



Государственное бюджетное образовательное учреждение высшего образования  
Московской области

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

---

Колледж космического машиностроения и технологий

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей  
программного обеспечения для компьютерных систем  
специальность 09.02.03 Программирование в компьютерных системах

Выполнил студент:

Денисов М.В.

\_\_\_\_\_ (подпись)

Ларченко М.А.

\_\_\_\_\_ (подпись)

Гусятинер Л.Б.

\_\_\_\_\_ (подпись)

\_\_\_\_\_ (оценка)

<b>Раздел 1. Техника решения задач с использованием структурного программирования.</b>	<b>3</b>
1.1. Установка интерпретатора Python 3 и настройка окружения..	3
1.2. Техника работы в командной строке и среде IDLE.....	8
1.3. Техника работы с линейными и разветвляющимися программами .....	<b>Ошибка! Закладка не определена.</b>
1.4. Техника работы с циклическими программами, цикл while .....	<b>Ошибка! Закладка не определена.</b>
1.5. Техника работы с числами...	<b>Ошибка! Закладка не определена.</b>
1.6. Техника работы со строками.	<b>Ошибка! Закладка не определена.</b>
1.7. Техника работы со списками.	<b>Ошибка! Закладка не определена.</b>
1.8. Техника работы с циклом for и генераторами списков.	<b>Ошибка! Закладка не определена.</b>
1.9. Техника работы с функциями.	<b>Ошибка! Закладка не определена.</b>
1.10. Техника работы со словарями .....	<b>Ошибка! Закладка не определена.</b>
1.11. Техника работы с множествами .....	<b>Ошибка! Закладка не определена.</b>
1.12. Техника работы с кортежами .....	<b>Ошибка! Закладка не определена.</b>
1.13. Техника работы с файлами .	<b>Ошибка! Закладка не определена.</b>

## Раздел 1. Техника решения задач с использованием структурного программирования.

### 1.1. Установка интерпретатора Python 3 и настройка окружения

Для установки интерпретатора Python на компьютер, первое, что нужно сделать – это скачать дистрибутив. Загрузить его можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>

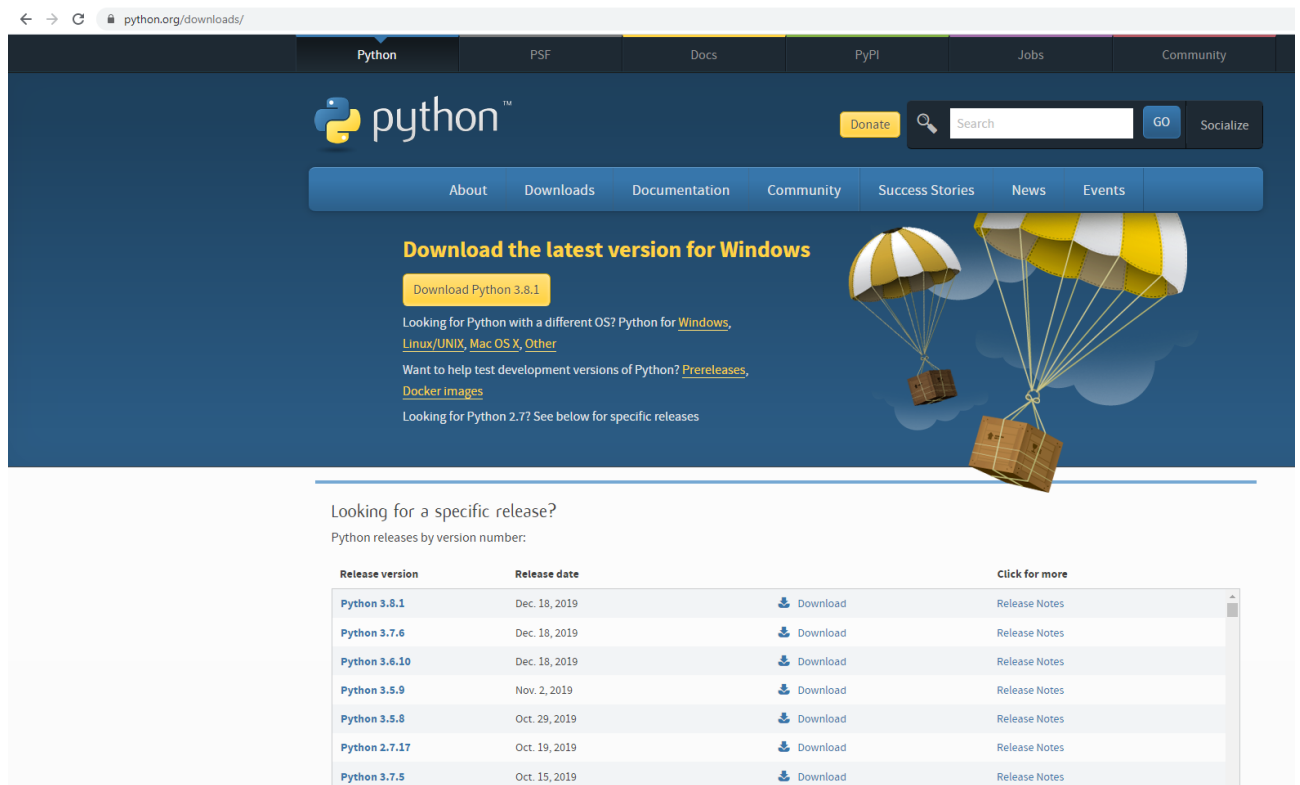


Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.
2. Выбрать способ установки.



