задания:

1) Скачайте и установите интерпретатор языка python:
https://www.python.org/ftp/python/3.6.2/python-3.6.2.exe

2) Скачайте и установите среду разработки PyCharm («Community»):
https://www.jetbrains.com/pycharm/download/
Не забудьте выбрать тип вашей операционной системы.

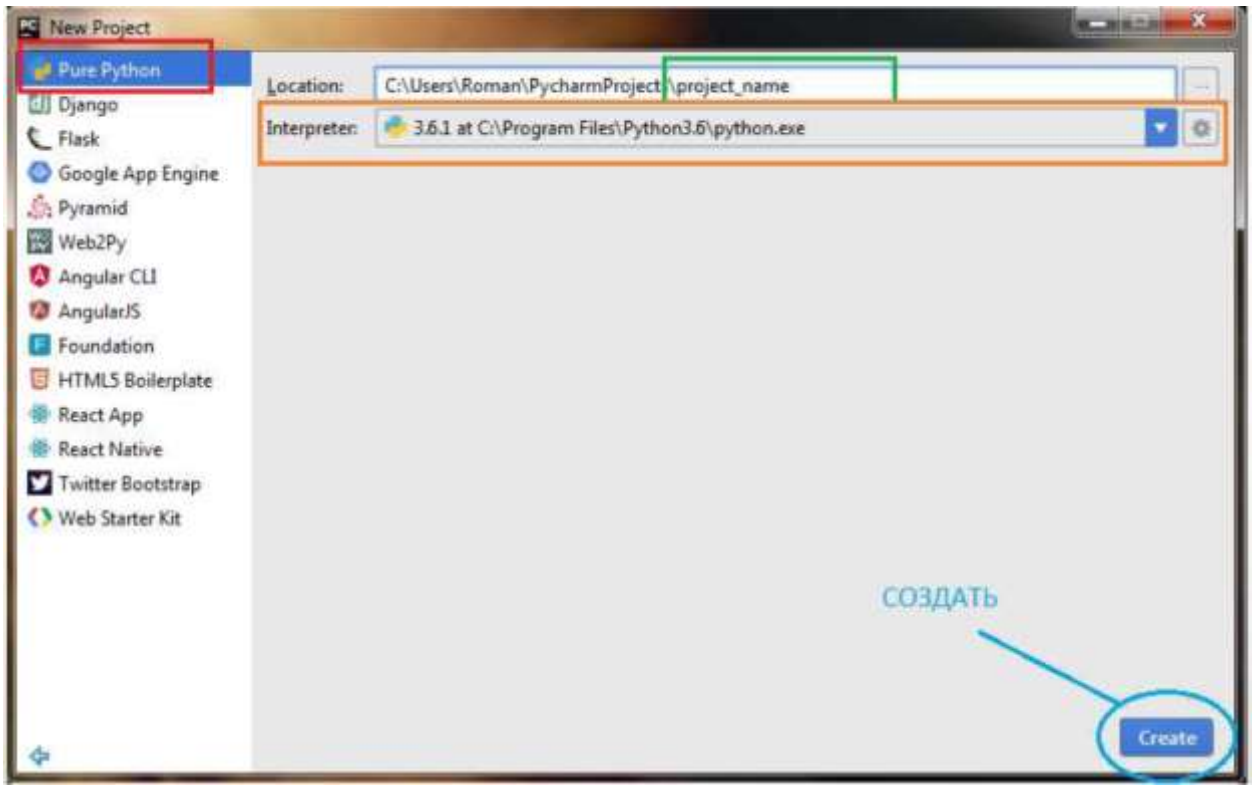
3) Запустите среду PyCharm и нажмите кнопку «Create New Project»:



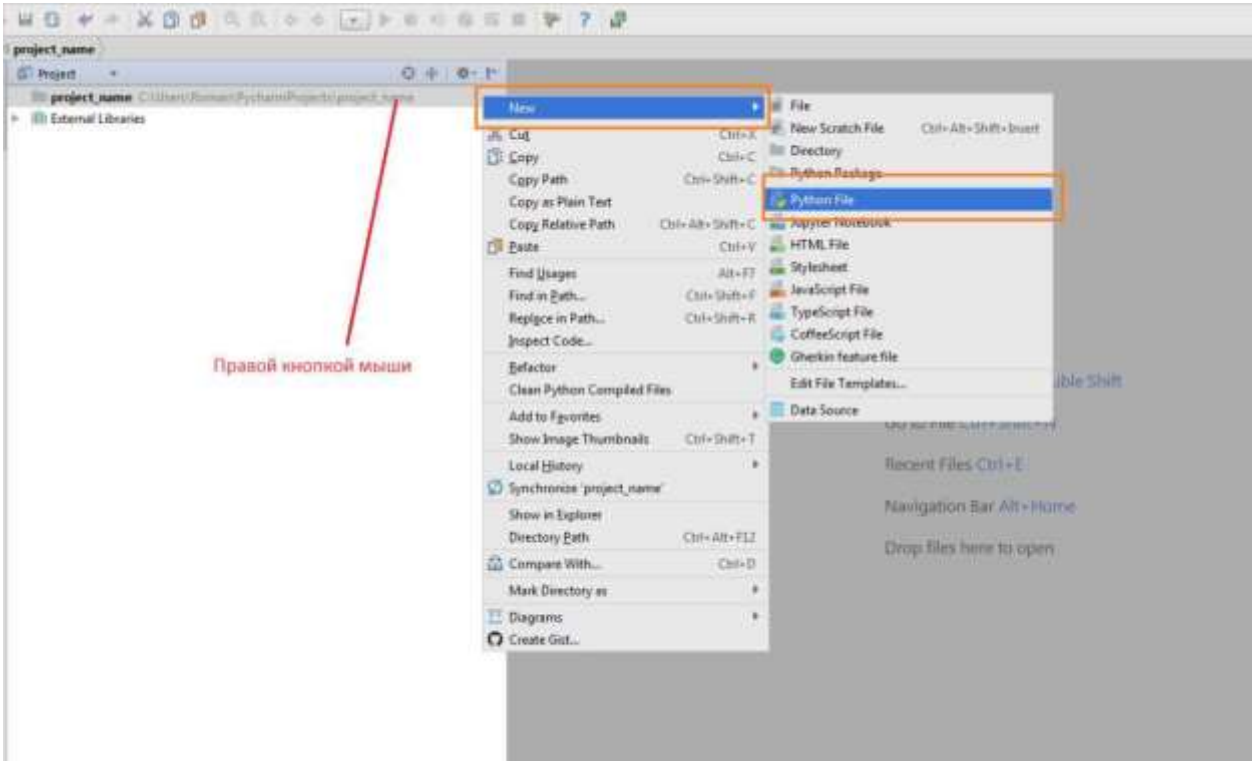
4) При создании проекта обратите внимание на три основных момента:

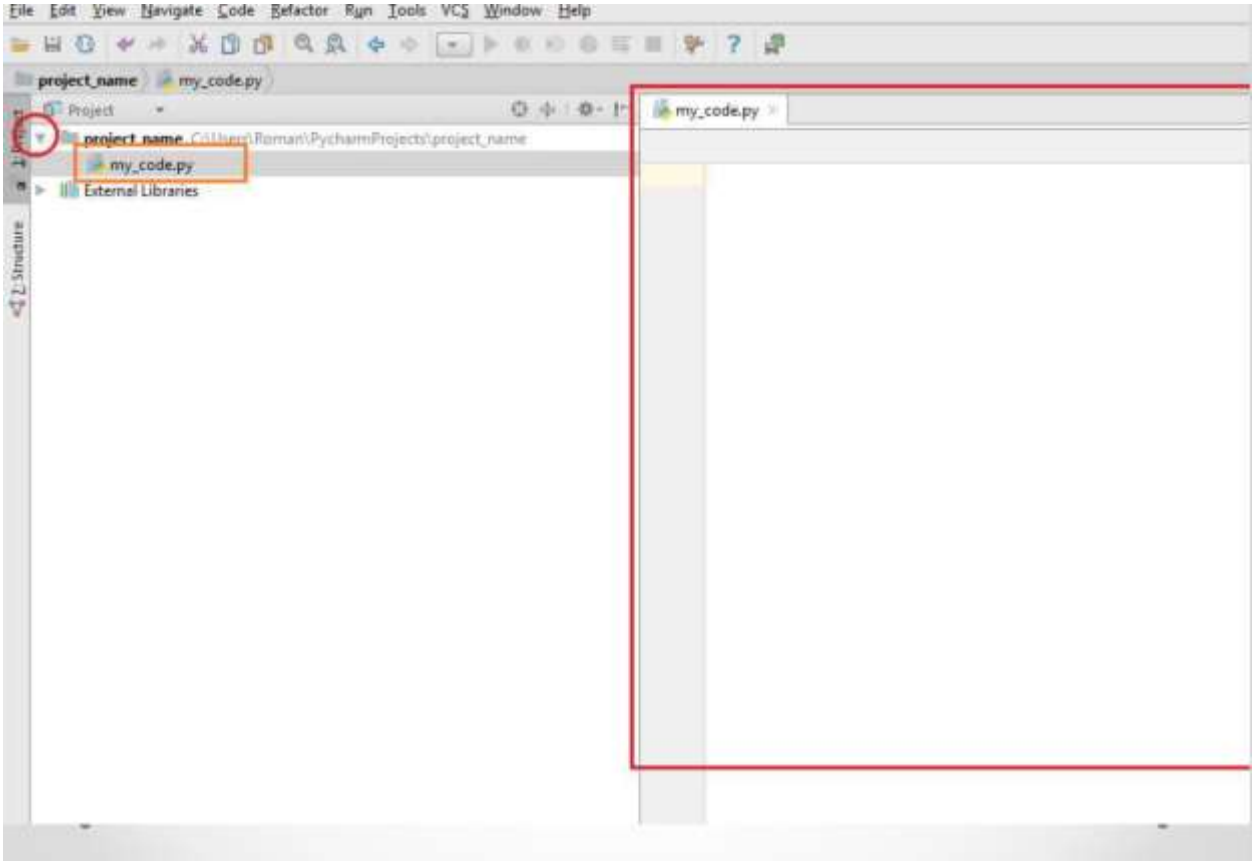
1. В меню слева должен быть выбран параметр «Pure Python» ;
2. Последняя директория в указании вашего пути – это название вашего проекта;
3. В поле «Interpreter» должен быть указан установленный интерпретатор.

Нажмите на кнопку «Create». Будет создан пустой проект.



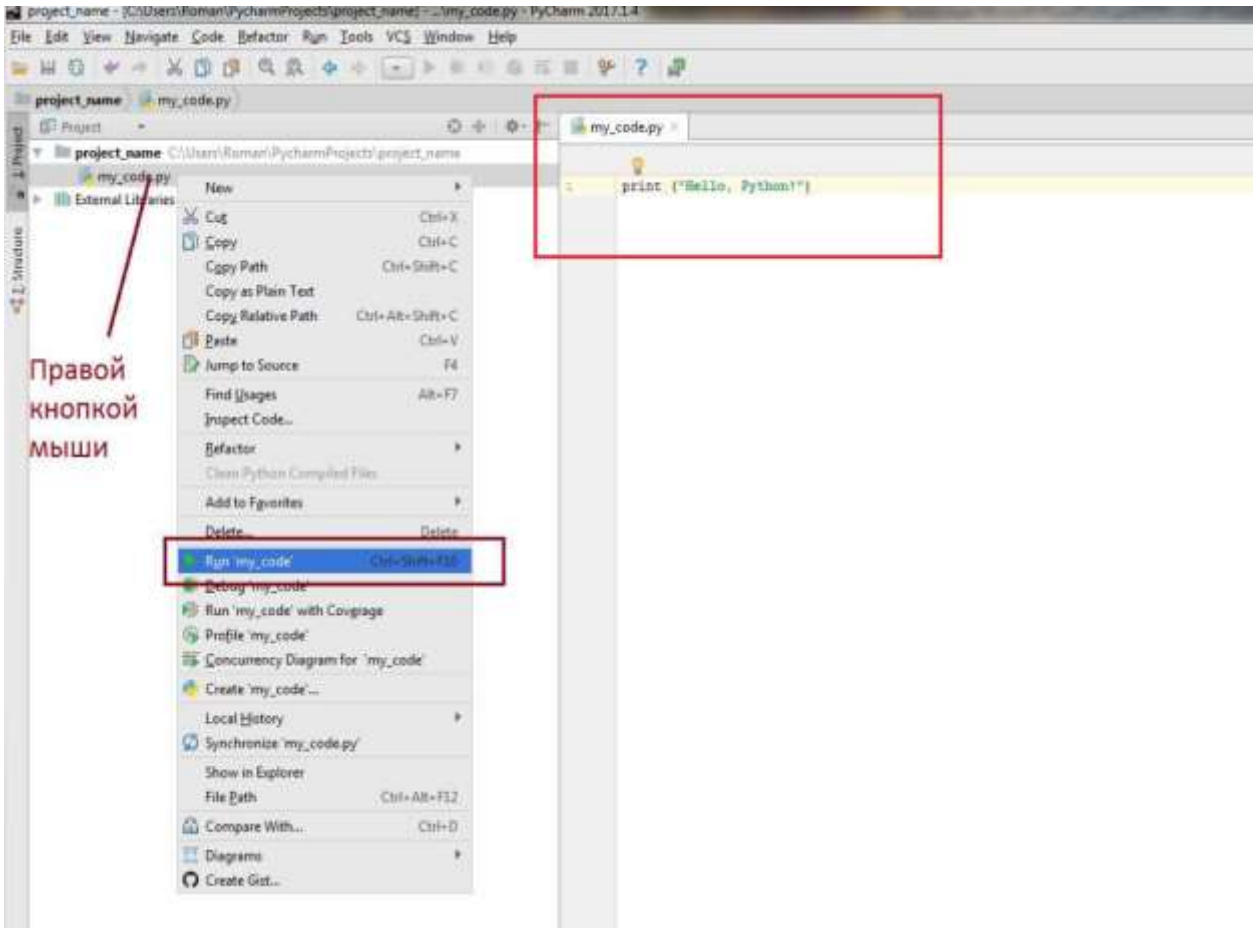
5) Для написания и выполнения кода необходимо создать файл с расширением «.py» в вашем проекте. В директории вашего проекта будет создан файл с указанным вами названием. Для просмотра файлов в директории разверните её. Выберите созданный файл двойным левым щелчком мыши. После этого можно приступать к написанию кода в созданном файле.