Рисунок 2. Установщик Python

3. Отметить необходимые опции установки (доступно при выборе Customize installation)



Рисунок 3. Опции установки

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Выбираю:

- Documentation – установка документаций.
- pip – установка пакетного менеджера pip.
- tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).

4. Выбираем место установки (доступно при выборе Customize installation)

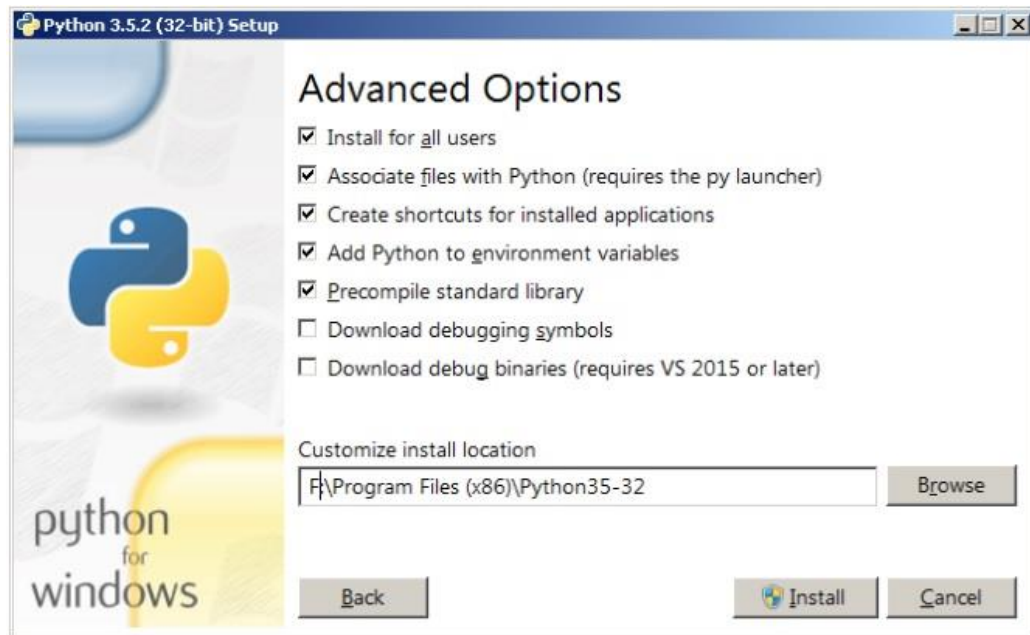


Рисунок 4. Продвинутые опции установки

5. После успешной установки:



Рисунок 5. Сообщение об установке

Окружение Python представляет собой контекст, в котором выполняется код Python. Различают глобальные, виртуальные

окружения и окружения Conda. Окружение состоит из интерпретатора, библиотеки и нескольких установленных пакетов. Вместе они определяют, какие языковые конструкции и синтаксис допустимы, какие возможности операционной системы доступны и какие пакеты можно использовать.

В Visual Studio для Windows есть окно **Окружения Python**, которое позволяет управлять окружениями и выбрать одно из них в качестве окружения по умолчанию для новых проектов.

Окружения, обнаруженные Visual Studio, отображаются в окне **Окружения Python**. Для открытия окна выберите команду меню **Просмотр > Другие окна > Окружения Python**.

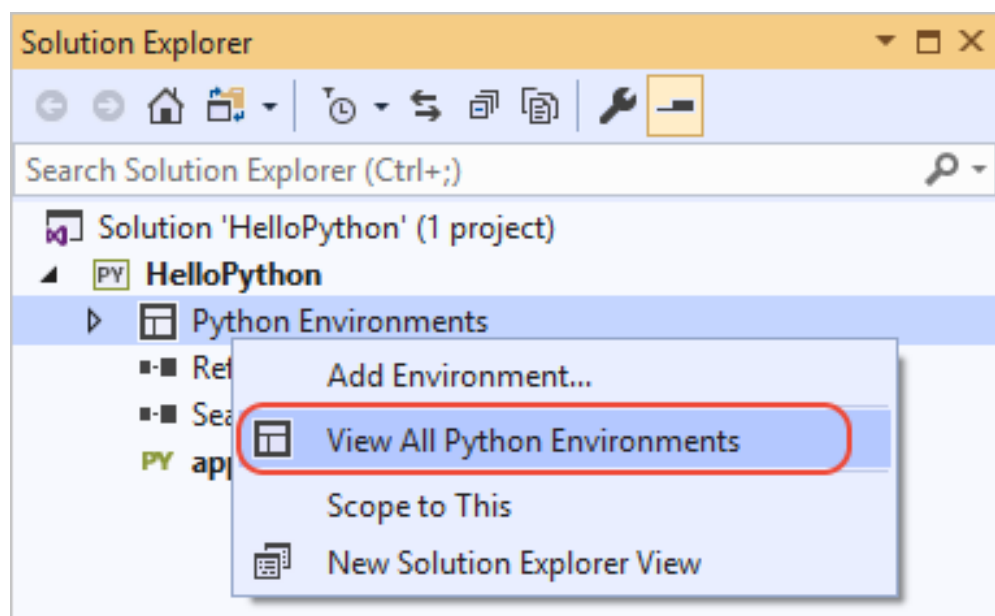


Рисунок 6. Показать Python инструменты

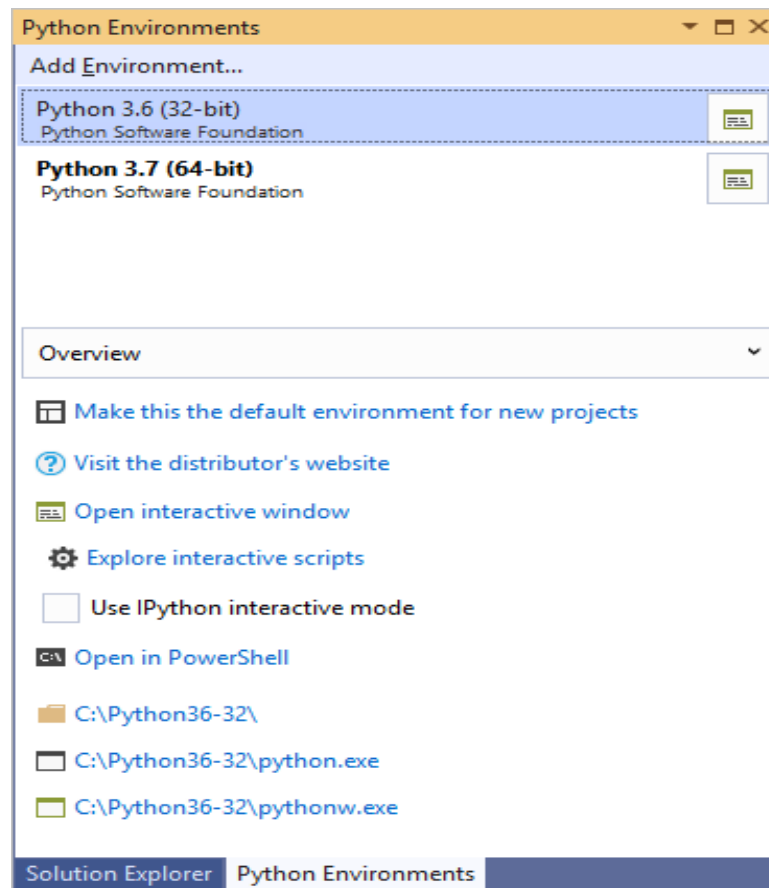


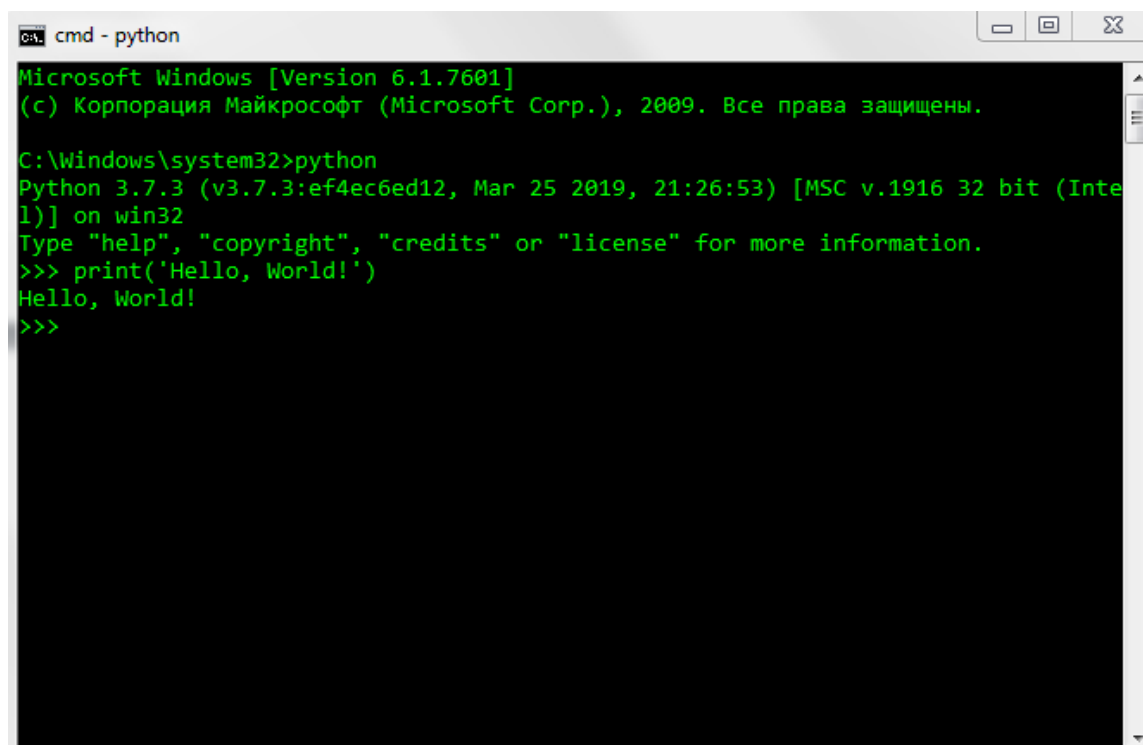
Рисунок 7. Выбор версии Python

При выборе окружения в списке на вкладке **Обзор** Visual Studio отображаются различные свойства и команды для этого окружения. Например, как видно на рисунке выше, интерпретатор находится в папке `C:\Python36-32`. Четыре команды в нижней части вкладки **Обзор** открывают командную строку с выполняющимся интерпретатором.

Справа от каждого окружения в списке есть элемент управления, который позволяет открыть **интерактивное** окно для этого окружения.

## 1.2. Техника работы в командной строке и среде IDLE

Выполняя (запуская) команду "python" в вашем терминале, вы получаете интерактивную оболочку Python.



```
cmd - python
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Windows\system32>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, World!')
Hello, World!
>>>
```

Рисунок 8. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

```
>>> exit()
```

или же

```
>>> quit()
```

Кроме того, **CTRL + D** закроет оболочку и вернет вас в командную строку терминала.