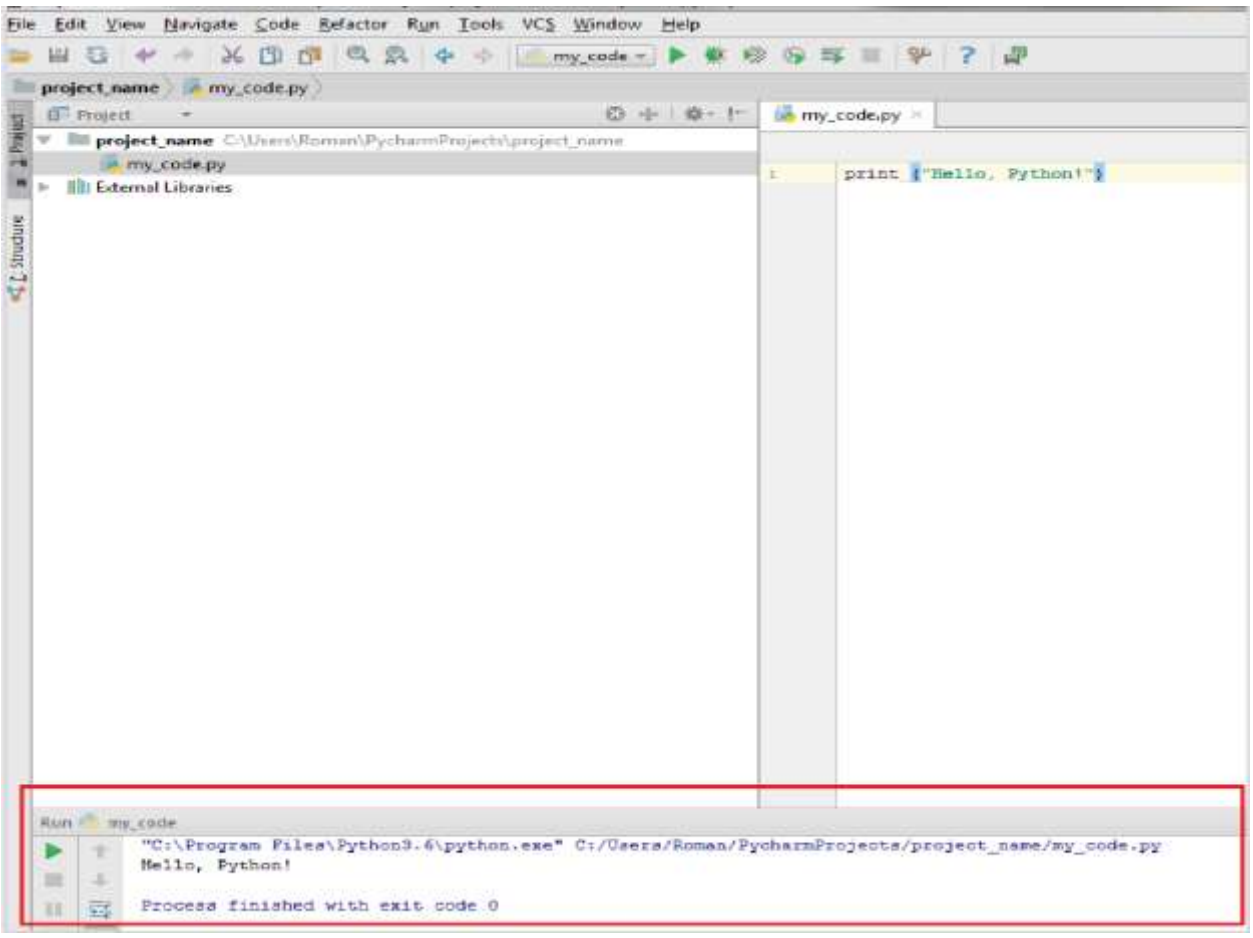


6) Добавьте в ваш файл строчку кода следующего содержания (без отступов от левого края): print (“Hello, Python!”)

После чего запустите выполнение: кликните правой кнопкой по вашему файлу –> выберите пункт всплывающего окна “Run ...”, либо нажатием сочетания Cntrl+Shift+f10.



Результат работы программы и статус её завершения будет отображен в консоли, появляющейся снизу



7) Самостоятельно изучите меню среды программирования и найдите следующие команды: Step Over, Step Into, Run to Cursor, Resume Program, Toogle Line Breakpoint, Save as.

8) Определите, в каком окне среды программирования отображаются значения переменных во время выполнения программы - окно Variables.

9) Создайте файл "lab1.py". Скопируйте текст из листинга ниже.

```
# This is the comment for the comments.py file
primt("Game over!") # this comment is for the second line
print("# this is not a comment")
# add new comment here
input("\n\n Enter any button to exit.")
```

Запустите программу. Объясните результат. Внесите изменения, необходимы для безаварийного запуска программы.

10) Напишите программу, которая будет выводить вашу имя, фамилию и группу (на русском языке). Данные должны быть сохранены в соответствующих переменных.

11) Напишите программу, которая будет выводить на экран понравившийся вам афоризм (https://aphorism.ru). В отдельной строке программа должны выводить автора афоризма (с отступом от левого края в 12 символов).

12) Письменно ответьте на контрольные вопросы:

1. Как создать новый проект?
2. Как создать новый ру-файл?
3. Как без использования команды меню Run и без использования контекстного меню запустить программу?
4. Как без использования команды меню Run и без использования контекстного меню запустить трассировку?
5. Как без использования команды меню Run и без использования контекстного меню установить точку останова?
6. Почему в вашей первой программе, написанной на python, слова написаны разными цветами?
7. Объясните назначение команды print? input?

Задание 2. Переменные. Типы

Выполните следующие задания:

1) Создайте файл "lab2-1.py". Скопируйте текст из листинга ниже.

```
a = b = 2      # Пример множественного оператора присваивания.
# Обратите внимание на значения переменных a и b после выполнения данной операции.

print("a = " + str(a))    #  Функция str() используется для приведения типа аргумента к строковому

print("b = " + "str(b)")


hello = "Привет'

Print("hello = " + str(hello))

# Переопределите значение переменной hello

print("hello = " + str(hello))

# Создайте ещё несколько переменных, а затем осуществите их вывод через пробел

# print(...)
```

Внимательно прочитайте комментарии в тексте программы. Запустите программу. Объясните результат. При необходимости исправьте ошибки.

2) Создайте файл "lab2-2.py". Скопируйте текст из листинга ниже.

```
# Целые числа и числа с плавающей точкой являются одними из самых
распространенных в языке Python

number = 9

print(type(number))    # Вывод типа переменной number

float_number = 9.0

# Создайте ещё несколько переменных разных типов и осуществите вывод их
типов



# Существует множество функций, позволяющих изменять тип переменных.
# Изучите такие функции как int(), float(), str() и последовательно
примените их к переменным, созданным ранее.
```

Внимательно прочитайте комментарии в тексте программы. Запустите программу. Объясните результат. При необходимости исправьте ошибки.

3) В файле "lab2-2.py" преобразуйте тип переменной float_number к типу integer. Результат выведите на экран.

4) Создайте файл "lab2-3.py". Скопируйте текст из листинга ниже.

```
"""
Как и в любом другом языке программирования, в Python поддерживаются
стандартные математические операции:
сложение (+), вычитание (-), умножение (*), деление (/).
Помимо этого в языке также поддерживается возведение в степень (**) и
получение остатка по модулю (%).
"""

number = 9.0

# Посчитайте целую часть от деления number на 2.



# Также посчитайте остаток от предыдущей операции и выведите результат
обеих операций в консоль

print("result = " + str())

print("remainder = " + str())
```

Внимательно прочитайте комментарии в тексте программы. Запустите программу. Исправьте программу так, чтобы она выводила результат деления number на 2, а также остаток от деления number на 2.

5) Создайте файл "lab2-4.py". Скопируйте текст из листинга ниже.

```
# Составной оператор присваивания объединяет в себе простое присваивание и
арифметическую бинарную операцию (+=, -= и т.д).

number = 9.0

print("number = " + str(number))

number -= 2

print("number = " + str(number))

# Попрактикуйтесь в использовании разных типов составных операторов для
переменной number, выведите результаты в консоль


print("number = " + str(number))
```

Внимательно прочитайте комментарии в тексте программы. Запустите программу. Используйте один из дополнительных операторов присваивания для увеличения значения переменной number на 5.

6) Создайте файл "lab2-5.py". Скопируйте текст из листинга ниже.

```
# Переменные булева типа могут принимать только два значения.
# С помощью оператора сравнения == выведите два возможных значения
переменной is_equal.
# Обратите внимание на результат применения функции int() к булевому
значению.

two = 2

three = 3

print(int())

print(str())

"""
В языке Python поддерживается множество операций сравнения( ==, >=, < и
т.д.).
Все подобные операции имеют одинаковый приоритет. Результат сравнения -
булева переменная.
Сравнения могут осуществляться в любом порядке.
"""

one = 1

two = 2

three = 3

print(one < two < three)  # Сравнения (one < two) и (two < three)
происходят в одно и то же время.

# Создайте несколько переменных и осуществите различные сравнения между
ними, выведите результаты в консоль.
```

Внимательно прочитайте комментарии в тексте программы. Отредактируйте текст программы таким образом, чтобы переменные is_equal и is_greater проверяли истинность высказываний "Переменная two равна переменной three" и "Переменная three больше переменной two" соответственно.

7) Реализуйте программу, которая запрашивает с клавиатуры имя пользователя и выводит сообщение: Hello, имя!.

Вычислите номер своего варианта по следующему алгоритму:

1) вычислить сумму даты и месяца рождения;

2) вычислить остаток от деления суммы на 4;

3) прибавить 1 к результату.

8) Решите одну из следующих задач (согласно вашему варианту):

1. Вычислить радиус сферы, описанной вокруг куба с ребром длины a.

2. Вычислить радиус сферы, вписанной в икосаэдр с ребром длины a.

3. Вычислить радиус сферы, описанной вокруг икосаэдра с ребром длины a.

4. Вычислить площадь равнобедренной трапеции с радиусом вписанной окружности r, и углом при основании α.

9) Решите одну из следующих задач (согласно вашему варианту):

1. $y = \frac{1}{\sqrt{1-x^4-x^2}}$

2. $y = e^{\sqrt{x+\sqrt{x}}}$

3. $y = \frac{x^2+1}{3(x^2-1)} + (x^2-1)(1-x)$

4. $y = \sqrt{x+\sqrt{x+\sqrt{x}}}$

10) Письменно ответьте на контрольные вопросы.