[IDLE](#) – простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

```
>>>
```

Теперь напишите в подсказке следующий код:

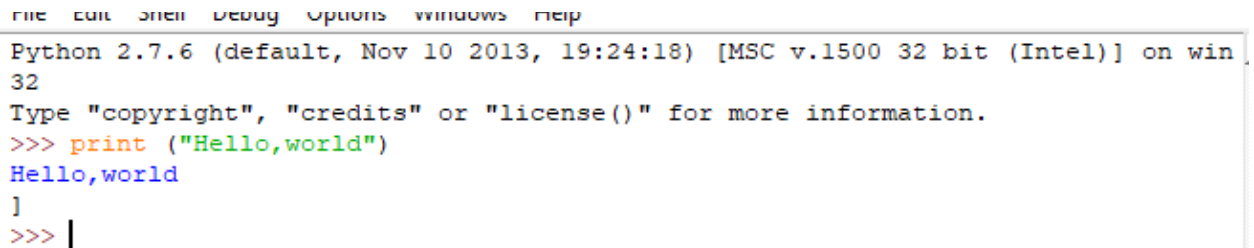
```
>>> print("Hello, World")
```

Нажмите **Enter** .

```
>> print("Hello,world")
```

```
Hello,world
```





```

File Edit Shell Debug Options Windows Help
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print ("Hello,world")
Hello,world
]
>>> |

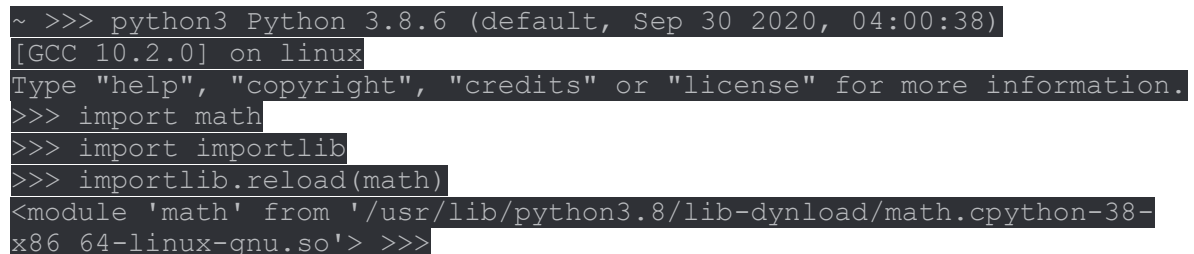
```

Рисунок 9.Первая программа

### 1.3.1.Техника работы в командной строке и среде IDLE

Продемонстрировать работу в командной строке, включая

- создание файла с кодом
- запуск
- import
- reload
- отработку ошибок



```

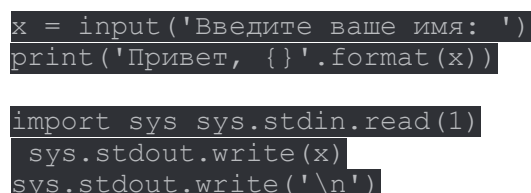
~ >>> python3 Python 3.8.6 (default, Sep 30 2020, 04:00:38)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> import importlib
>>> importlib.reload(math)
<module 'math' from '/usr/lib/python3.8/lib-dynload/math.cpython-38-
x86_64-linux-gnu.so'> >>>

```

### 1.4.1.Техника работы с линейными программами

Разработать программы по темам

- input
- Функция input() в Python, ввод данных с клавиатура.
- print
- Функция print() в Python, печатает объект.
- stdin,stdout,stderr модуля sys в Python.
  - форматная строка и метод формат



```

x = input('Введите ваше имя: ')
print('Привет, {}'.format(x))

import sys sys.stdin.read(1)
sys.stdout.write(x)
sys.stdout.write('\n')

```

#### 1.4.2. Техника работы с разветвляющимися программами Задание №1.

Место для кода

#### Задание №2.

```
# Задание 2. Разработать программу с меню для демонстрации
работы с типами данных:
# список(list), словарь(dict), множество(set)
# Меню -> выбор типа данных -> выбор метода -> краткая справка

"
list_menu = {
'append': 'добавляет элемент в конец списка',
'clear': 'удаляет все элементы из списка',
'copy': 'возвращает копию списка',
'count': 'возвращает количество элементов с заданным значением',
'extend': "добавить элементы списка (или любого итеративного) в
конец текущего списка".,
'index': 'возвращает индекс первого элемента с указанным
значением',
'insert': 'добавляет элемент в указанную позицию',
'pop': 'удаляет элемент в указанном положении',
'remove': 'удаляет первый элемент с указанным значением',
'reverse': 'меняет порядок списка',
'sort': 'сортировка списка'
}

dict_menu = {
'clear': 'удаляет все элементы из словаря',
'copy': 'возвращает копию словаря',
'fromkeys': 'возвращает словарь с указанными ключами и
значением',
'get': 'возвращает значение указанного ключа',
'items': 'возвращает список, содержащий Кортеж для каждой пары
ключ-значение',
'keys': 'возвращает список, содержащий ключи словаря',
'pop': 'удаляет элемент с указанным ключом',
'popitem': 'удаляет последнюю вставленную пару ключ-значение',
'setdefault': 'возвращает значение указанного ключа. Если ключ
не существует: вставьте ключ с указанным значением',
'update': 'обновляет словарь с помощью указанных пар ключ-
значение',
'values': 'возвращает список всех значений в словаре'
}

set_menu = {
'add': 'добавляет элемент в набор',
'clear': 'удаляет все элементы из набора',
'copy': 'возвращает копию набора',
'difference': 'возвращает набор, содержащий разницу между двумя
или более наборами',
```

```

'difference_update': 'удаляет элементы в этом наборе, которые
также включены в другой, указанный набор',
'discard': 'удалить указанный элемент',
'intersection': 'возвращает набор, то есть пересечение двух
других наборов',
'intersection_update': 'удаляет элементы в этом наборе, которые
отсутствуют в других, указанных наборах',
'isdisjoint': 'возвращает, имеют ли два множества пересечение
или нет',
'issubset': 'возвращает, содержит ли другой набор этот набор или
нет',
'issuperset': 'возвращает, содержит ли этот набор другой набор
или нет',
'pop': 'удаляет элемент из набора',
'remove': 'удаляет указанный элемент',
'symmetric_difference': 'возвращает набор с симметричными
разностями двух наборов',
'symmetric_difference_update': 'вставляет симметричные разности
из этого набора и другого',
'union': 'возвращает набор, содержащий объединение множеств',
'update': 'обновить набор с Союзом этот набор и другие
}
"

type_menu = {'list': list_menu, 'dict': dict_menu, 'set':
set_menu}

print('\n'.join(type_menu))
sel_type = input('> ')
if sel_type not in type_menu.keys():
    print('no such entry')
    exit(1)
print('\n'.join(type_menu[sel_type].keys()))

sel_method = input('> ')
if sel_method not in type_menu[sel_type].keys():
    print('no such entry')
    exit(1)
print(type_menu[sel_type][sel_method])

```

### 1.5.1. Техника работы с циклическими программами цикл while

#### Задание №1.

# На плоскости нарисован квадрат заданного размера с левой нижней вершиной в начале координат. В квадрат вписывается окружность.

Случайным образом в квадрате выбирается 1000 точек.

а) нужно определить, сколько точек попало внутрь круга

б) считая количество точек пропорциональным площади, найти отношение площадей круга и квадрата

в) по этому отношению определить приближённое значение числа пи

г) определить, насколько найденное значение отличается от "библиотечного".

```

from random import randrange
import math

```

```

tochki = 1000
vershina = int(input())
radiys = vershina/2
schetchik = 0
for i in range(tochki):
    x = randrange(0, vershina)
    y = randrange(0, vershina)
    if ((x-radiys)**2 + (y-radiys)**2 <= radiys**2):
        schetchik += 1
otnoshenie = schetchik/tochki
p = 4*schetchik/tochki
raznica = 3.14-p

print("Точек внутри круга:", schetchik)
print("Отношение площадей круга и квадрата:", otnoshenie)
print("Приблизительное число PI:", p)
print("Отличие от библиотечного числа pi:", raznica)

```

#### Задание№2.

# Придумать пример(ы) на использование break / continue /else.

```

import random

for i in range(5):
    x = random.randint(0, 25)
    if 10 <= x <= 15:
        print('выпало число', x, 'на итерации', i)
        break
    else:
        continue
else:
    print('ни одно число не попало в промежуток 10 <= x <= 15')

```

#### 1.5.2.

#### Задание№1.

#### Задание№2.

# <https://stepik.org/lesson/3364/step/11?unit=947>

#Напишите программу, которая считывает со стандартного ввода #целые числа, по одному числу

#в строке, и после первого введенного нуля выводит сумму #полученных на вход чисел.

```

sum = 0
num = 1

while num != 0:
    num = int(input())
    sum += num

print(sum)

```

#### Задание№3.

# Разработать программу для нахождения наибольшего общего делителя

```
from collections import Counter
```

# [https://ru.wikipedia.org/wiki/Перебор\\_делителей](https://ru.wikipedia.org/wiki/Перебор_делителей)

# на больших числах типа 1ккк вроде работает быстро

```
def integer_factorization(n):
    tmp = n
    p_nums = []
    while True:
        for p_num in prime_numbers():
            if tmp % p_num == 0:
                tmp = tmp / p_num
                p_nums.append(p_num)
                break

    i = 2
    j = 0
    while (i * i <= tmp) and (j != 1):
        if tmp % i == 0:
            j = 1
            i += 1

    if j == 0:
        p_nums.append(int(tmp))
        break
    return Counter(p_nums)
```

```
def prime_numbers():
    n = 2
    while True:
        i = 2
        j = 0
        while (i * i <= n) and (j != 1):
            if n % i == 0:
                j = 1
                i += 1

        if j != 1:
            yield n

        n += 1
```

```
def gcd(nums):
    diff = integer_factorization(nums[0]) &
integer_factorization(nums[1])
    if diff == Counter():
        return None
    else:
        gcd_n = 1
        for n in diff:
```

```

        gcd_n *= n
    return gcd_n

n1 = int(input('Введите первое число НОД: '))
n2 = int(input('Введите второе число НОД: '))
print(gcd([n1, n2]))

```

#### Задание №4.

# С использованием результата задания 2 разработать программу # для нахождения наименьшего  
# общего кратного

```

sum = 0
num = 1
sum1 = 0
while num != 0:
    sum1 = sum
    num = int(input())
    sum += num
i = min(sum, sum1)
while True:
    if i % sum == 0 and i % sum1 == 0:
        break
    i += 1
print(i)