1. Дайте определение термину "chained assignment"? Каким образом этот термин звучит на русском языке?

2. Какие символы можно/нельзя использовать при именовании переменных в Python?

3. Что выдаст программа, если вывести на экран переменную, не имеющую значения?

4. Что такое "составной оператор присваивания"?

5. Что такое "множественный оператор присваивания"?

6. Как нужно изменить текст программы lab4-5.py, чтобы переменная is_greater была истинна для высказывания "Переменная three не меньше переменной two"?

7. Перечислите типы Python.

8. Каким образом в Python осуществляется вывод данных на экран?

Задание 3. Строки

Выполните следующие задания:

1) Создайте файл "lab3-1.py". Скопируйте текст из листинга ниже.

```
# Составной оператор присваивания объединяет в себе простое присваивание и
арифметическую бинарную операцию (+=, -= и т.д).

number = 9.0

print("number = " + str(number))

number -= 2

print("number = " + str(number))

# Попрактикуйтесь в использовании разных типов составных операторов для
переменной number, выведите результаты в консоль


print("number = " + str(number))
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

2) Создайте файл "lab3-2.py". Скопируйте текст из листинга ниже.

```
# Ввод данных с клавиатуры осуществляется с помощью функции input().


a = input()      # В переменную a записывается введенное с клавиатура
значение


print(a)



name = input("What is your name?")  # Текст в скобках выводится перед
считыванием данных

print(...)

# Дополните программу так, чтобы она выводила на экран "Hello, <ваше имя>"




"""Функция input() возвращает только строковые значения.
Если вам нужны данные другого типа, введенное значение нужно явно
преобразовать в необходимый тип
"""



a = int(input())   # Преобразование введенной строки к целочисленному типу
данных

b = ???     # Считайте с клавиатуры вещественное число. Используйте функцию
float()
print(...)     # Выведите на экран сумму a и b
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

3) Создайте файл "lab3-3.py". Скопируйте текст из листинга ниже.

```
# Ввод данных с клавиатуры осуществляется с помощью функции input().


a = input()      # В переменную a записывается введенное с клавиатура
значение


print(a)



name = input("What is your name?")  # Текст в скобках выводится перед
считыванием данных

print(...)

# Дополните программу так, чтобы она выводила на экран "Hello, <ваше имя>"




"""Функция input() возвращает только строковые значения.
Если вам нужны данные другого типа, введенное значение нужно явно
преобразовать в необходимый тип
"""



a = int(input())   # Преобразование введенной строки к целочисленному типу
данных

b = ???     # Считайте с клавиатуры вещественное число. Используйте функцию
float()
print(...)     # Выведите на экран сумму a и b
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

4) Создайте файл "lab3-4.py". Скопируйте текст из листинга ниже.

```
""" Срез используется для получения нескольких символов (подстроки) из
строки.
Его синтаксис аналогичен синтаксису индексации, но вместо одного индекса вы
используете два индекса (целых числа), разделенных двоеточием.
Например, str[2:5]
"""

monty_python = "Monty Python"

print(monty_python[1:5])

monty = monty_python[:5]  # Один или оба индекса могут быть отброшены.
monty_python[:5]   # Эквивалентен выражению monty_python[0:5]
print(monty)

python = ???     # Используйте оператор среза для получения подстроки
“Python”
print(python)



# Используйте ключевое слово in для проверки того, содержится ли подстрока
в строке.

ice_cream = "ice cream"

print("cream" in ice_cream)  # Обратите внимание на результат операции

# Проверьте есть ли подстроки "ice", "Ice" и "ice " в строке "ice cream".
Выведите результаты каждой из проверок.
print(???)



# Функция len() используется для подсчета количества символов в строке

# Символы """ и """ используются для создания многострочной строки

phrase = """

It is a really long string

triple-quoted strings are used

to define multi-line strings

"""

# Получите первую половину строки, хранящейся в переменной phrase.
# Примечание: не забудьте о преобразовании типов.
first = ???
print(first)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

При необходимости создать строку, подставив в неё некоторые данные, полученные в процессе выполнения программы (пользовательский ввод,

данные из файлов и т. д.), можно с помощью форматирования строк. Форматирование можно сделать с помощью оператора %.

Если для подстановки требуется только один аргумент, то значение – это сам аргумент:

```
>>> 'Hello, %s!' % 'Vasya'
'Hello, Vasya!'
```

А если требуется несколько аргументов, то значением будет являться кортеж со строками подстановки:

```
>>> '%d %s, %d %s' % (6, 'bananas', 10, 'lemons')
'6 bananas, 10 lemons'
```

Обозначения “%d” и “%s” называются *спецификаторами*. От спецификатора зависит подставляемое значение.

Тип спецификатора	**Подставляемое значение**
'%d', '%i', '%u'	Десятичное число
'%o'	Число в восьмеричной системе счисления
'%x'	Число в шестнадцатеричной системе счисления (буквы в нижнем регистре)
'%X'	Число в шестнадцатеричной системе счисления (буквы в верхнем регистре)
'%e'	Число с плавающей точкой с экспонентой (экспонента в нижнем регистре)
'%E'	Число с плавающей точкой с экспонентой (экспонента в верхнем регистре)
'%f', '%F'	Число с плавающей точкой (обычный формат)
'%g'	Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат
'%G'	Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат
'%c'	Символ (строка из одного символа или число - код символа)
'%r'	Строка (литерал python)
'%s'	Строка (как обычно воспринимается пользователем)
'%%'	Символ '%'

Спецификаторы преобразования записываются в следующем порядке:

1. Сначала символ '%'.
2. Ключ (опционально), определяет, какой аргумент из значения будет подставляться.
3. Флаги преобразования.

4. Минимальная ширина поля. Если *, значение берётся из кортежа.
5. Точность, начинается с '.', затем - желаемая точность.
6. Модификатор длины (опционально).
7. Тип (см. таблицу выше).

```
>>> print ('%(language)s has %(number)03d quote types.' % {"language":
"Python", "number": 2})
Python has 002 quote types.
```

Доступны следующие флаги преобразования:

Флаг	Значение
"#"	Значение будет использовать альтернативную форму
"0"	Свободное место будет заполнено нулями
"-"	Свободное место будет заполнено пробелами справа
" "	Свободное место будет заполнено пробелами справа
"+"	Свободное место будет заполнено пробелами слева

5) Создайте файл "lab3-5.py". Напишите программу, которая запрашивает и считывает с клавиатуры ваши данные: фамилия, имя, группа. Далее программа должна выдать текст:

Привет, <фамилия> <имя> из группы <группа>!
Введи свою электронную почту?

6) Далее программа ожидает ввода почты и возвращает на экран зашифрованный следующим образом текст: 5 символов фамилии + 5 символов имени (дважды) + 5 символов электронной почты (трижды), все символы в нижнем регистре.

7) Реализуйте программу, которая считывает целое число 1<N<10 с клавиатуры и печатает на экран N изображений в одну строку в соответствии с вариантом:

1	2	3	4	5
(__/) (='.'=) (")_(")	_~_ (o o) / V \ /(_)\ ^^ ^^	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ –,,,– ~ ~ ~ ~ ~ <°)3333><(~ ~ ~ ~ "?""? ~ ~ ~ ~	~ ><((((o> ~	``@..@ `(_____) (>____<) ^^~~~~^^

При реализации программы можно использовать операторы input, print. Использование других операторов (в том числе цикла) запрещено.

8) Письменно ответьте на контрольные вопросы.

1. Каким образом в Python осуществляется чтение данных с клавиатуры?
2. Дайте определение термину "concatenation" ?
3. Как с помощью оператора умножения (*) получить строку вида: "strstrstr" ?
4. Как с помощью операции среза получить подстроку "like" из строки "I like to programming!" ?
5. Каким будет результат, если к строковой переменной применить операцию среза: [1:] ?
6. Каким образом можно определить длину любой строки?
7. Что означает ключевое слово "in" ?
8. Для чего в языке Python используются символы %, \, ", ', \n ?