```

#### Задание №5.

# <https://stepik.org/lesson/3369/step/8?unit=952>  
# напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 ...  
# (число повторяется столько раз, чему равно).  
# На вход программе передаётся неотрицательное целое число n — столько элементов последовательности должна отобразить программа.  
# На выходе ожидается последовательность чисел, записанных через пробел в одну строку.  
# Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.  
# Sample Input:  
# 7  
# Sample Output:  
# 1 2 2 3 3 3 4

```

konec = int(input())
chislo = 1
schetchik = 0
while schetchik != konec:
    for i in range(chislo):
        print(chislo, end=' ')
    schetchik += 1
    if schetchik == konec:
        break
    chislo += 1

```

### Задание №1.

#Составить и выполнить по 3 примера использования модулей для #работы с дробными числами (fractions), для точных вычислений #(decimal).

```
///1///
import fractions

>>>Fraction(153,272)          #автоматическое уменьшение дроби
Fraction(9,16)

>>>Fraction(1,2) + Fraction(3,4) #двоичные операции над дробью
Fraction(5,4)

>>>Fractions.gcd(6,9)         #Наибольший общий делитель

///2///
import decimal

>>>num1=Decimal("0.1")
>>>num2=Decimal("0.7")        #Округление десятичной дроби
>>>print(num1+num2)           с плавающей точкой
0.8

>>>getcontext().prec=2
>>>print(Decimal("4.34")/4)    #Точность в значениях
4

>>>decimal.Decimal("3.14")     #Представление в виде дроби
```

### 1.6.2.

### Задание №1.

Подготовить инструкцию по использованию модулей math и cmath.

#### Арифметические функции

Эти функции выполняют различные арифметические операции, такие как вычисление пола, потолка или абсолютного значения числа с использованием функций floor(x), ceil(x) и fabs(x) соответственно. Функция ceil(x) вернет наименьшее целое число, которое больше или равно x. Аналогично, floor(x) возвращает наибольшее целое число, меньшее или равное x. Функция fabs(x) возвращает абсолютное значение x. Вы также можете выполнять нетривиальные операции, такие как вычисление факториала числа с использованием factorial(x). Факториал является произведением целого числа и всех положительных целых чисел, меньших его. Он широко используется при работе с комбинациями и перестановками. Его также можно использовать для вычисления значения функций синуса и косинуса.

#### Тригонометрические функции

Эти функции связывают углы треугольника по бокам. У них много приложений, в том числе изучение треугольников и моделирование периодических явлений, таких как звуковые и световые волны. Имейте в виду, что угол, который вы предоставляете, находится в радианах. Вы можете рассчитать sin(x), cos(x) и tan(x) непосредственно с помощью этого модуля. Однако нет прямой формулы для вычисления cosec(x), sec(x) и cot(x), но их значение равно обратному значению,

возвращаемому  $\sin(x)$ ,  $\cos(x)$  и  $\tan(x)$  соответственно. Вместо того, чтобы вычислять значение тригонометрических функций под определенным углом, вы также можете сделать обратный и рассчитать угол, в котором они имеют заданное значение, используя  $\text{asin}(x)$ ,  $\text{acos}(x)$  и  $\text{atan}(x)$ .

#### Гиперболические функции

Гиперболические функции являются аналогами тригонометрических функций, которые основаны на гиперболе вместо круга. В тригонометрии точки  $(\cos b, \sin b)$  представляют точки единичного круга. В случае гиперболических функций точки  $(\cosh b, \sinh b)$  представляют точки, которые образуют правую половину равносторонней гиперболы. Точно так же, как тригонометрические функции, вы можете непосредственно вычислить значение  $\sinh(x)$ ,  $\cosh(x)$  и  $\tanh(x)$ . Остальные значения могут быть рассчитаны с использованием различных отношений между этими тремя значениями. Существуют также другие функции  $\text{asinh}(x)$ ,  $\text{acosh}(x)$  и  $\text{atanh}(x)$ , которые могут быть использованы для вычисления обратных соответствующих гиперболических значений.

#### Сложные числа

Сложные числа хранятся внутри с использованием прямоугольных или декартовых координат. Комплексное число  $z$  будет представлено в декартовых координатах как  $z = x + iy$ , где  $x$  представляет действительную часть, а  $y$  представляет собой мнимую часть.

Другим способом их представления является использование полярных координат.

В этом случае комплексное число  $z$  будет определяться комбинацией модуля  $r$  и фазового угла  $\phi$ . Модуль  $r$  является расстоянием между комплексным числом  $z$  и началом. Угол  $\phi$  - угол против часовой стрелки, измеренный в радианах от положительной оси  $x$  до отрезка линии, соединяющего  $z$  и начало координат.

При работе с комплексными числами модуль `cmath` может оказать большую помощь.

Модуль комплексного числа может быть рассчитан с использованием встроенной функции `abs()`, и его фаза может быть рассчитана с использованием функции `phase(z)`, доступной в модуле `cmath`. Вы можете преобразовать комплексное число в прямоугольной форме в полярную форму, используя `polar(z)`, которая вернет пару  $(r, \phi)$ , где  $r = \text{abs}(z)$ , а  $\phi = \text{phase}(z)$ .

#### Заключение

Все эти функции, о которых мы говорили выше, имеют свои конкретные приложения.

Например, вы можете использовать функцию `factorial(x)` для решения проблем с перестановкой и комбинацией. Вы можете использовать тригонометрические функции для преобразования вектора в декартовы координаты. Вы также можете использовать тригонометрические функции для имитации периодических функций, таких как звуковые и световые волны.

Аналогично, кривая веревки, висящая между двумя полюсами, может быть определена с использованием гиперболической функции. Поскольку все эти функции доступны непосредственно в модуле `math`, очень легко создавать небольшие программы, которые выполняют все эти задачи.