Задание 4. Структуры данных

Выполните следующие задания:

1) Создайте файл "lab4-1.py". Скопируйте текст из листинга ниже.

```
"""
СПИСОК - это структура данных, которую вы можете использовать для хранения
коллекции различных данных в одной переменной.

Список может быть записан в виде последовательности значений (элементов),
разделенных запятыми, в квадратных скобках, например, lst = [item1, item2].
Списки могут содержать элементы разных типов, но обычно все элементы в
списке одного типа.
Списки, так же как и строки, поддерживают операции среза и обращения по
индексу.
"""

squares = [1, 4, 9, 16, 25]    # Создание нового списка

print(squares)

# Выведите на экран подсписок [4, 9, 16].

print(...)


"""
Вы можете добавлять новые элементы в конец списка, используя метод append()
и конкатенацию.

В отличие от строк, списки являются изменяемым типом, то есть их содержимое
можно изменить с помощью lst[index] = new_item
"""

animals = ['elephant', 'lion', 'tiger', "giraffe"]  # Создание нового
списка

print(animals)
```

```
animals += ["monkey", 'dog']     # Добавление новых элементов в список с
помощью операции конкатенации

print(animals)

animals.append("dino")       # Добавление нового элемента с помощью метода
append()

print(animals)


# Замените элемент 'dino' на 'dinosaur'

print(...)


"""
Также возможно присвоение срезов, что может изменить размер списка или
полностью его очистить.
"""

animals = ['elephant', 'lion', 'tiger', "giraffe", "monkey", 'dog']   #
Создание нового списка
print(animals)

animals[1:3] = ['cat']       # Замена сразу двух элементов - 'lion' и
'tiger' - на один элемент - 'cat'
print(animals)

animals[1:3] = []         # Удаление элементов 'cat' и 'giraffe' из списка
print(animals)

# Теперь полностью очистите список

?????
print(animals)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

2) Создайте файл "lab4-2.py". Скопируйте текст из листинга ниже.

```
"""
КОРТЕЖИ практически идентичны спискам.

Единственное существенное различие между кортежами и списками заключается в
том, что они не могут быть изменены: вы не можете добавлять,
изменять или удалять элементы из кортежа.

Кортежи создаются с помощью запятых, заключенных в скобки, например (a, b,
c).
Заметьте, что для создания кортежа из одного элемента - в скобках должна
быть запятая, например (d,).
"""

alphabet = ('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
            'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
            'y', 'z')
print()     # Выведите число букв в алфавите (количество элементов кортежа)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

3) Создайте файл "lab4-3.py". Скопируйте текст из листинга ниже.

```
"""
СЛОВАРЬ похож на список, за исключением того, что доступ к его значениям
осуществляется с помощью ключа вместо индекса. Ключ может быть любой
строкой или числом.
Словари заключены в фигурные скобки, например dct = {'key1': "value1",
"key2": "value2"}.
Аналог словаря в других языках программирования - ассоциативный массив
"""

# Создание нового словаря

phone_book = {"John": 123, "Jane": 234, "Jerard": 345}    # "John", "Jane"
и "Jerard" - ключи, соответствующие числа - значения

print(phone_book)


# Добавление нового элемента в словарь

phone_book["Jill"] = 345
print(phone_book)

# Удаление пары “ключ-значение” из словаря

del phone_book['John']

# Выведите номер Jane из телефонной книги

print(???)


"""
В словарях есть много полезных методов, например keys() и values().
Вы можете изучить остальные, используя Ctrl + Пробел после имени dict_name
с точкой.
"""

phone_book = {"John": 123, "Jane": 234, "Jerard": 345}  # Создание нового
словаря
print(phone_book)


# Добавление элемента с существующим ключом

phone_book["Jill"] = 456
print(phone_book)  # Объясните результат


print(phone_book.keys())


# Выведите все номера из телефонной книги

print(...)
```

```
"""
Ключевое слово in используется для проверки наличия в списке или словаре
определенного элемента.
Вы можете применять к спискам или словарям так же, как со строками.
"""

grocery_list = ["fish", "tomato", 'apples']    # Создание нового списка



print("tomato" in grocery_list)          # Проверка наличия элемента
"tomato" в списке



grocery_dict = {"fish": 1, "tomato": 6, 'apples': 3}   # Создание нового
словаря

# Проверьте, есть ли в словаре запись с ключом "fish" и "potato"

print(...)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

4) Создайте файл "lab4-4.py". Скопируйте текст из листинга ниже.

```
"""
МНОЖЕСТВО в Python -  это структура данных, содержащая не повторяющиеся
элементы в случайном порядке.
"""

a = set()   # Создание пустого множества


a.add(1)    # Добавление элемента в множество

a.add(2)

a.add(1)

print(a)    # Объясните результат


b = set('hello')  # Преобразование строки в множество

print(b)    # Объясните результат

"""
Множества удобно использовать для удаления повторяющихся элементов из
списка
"""

a = ['aa','ab','aa','ba']

print(set(a))
```

```
"""
Множества удобно использовать для удаления повторяющихся элементов из
списка
"""

a = [‘aa’,’ab’,’aa’,’ba’]

print(set(a))


"""
Множества поддерживают стандартные операции других структур данных - len,
in, clear и т.п.
"""

# Вставьте пропущенную строку, чтобы оператор print выводил True

b = [‘aa’,’ab’,’aa’,’ba’]

print(len(b) == 3)


"""
Помимо базовых операций, множества в Python поддерживают операции
математических множеств (объединение(union), пересечение(intersection))
"""

a = set{1,2,3,4}      # Создание множества

b = set{3,4,5,6}


c = a.union(b)      # Объединение множества

print(c)


# Вставьте пропущенное действие, чтобы в консоль вывелось пересечение
множеств a и b

d = a.???

print(d)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

5) Напишите программу ("lab4-5.py"), которая считывала бы с клавиатуры данные о пяти призывниках (Фамилия, Имя, Отчество, год рождения, заболевание) и выводила бы результат в виде таблицы на экран (считывание и сохранение данных должно производиться в словарь). Структуру данных для хранения информации продумайте самостоятельно.

6) Письменно ответьте на контрольные вопросы.

1. В чем разница между списком, словарем и кортежем?

2. Как добавить элемент в конце списка? в середину?

3. Как удалить элемент с конца списка? из середины?

4. Каково назначение операции in ? len ? clear ? Какие еще операции можно применять к структурным данным языка Python?

5. Какие математических операции над множествами поддерживает язык Python?

2. Решение задач

Задание 5. Условный оператор. Подпрограммы

Выполните следующие задания:

1) Создайте файл "lab5-1.py". Скопируйте текст из листинга ниже.

```
""" Логические операции возвращают результаты в булевых (логических)
значениях.

Логическая операция and возвращает True, когда и первый и второй операнды
истинны.

Логическая операция or возвращает True, если хотя бы один из операндов
истинен.

Логическая операция not инвертирует логическое выражение, которому она
предшествует. """


name = "John"
age = 17
print(name == "John" or age == 17)        # True если человека зовут John
или ему 17

print(???)          # Напишите логическое выражение со значением "человека
зовут John или ему 17" с использованием оператора not




""" Булевы операции не всегда выполняются слева направо!

Порядок выполнения операций для логических операций следующий:
not оценивается первым,
and оценивается следующим,
or оценивается последним.
"""

print(name == "John" or not age > 17)     # True если человека зовут John
или ему не более 17 лет

# Напишите логическое выражение, проверяющее следующее высказывание:
"Человека зовут не John, он старше 20 лет и первая буква его имени не 'E'".

print(???)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

2) Создайте файл "lab5-2.py". Скопируйте текст из листинга ниже.

```
""" Ключевое слово if используется для построения условного оператора,
который выполняет определенный код после проверки некоторого условия
(логического выражения) на истинность.

Для определения вложенных блоков кода в Python используются отступы!
"""

name = "John"
age = 17

if name == "John" or age == 17:  # проверяем, зовут ли человека "John " или
ему 17 лет
# Если выполняется хотя бы одно условие, вывести следующие 2 строчки
    print("name is John")      # обратите внимание на отступы перед словом
print
    print("John is 17 years old")


tasks = ['task1', 'task2']     # создаем список

...       # Выведите сообщение "List is empty", если лист tasks пустой
    print("empty")



""" Оператор else дополняет оператор if.