#### 1.7.1. Техника работы со строками

##### Задание №1.

`#https://stepik.org/lesson/201702/step/5?unit=175778`

#С клавиатуры вводятся строки, последовательность заканчивается #точкой. Выведите буквы введенных слов в верхнем регистре, #разделяя их пробелами.

```
slovo = input()
while slovo != ".":
    print(" ".join(slovo.upper()))
    slovo = input()
```



#### Задание№2.

<https://stepik.org/lesson/201702/step/8?unit=175778>

#Известно, что для логина часто не разрешается использовать #строки содержащие пробелы. Но пользователю нашего сервиса #особенно понравилась какая-то строка. Замените пробелы в строке #на символы нижнего подчеркивания, чтобы строка могла сгодиться #для логина. Если строка состоит из одного слова, менять ничего #не нужно.

#Sample Input: python sila

#Sample Output: python\_sila

```
print(input().replace(' ', '_'))
```

#### Задание№3.

<https://stepik.org/lesson/201702/step/9?unit=175778>

#Уберите точки из введенного IP-адреса. Выведите сначала четыре #числа через пробел, а затем сумму получившихся чисел.

Sample Input:

192.168.0.1

Sample Output:

192 168 0 1

361

```
chisla = input().replace(".", " ")
```

```
print(chisla)
```

```
chisla =chisla.split()
```

```
print(int(chisla[0])+int(chisla[1])+int(chisla[2])+int(chisla[3])
)
```

#### Задание№4.

<https://stepik.org/lesson/201702/step/14?unit=175778>

Программист логирует программу, чтобы хорошо знать, как она себя ведет (эта весьма распространенная и важная практика). Он использует разные типы сообщений для вывода ошибок (error), предупреждений (warning), информации (info) или подробного описания (verbose). Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы, которые отличают сообщения друг от друга, позволяя программисту понять, к какому из типов относится сообщения. Модификаторы состоят из двух одинаковых символов и записываются по разу в начале и в конце строки.

@@ обозначает ошибку

!! обозначает предупреждение

// обозначает информационное сообщение

\*\* обозначает подробное сообщение

Напишите программу, которая принимает строки до точки и выводит, какого типа это сообщение. Если сообщение не содержит модификаторов, проигнорируйте его.

Sample Input:

!! cannot resolve this method !!

@@ invalid type @@

@@ StackOverflowException @@

// here I change the variables name //

\*\* this class is used for operating with the database, including CRUD operations and registering new users \*\*

error on line 42

// TODO: optimize recursive calls //

Sample Output:

предупреждение

ошибка

ошибка

информация

подробное сообщение

информация

```
while True:
    b = input()
    if b == "***":
        print("подробное сообщение")
    elif "@" in b:
        print("ошибка")
    elif "!!" in b:
        print("предупреждение")
    elif "/" in b:
        print("информация")
    elif "." in b:
        break
```

1.7.2.

Задание №1.

#Подготовить сравнительную инструкцию по использованию

#форматирования строк

## f-строки в Python

Начиная с версии 3.6 в Python появился новый тип строк — f-строки, которые буквально означают «formatted string». Эти строки улучшают читаемость кода, а также работают быстрее чем другие способы форматирования.

1. Конкатенация. Грубый способ форматирования, в котором мы просто склеиваем несколько строк с помощью операции сложения:

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут " + name + ". Мне " + str(age) + " лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

2. %-форматирование. Самый популярный способ, который перешел в Python из языка C. Передавать значения в строку можно через списки и кортежи, а также и с помощью словаря. Во втором случае значения помещаются не по позиции, а в соответствии с именами.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут %s. Мне %d лет." % (name, age))
>>> Меня зовут Дмитрий. Мне 25 лет.
>>> print("Меня зовут %(name)s. Мне %(age)d лет." % {"name":
name, "age": age})
>>> Меня зовут Дмитрий. Мне 25 лет.
```

3. Template-строки. Этот способ появился в Python 2.4, как замена %-форматированию (PEP 292), но популярным так и не стал. Поддерживает передачу значений по имени и использует \$-синтаксис как в PHP.

```
>>> from string import Template
>>> name = "Дмитрий"
>>> age = 25
>>> s = Template('Меня зовут $name. Мне $age лет.')
>>> print(s.substitute(name=name, age=age))
>>> Меня зовут Дмитрий. Мне 25 лет.
```

4. Форматирование с помощью метода format(). Этот способ появился в Python 3 в качестве замены %-форматированию. Он также поддерживает передачу значений по позиции и по имени.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print("Меня зовут {}. Мне {} лет.".format(name, age))
>>> Меня зовут Дмитрий. Мне 25 лет.
>>> print("Меня зовут {name} Мне {age} лет.".format(age=age,
name=name))
>>> Меня зовут Дмитрий. Мне 25 лет.
```

5. f-строки. Форматирование, которое появилось в Python 3.6 (PEP 498). Этот способ похож на форматирование с помощью метода format(), но гибче, читабельней и быстрее.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

### Погружение в F-строки

f-строки делают очень простую вещь — они берут значения переменных, которые есть в текущей области видимости, и подставляют их в строку. В самой строке вам лишь нужно указать имя этой переменной в фигурных скобках.

```
>>> name = "Дмитрий"
>>> age = 25
>>> print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

f-строки также поддерживают расширенное форматирование чисел:

```
>>> from math import pi
>>> print(f"Значение числа pi: {pi:.2f}")
>>> Значение числа pi: 3.14
```

С помощью f-строк можно форматировать дату без вызова метода strftime():