Ключевое слово elif сокращенная запись "else if".  """


x = 28
if x < 0:
    print('x < 0') # выполняется только если x < 0
elif x == 0:
    print('x is zero')  # если x < 0 не истина, то проверяется равенство x
== 0
elif x == 1:
    print('x == 1')   # если x < 0 не истина и x != 0 не истина, проверяется
равенство x == 1
else:
    print('non of the above is true')   # объясните в каком случае
выполняется этот оператор?



name = "John"

# Выведите True если значение переменной name "John" и False в противном
случае

...
    print(True)
...
    print(False)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

3) Создайте файл "lab5-3,py". Скопируйте текст из листинга ниже.

```
""" Функции - это удобный способ разделить ваш код на полезные блоки,
сделать его более читабельным и помочь в его повторном использовании.

Функции определяются с помощью ключевого слова def, за которым следует имя
функции. """

def hello_world():      # Функция с именем "hello_world"
    print("Hello, World!")   # Обратите внимание на отступ!

# Определите функцию с именем fun,
#печатающую при своем вызове любой афоризм


# Это основная программа
#(Напишите эту заметку при написании основной программы)

for i in range(3):
    hello_world()     # вызов функции, определенной выше


# замените дубликаты строк в файле
#(реализуйте подпрограмму, которая печатает сообщение на экран)

print('I want to be a function')
print('I want to be a function')
print('I want to be a function')


#-------------------------------------------------------------------#


""" Параметры функции определены в скобках (), следующих после имени
функции.

Параметры ведут себя как переменные, видимые исключительно внутри тела
функции """

def foo(x):                  # x - параметр функции
    print("x = " + str(x))


foo(5)    # Здесь мы вызываем функцию foo, передавая 5 в качестве
аргумента.


# Определите функцию с именем square, печатающую квадрат передаваемого ей
значения.

square(4)
square(8)
square(15)
square(23)
square(42)


#-------------------------------------------------------------------#
```

```
""" Функции могут возвращать значения, использую ключевое слово return.

Вы можете использовать возвращаемое значение, чтобы присвоить его
переменной или просто вывести его на экран. """


def sum_two_numbers(a, b):
    return a + b            # возвращаем результат


c = sum_two_numbers(3, 12)  # присвоить результат переменной 'c'




# В последовательности Фибоначчи первые два числа равны 1 и 1, и каждое
следующее число равно сумме двух предыдущих.
# Напишите функцию, возвращающую список из чисел Фибоначчи до n.

def fib(n):
"""Эта строка документации для данной функции. Она будет доступна по
fib.__doc__()
    Возвращает список, содержащий числа Фибоначчи до n
"""
    result = []
    a = 1
    b = 1
    while a < n:
        result.append(a)
        tmp_var = b
        # обновите значение переменной b, поместив в нее сумму значений
        # обновите значение переменной a, поместив в нее значение
вспомогательной переменной
    return result

print(fib(10))
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

4) Решите задачу «Вид треугольника».

Входные данные: вводятся с клавиатуры
Выходные данные: выводятся на экран

Напишите программу, которая по заданным длинам сторон определяет вид треугольника:

- треугольник не существует, или
- треугольник равносторонний, или
- треугольник равнобедренный, или
- треугольник общего вида.

Входные данные

Три вещественных числа – длины сторон треугольника.

Выходные данные

В зависимости от вида треугольника выдать одно из следующих сообщений:

- Треугольник не существует
- Треугольник равносторонний
- Треугольник равнобедренный
- Треугольник общего вида

Примеры входных и выходных данных

Входные данные	**Выходные данные**
-1 2 4	Треугольник не существует
0 0 0	Треугольник не существует
2 1 1	Треугольник не существует
1 1 1	Треугольник равносторонний
2 2 1	Треугольник равнобедренный
3 4 5	Треугольник общего вида
6 7 8	Треугольник общего вида

5) Решите задачу «Квадратное уравнение».

Входные данные: вводятся с клавиатуры

Выходные данные: выводятся на экран

Напишите программу, которая по заданным коэффициентам квадратного уравнения находит его вещественные корни и их количество.

Входные данные

Три вещественных числа A, B, C – коэффициенты квадратного уравнения $Ax^2+Bx+C=0$, коэффициент A не может быть равен нулю.

Выходные данные

Результаты вычислений выдать в следующем формате (все вещественные числа выводить с 5-ю знаками после запятой):

- если уравнение не имеет вещественных корней

```
Уравнение
(<A>)*X^2+(<B>)*X+(<C>)=0
Количество корней: < количество вещественных корней>
```

- если уравнение имеет вещественные корни

```
Уравнение
(<A>)*X^2+(<B>)*X+(<C>)=0
Количество корней: <количество вещественных корней>
<1-й корень уравнения (меньший из двух корней)>
<2-й корень уравнения>
```

Примеры входных и выходных данных

Входные данные	**Выходные данные**
1 -7 12	Уравнение (1.00000)*X^2+(-7.00000)*X+(12.00000)=0 Количество корней: 2 3.00000 4.00000
1 4 4	Уравнение: (1.00000)*X^2+(4.00000)*X+(4.00000)=0 Количество корней: 1 -2.00000 -2.00000
4 0 -16	Уравнение: (4.00000)*X^2+(0.00000)*X+(-16.00000)=0 Количество корней: 2 -2.00000 2.00000
1 4 5	Уравнение: (1.00000)*X^2+(4.00000)*X+(5.00000)=0 Количество корней: 0

6) Решите задачу «Время прибытия».

Входные данные: вводятся с клавиатуры

Выходные данные: выводятся на экран

Известно время отправления поезда: Hotp – часы и Motp – минуты. Известно время в пути: Hp – часы и Mp – минуты.

Напечатать время прибытия поезда на конечную станцию в виде «HH час : MM мин».

Определить также количество полных суток в пути до конечной станции.

Входные данные

Четыре целых числа – часы и минуты отправления поезда, а также часы и минуты продолжительности поездки.

Выходные данные

Время прибытия: «HH час. : MM мин.»

Количество полных суток – целое число.

Примеры входных и выходных данных

Входные данные	**Выходные данные**
12 20 20 40	09 hours : 00 minutes 0 days
1 15 36 37	13 hours : 52 minutes 1 days

```
0         01 hours : 03 minuts
48        2 days
48
15
```

7) Решите задачу «Копейка рубль бережет».

Входные данные: вводятся с клавиатуры
Выходные данные: выводятся на экран

Недоверчивый покупатель подсчитывает общую сумму покупок в копейках. Но, подходя к кассе, ему трудно разобраться, сколько рублей и копеек он должен заплатить. Напишите программу, которая поможет ему.

Входные данные

Одно целое число из диапазона 1..100 000 – сумма покупки в копейках.

Выходные данные

Необходимо выдать запись о том, сколько рублей и копеек должен заплатить покупатель. При этом нужно соблюдать следующие правила:

1. Слова "рубль" и "копейка" нужно согласовать с числительными. Например: "1 рубль", но "5 рублей", "1 копейка", но "23 копейки".

2. Вывод числа рублей и копеек нужно производить в отдельных строках. Если число рублей или копеек равно нулю, то соответствующую строку выводить не нужно.

3. Слова "рубль", "копейка" нужно выводить БОЛЬШИМИ буквами.

Примеры входных и выходных данных

Входные данные	**Выходные данные**
101	1 РУБЛЬ 1 КОПЕЙКА
13	13 КОПЕЕК
99999	999 РУБЛЕЙ 99 КОПЕЕК
100000	1000 РУБЛЕЙ

8) Письменно ответьте на контрольные вопросы.

1. Напишите, какое значение будет иметь переменная Z после выполнения операторов:

```
Z = 0
if X > 0 :
  if Y > 0 :
    Z = 1
  else :
    Z = 2
```

при следующих значениях переменных X и Y:

a) X=1, Y=1

b) X=1, Y=-1

c) X=-1, Y=1

2. Пусть имеется оператор присваивания X=A or B and C, где переменные X, A, B, C имеют тип bool. Напишите условный оператор, который эквивалентен данному?

Например, оператору присваивания X=not A эквивалентен условный оператор

```
if A :
  X = False
else :
  X = True
```

3. Пусть имеется условный оператор

```
if A :
  X = B
else :
  X = C
```

где переменные X, A, B, C имеют тип bool. Напишите оператор присваивания, который эквивалентен данному?

4. Напишите синтаксис условного оператора? Какие есть особенности использования elif? Какие есть особенности использования else?

5. Вычислите выражения:

```
123 // 10;
-123 // 10;
123 % -10;
-123 % -10
```

Объясните результат. Какой результат выдается интерпретатором Python?

Задание 6. Циклы. Модули. Файлы

Выполните следующие задания:

1) Создайте файл "lab6-1.py". Скопируйте текст из листинга ниже.

```
""" Циклы for используются для прохождения по заданной последовательности.
На каждой итерации переменной, определенной в цикле for, будет
присваиваться следующее значение в списке
"""

# Функция range (n) возвращает список [0, 1, 2, 3, ..., n-1]

for i in range(5):      # для каждого числа i в диапазоне 0-4
    print(i)          # эта строка выполняется 5 раз. Первый раз i равно 0,
затем 1, затем 2, затем 3, затем 4


primes = [2, 3, 5, 7]   # создание нового списка

# Выведите на экран каждый элемент из primes, используя цикл for по i

for ?? in ???
    print(primes[i])

# Перебор элементов списка возможен и без использования индекса элемента
for prime in primes
    print(prime)


#--------------------------------------#


""" Строки очень похожи на списки в Python. Вы можете использовать другую
строку, чтобы перебрать ее.
"""

hello_world = "Hello, World!"

for ch in hello_world:    # печать каждого символа строки
    print(ch)



length = 0         # переменная length

# Используйте цикл, чтобы подсчитать, сколько символов содержит
hello_world. Сохраните это число в переменной length

for ... in hello_world:
    length += 1      # увеличить переменную на 1 при каждом проходе цикла


print(len(hello_world) == length)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

2) Создайте файл "lab6-2.py". Скопируйте текст из листинга ниже.

```
""" Цикл while похож на оператор if в том смысле, что он выполняет
некоторый код, если выполняется некоторое условие.
 Основное отличие состоит в том, что он будет продолжать выполнять тело
цикла до тех пор, пока условие имеет значение True.
"""

square = 1

while square <= 10:
    print(square)    # Эта операция выполнится 10 раз
    square += 1      # Эта операция выполнится 10 раз
print("Finished")    # Эта операция выполнится один раз


square = 0
number = 1

# Выведите квадраты чисел от 1 до 9 (1, 4, ... , 81). Используйте числовую
переменную в цикле while.

while … :
    square = number ** 2
    print(square)
    number += 1

#-----------------------------------------------#

""" Бесконечный цикл - это цикл, который никогда не завершается.
Если условие цикла всегда истинно, такой цикл становится бесконечным.
Ключевое слово break используется для выхода из текущего цикла.
"""

count = 0
while True:      # это условие всегда истинно
    print(count)
    count += 1
    if count >= 5:
        break           # выйти из цикла, если count >= 5


# Выйдите из вечного цикла, правильно использовав break

zoo = ["lion", 'tiger', 'elephant']
while True:             # это условие всегда истинно
    animal = zoo.pop()  # извлечь элемент с конца списка
    print(animal)

# выйдите из цикла, когда текущее животное - это 'elephant'
```

```
#-----------------------------------------------#



"""
Ключевое слово continue используется для пропуска остальной части кода
внутри текущего выполняемого цикла и возврата к оператору for или while.
"""

for i in range(5):
    if i == 3:
        continue     # пропустить остаток кода внутри цикла для заданного
значения i
    print(i)


# Выведите только нечетные числа 1, 3, 5, 7, 9.

for x in range(10):
    if ...           # Проверьте x на четность
        continue     # пропустить print(x)
    print(x)
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

3) Создайте в рабочем каталоге файлы "calculator.py" и "my_module.py". Скопируйте в них текст из листинга ниже.

```
"""
Содержимое файла calculator.py
Этот модуль содержит класс Calculator
"""

class Calculator:

    def __init__(self):
        self.current = 0

    def add(self, amount):
        self.current += amount

    def get_current(self):
        return self.current
```

```
"""
Содержимое файла calculator my_module.py
Этот модуль содержит функцию hello_world
"""

# Модули в Python - это просто файлы Python с расширением .py, содержащие
функции и операторы Python.
# Модули могут быть полезны, когда вы хотите использовать свою функцию в
ряде программ без копирования ее определения в каждую программу.
# Модули импортируются в другие файлы с помощью ключевого слова import и
имени файла без расширения.
# При первой загрузке модуля в работающий скрипт Python он инициализируется
однократным выполнением кода в модуле.


def hello_world(name):
  print(f"Hello, world! My name is {name}")
```

Внимательно прочитайте комментарии в тексте.

4) Создайте файл "lab6-3.py". Скопируйте текст из листинга ниже.

```
import calculator

calc = calculator.Calculator()
# создать новый экземпляр класса Calculator, определенный в модуле
calculator

calc.add(2)
print(calc.get_current())


# Импортируйте модуль my_module и используйте функцию hello_world.

...      # Импортируйте my_module здесь

...      # Вызовите функцию hello_world из модуля my_module


#-----------------------------------------------------#

"""
Python поставляется с библиотекой стандартных модулей.
Запомните, что вы можете использовать Ctrl + Space после точки (.), чтобы
изучить доступные методы модуля.
"""

import datetime

current_date = ...
print(current date)    # Выведите текущую дату, используя встроенный модуль
datetime


#-----------------------------------------------------#
```

```
"""
Специальная форма оператора импорта импортирует имена из модуля
непосредственно в таблицу символов импортирующего модуля.
Таким образом, вы можете использовать функции напрямую без префикса
module_name.
"""

from calculator import Calculator

calc = Calculator()     # теперь мы можем использовать класс Calculator без
префикса calculator.
calc.add(2)
print(calc.get_current())

# Импортируйте функцию hello_world из модуля my_module.
# Сравните с предыдущим примером.
...
print(hello_world())
# Функция hello_world должна вызываться без указания модуля
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

5) Создайте файл "lab6-4.py". Скопируйте текст из листинга ниже.

```
"""
Python имеет ряд встроенных функций для чтения и записи информации из файла
на вашем компьютере.
Функция open используется для открытия файла.
Файл можно открыть в режиме чтения (используя «r» в качестве второго
аргумента) или в режиме записи (используя «w» в качестве второго
аргумента).
Функция open возвращает объект файла. Вам нужно сохранить его, чтобы
закрыть файл позже.
"""

# Создайте в папке с проектом файлы input.txt и input1.txt

f = open("input.txt", "r")   # здесь мы открываем файл "input.txt"
# Второй аргумент используется для определения режима работы с файлом (в
данном случае - чтение из файла)
# Примечание: если вы хотите записывать в файл, используйте «w» в качестве
второго аргумента


# Выведите содержимое «input.txt». Выведите первую строку «input1.txt».
Затем закройте файлы.
for line in f.readlines():   # чтение построчно
    …  # выведите каждую строку
f.close()                    # Важно закрыть файл, чтобы освободить
системные ресурсы.

f1 = open("input1.txt", "r")

# выведите только первую строку input1.txt. Не забудьте закрыть файл



#-------------------------------------------------------#
```

```
""" Если вы откроете файл, используя «w» в качестве второго аргумента (на
запись), будет создан новый пустой файл.
Обратите внимание, что если существует другой файл с таким же именем, он
будет удален.
Если вы хотите добавить какой-либо контент в существующий файл, вы должны
использовать модификатор «a» (добавление).
"""

zoo = ['lion', "elephant", 'monkey']

# Добавьте элементы списка zoo в файл "output.txt"

if __name__ == "__main__":
    f = open("output.txt", ...)  # добавьте нужный аргумент

    for i in zoo:
        # добавьте все элементы в файл

    # закройте файл
```

Внимательно прочитайте комментарии в тексте программы. Выполните задания в комментариях.

6) Изучите материалы по модулю os: https://all-python.ru/osnovy/os.html .

7) Решите задачу «Прямоугольники и квадраты».

Входные данные: считываются из файла input.txt
Выходные данные: записываются в файл

Напишите отдельный модуль, экспортирующий следующий функции:

- PrintRectangle(a, b, file) – печатает в файл с именем file прямоугольник из символов * со сторонами a и b,
- PrintSquare(a, file) – печатает в файл с именем file квадрат из символов * со стороной a.

Напишите основную программу, которая обращается к указанным функциям с разными параметрами.

Примеры входных и выходных данных

Входные данные	**Выходные данные**
3 4 "output.txt"	Содержимое файла "output.txt": * * * * * * * * * *
5 "out.txt"	Содержимое файла "out.txt": * * * * * * * * * * * * * * * *

8) Решите задачу «Все о цифрах в числе».