```
>>> from datetime import datetime as dt
>>> now = dt.now()
>>> print(f"Текущее время {now:%d.%m.%Y %H:%M}")
>>> Текущее время 24.02.2017 15:51
```

Они поддерживают базовые арифметические операции. Да, прямо в строках:

```
>>> x = 10
>>> y = 5
>>> print(f"{x} x {y} / 2 = {x * y / 2}")
>>> 10 x 5 / 2 = 25.0
```

Позволяют обращаться к значениям списков по индексу:

```
>>> planets = ["Меркурий", "Венера", "Земля", "Марс"]
>>> print(f"Мы живим не планете {planets[2]}")
>>> Мы живим не планете Земля
```

А также к элементам словаря по ключу:

```
>>> planet = {"name": "Земля", "radius": 6378000}
```

```
>>> print(f"Планета {planet['name']}. Радиус
{planet['radius']/1000} км.")
>>> Планета Земля. Радиус 6378.0 км.
```

Причем вы можете использовать как строковые, так и числовые ключи. Точно также как в обычном Python коде:

```
>>> digits = {0: 'ноль', 'one': 'один'}
>>> print(f"0 - {digits[0]}, 1 - {digits['one']}")
>>> 0 - ноль, 1 - один
```

Вы можете вызывать в f-строках методы объектов:

```
>>> name = "Дмитрий"
>>> print(f"Имя: {name.upper()}")
>>> Имя: ДМИТРИЙ
```

А также вызывать функции:

```
>>> print(f"13 / 3 = {round(13/3)}")
>>> 13 / 3 = 4
```

f-строки очень гибкий и мощный инструмент для создания самых разнообразных шаблонов.

### 1.8.1. Техника работы со списками

### 1.8.2.

#### 1.9.1. Техника работы с циклом for и генераторами списков

##### Задание №1.

```
# Каждый студент может программировать только на одном языке
# и занимать только одну позицию.
# Дан текстовый файл, содержащий перечень студентов с указанием языка и позиции
# (каждый студент с новой строки)
# Требуется
# 1. Получить список студентов с указанием языка и позиции
# 2. Сформировать список всевозможных команд
# 3. Вывести список команд с указанием состава и названия команды:
# Команда 1
# coder: ...
# designer: ...
# tester: ...
# writer: ...
#
# Команда 2
# ...
# Пункты 1 и 2 выполнить с использованием генераторов списка
# Name Team Speciality Lang
```

```
import sys
```

```
print('Team analyzer v0.1')
```

```
if len(sys.argv) > 1:
    filename = sys.argv[1]
else:
    filename = 'teams.txt'
```

```

file = open(filename)

people = [x.split() for x in file]
#print(people)
print('Lines readed:', len(people))
teams = {x[1] for x in people}

specialities = ['coder', 'writer', 'tester', 'designer']

for team in teams:
    print(team)
    team_members = [x for x in people if x[1] == team]
    for speciality in specialities:
        member = [x for x in team_members if x[2] == speciality]
        if len(member) == 0:
            print(' {}: no member with this
speciality'.format(speciality))
        else:
            member = member[0]
            print(' {}: {} {}'.format(speciality, member[0],
member[3]))

```

### 1.9.2.

#### Задание №5.

# Matrix56. Дана матрица размера  $M \times N$  ( $N$  — четное число). Поменять местами  
# левую и правую половины матрицы.

```

from random import randint
import sys

if len(sys.argv) < 3:
    print('too few arguments')
    exit(1)

M = int(sys.argv[1])
N = int(sys.argv[2])

if (N % 2) != 0:
    print('N is not even')
    exit(1)

def print_matrix(mat):
    for row in mat:
        print(' '.join(map(str, row)))

matrix = [[randint(0, 9) for _ in range(N)] for _ in range(M)]
print('original:')
print_matrix(matrix)

for row in matrix:
    for x in range(N//2):
        # print(x, N-x-1)
        row[x] += row[N-x-1]

```

```

row[N-x-1] = row[x] - row[N-x-1]
row[x] = row[x] - row[N-x-1]

print('mirrored:')
print_matrix(matrix)

```

#### 1.10.1. Техника работы с функциями

##### Задание №2.

# Func6. Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых чисел от A до B включительно (A и B — целые). Если A > B, то функция возвращает 0.  
# С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.

```

def SumRange(A: int, B: int) -> int:
    return sum(range(A, B+1))

```

```

A = int(input('A: '))
B = int(input('B: '))
C = int(input('C: '))

```

```

print('sum A -> B:', SumRange(A, B))
print('sum B -> C:', SumRange(B, C))

```

#### 1.10.2.

##### Задание №3.

# Использовать lambda, filter.  
# Array55. Дан целочисленный массив A размера N (<= 15). Переписать в новый целочисленный массив B все элементы с нечетными порядковыми номерами (1, 3, ...) и вывести размер полученного массива B и его содержимое. Условный оператор не использовать.

```

print(
    list(
        map(lambda x: x[1],
            filter(lambda x: (x[1] % 2) != 0,
                enumerate(
                    map(int, input(': ').split())
                )
            )
        )
    )
)

```

#### 1.11.1. Техника работы со словарями

##### 1.11.2.

#### 1.12.1. Техника работы с множествами

##### 1.12.2.

#### 1.13.1. Техника работы с кортежами

##### 1.13.2.

#### 1.14.1. Техника работы с файлами

##### 1.14.2.

#### 1.15.1. Техника работы с модулями

##### 1.15.2.

##### 1.15.3.

##### 1.15.4.

#### 1.16.1. Техника работы с классами

1.16.2.

1.16.3.

1.16.4.