Входные данные: считываются из файла input.txt
Выходные данные: записываются в файл output.txt

Напишите программу, которая получает список файлов в текущей директории, оценивает количество текстовых файлов в директории и определяет присутствует ли в директории файл input.txt. Если файл input.txt имеется в текущей директории, то программа должна считать из файла input.txt число. Выведите в файл output.txt:

- само число,
- количество цифр в числе,
- сумму цифр в числе,
- произведение цифр в числе.

Если файл input.txt не обнаружен в рабочем каталоге – программа должна выдать соответствующее сообщение.

Примеры входных и выходных данных

Входные данные	Выходные данные
12	Число: 12 Количество цифр: 2 Сумма цифр: 3 Произведение цифр: 2
3 4 4	Число: 3 Количество цифр: 13 Сумма цифр: 3 Произведение цифр: 3
10	Число: 10 Количество цифр: 2 Сумма цифр: 1 Произведение цифр: 0
-	Файл с входными данными не обнаружен

9) Решите задачу «Генератор простых чисел».

Входные данные: считываются из файла input.txt
Выходные данные: записываются в файл output.txt

Напишите программу, которая анализирует рабочий каталог на наличие файла с входными данными input.txt, если файл найден считайте из него число N. Выведите в файл output.txt все простые числа от 1 до N. Если файл не найден – выведите соответствующее сообщение.
Все необходимые функции вынесите в отдельный модуль.

Примеры входных и выходных данных

Входные данные	Выходные данные
8	2 3 5 7

40	2 3 5 7 11 13 17 19 23 29 31 37
10	Число: 10 Количество цифр: 2 Сумма цифр: 1 Произведение цифр: 0
	Файл с входными данными не обнаружен

10) Решите задачу «Особые точки».

Входные данные: считываются из файла input.txt
Выходные данные: записываются в файл output.txt

Напишите программу находящую, сколько точек с целочисленными координатами попадает в круг радиуса R с центром в точке (x,y). Расчетные функции вынесите в отдельный модуль.

В начале файла output.txt программа должна записать сегодняшнюю дату и время начала работы программы. В конец файла – время выполнения в секундах.

Примеры входных и выходных данных

Входные данные	Выходные данные
1 0 0	01.09.2019 11:02 5 0,8
2 1 1	02.09.2019 15:39 13 0,98

11) Решите задачу «Матрицы».

Входные данные: считываются из файла input.txt
Выходные данные: записываются в файл output.txt

Программно сгенерируйте случайную прямоугольную матрицу A размера NxM (N и M считываются из файла input.txt). Выведите матрицу A в файл. Разделите элементы каждой строки на наибольший элемент этой строки.

Сгенерируйте случайную прямоугольную матрицу B размера MxK (K – случайное число из диапазона 5..15). Выведите матрицу B в файл. Посчитайте и выведите в файл произведение матриц A и B.

Входные данные: N, M – целочисленные числа

Примеры входных и выходных данных

Входные данные	Выходные данные
4 3 Пусть K = 2	Матрица A: 0 1 1 1 0 1

	1 1 0 1 1 1 Матрица B: 8 4 7 1 4 2 Матрица A*B: 11 3 12 6 15 5 19 7
2 2 Пусть K = 3	Матрица A: 1 4 2 5 Матрица B: 7 4 2 3 1 2 Матрица A*B: 4,25 2 2,5 5,8 2,6 2,8

Указание. Для генерации случайных чисел используйте стандартный модуль random.

Пример.

```
import random
a = random.randint(0, 9)
b = random.randint(0, 9)
print(a)
print(b)
```

Метод	Характеристика
random()	возвращает число в диапазоне от 0 до 1
randint(a, b)	возвращает целое число в диапазоне от a и b
uniform(a, b)	возвращает вещественное число в диапазоне от a и b
shuffle(A)	перемешивает значения в списке A
choice(A)	возвращает случайный элемент из списка A
sample(A, b)	возвращает последовательность длиной b из набора A

12) Письменно ответьте на контрольные вопросы.

1. Напишите синтаксис цикла for? Опишите семантику оператора.
2. Напишите синтаксис цикла while? Опишите семантику оператора.
3. Каким образом модуль подключается к программе?
4. Поясните значение строки программы: if __name__ == "__main__": .
5. Что такое __name__, __main__?
6. Каким образом можно открыть файл на чтение? на запись?

7. Что произойдет, если мы с помощью функции open() попробуем открыть несуществующий файл?

8. Определите значение переменной S после выполнения следующих операторов:

А)

```
s = 0
i = 0
while i < 5 :
    i += 1
s += 1/i
```

Б)

```
s = 0
i = 1
while i > 1 :
    s = s+1/i
    i = i-1
```

В)

```
s = 1
n = 1
for i in range(2, n) :
  s = s+1/i
```

3. Использование библиотек Python

Задание 7. Библиотека Numpy

Выполните следующие задания:

1) Перейдите по ссылке (https://goo.su/vozHVkz), сохраните блокнот в свою папку на гугл-диске. Для сохранения блокнота нажмите команду меню "Файл", далее "Сохранить копию на Диске" (нужна авторизация). Ознакомьтесь с информацией в файле.

2) Перейдите по ссылке (https://goo.su/eGKDvx) и сохраните блокнот в свою папку на гугл-диске.

3) Откройте настройки (шестеренка в правом верхнем углу, рядом с фото). В разделе "Сайт" выберите тему "dark", в разделе "Редактор" установите галочку "Показывать номера строк".

4) Ознакомьтесь с информацией в блокноте, выполните все ячейки с кодом последовательно (для выполнения кода в ячейке необходимо нажать на кнопку "Play").

5) Выполните самостоятельные задания: 1 – 6.

6) Нажмите на кнопку "Поделиться" (в верхнем правом углу окна редактора), в разделе "Общий доступ" установите права "Читатель" для всех у кого есть ссылка. Нажмите кнопки "Копировать ссылку" и "Готово". В качестве ответа на задание прикрепите ссылку на свой блокнот.

7) Письменно ответьте на контрольные вопросы.

1. Каким образом подключается библиотека Numpy?
2. Какой класс является основным в библиотеке Numpy?
3. Каким образом можно определить одномерный/многомерный массив?
4. Каким образом можно определить размерность массива?
5. Каким образом можно переопределить размерность массива?
6. Могут ли элементы Numpy-массива, быть разных типов?
7. Что определяет метод zeros()?
8. Что определяет метод ones()?
9. Какие типы данных доступны в Numpy?
10. Какие арифметические операции можно производить с массивами Numpy?
11. Что такое «индексирование»?
12. Каким образом можно транспонировать оси многомерного массива в Numpy?
13. Каковы особенности применения арифметических операций для массивов NumPy?
14. Какова особенность операции среза, реализованной для массивов NumPy?

Задание 8. Библиотека Pandas

Выполните следующие задания:

1) Перейдите по ссылке и сохраните блокнот в свою папку на гугл-диске.

2) Ознакомьтесь с информацией в блокноте, выполните все ячейки с кодом последовательно.

3) Выполните самостоятельные задания: 1 - 6.

4) В качестве ответа на задание прикрепите ссылку на блокнот с режимом доступа "Читатель".

12) Письменно ответьте на контрольные вопросы.

1. На основе какой библиотеки python реализована библиотека Pandas?
2. Дайте определение объекту библиотеки Index?
3. Перечислите основные операции для работы с объектами Index?
4. В чем особенность операций пересечения и объединения, реализованных для Index?
5. Дайте определение объекту библиотеки Series?
6. Каков тип Series.values?
7. Какие параметры у функции Series?
8. Можно ли в качестве параметра с данными передавать строковые значения? Каков будет тип данных в серии?
9. Перечислите основные операции для работы с объектами Series?
10. Можно ли изменить значения индекса в уже созданной серии? Если можно, то каким образом?
11. Дайте определение объекту библиотеки DataFrame?
12. Каков тип DataFrame.values?
13. Какие параметры у функции DataFrame?
14. Можно ли в функцию DataFrame передать список списков в качестве параметра с данными? Какой будет индекс у нового объекта?
15. Какие еще типы данных можно в качестве параметра с данными передавать в функцию DataFrame?
16. Перечислите основные операции для работы с объектами DataFrame?
17. В чем особенность среза для объекта DataFrame?
18. Как извлечь значения столбца? По порядковому номеру? По имени?
19. Как извлечь значение строки? По порядковому номеру? По имени?
20. Перечислите функции библиотеки Pandas для работы с файлами?