



Государственное бюджетное образовательное учреждение высшего образования  
Московской области

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**  
имени дважды Героя Советского Союза, летчика-космонавта А.А. Леонова

---

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

Выполнили студенты:

Митюшин П. А. и Слепов А. Д.

Группа: П1-18

Преподаватель:

Гусятинер Л. Б.

Королев, 2020

## Оглавление

<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 .....	6
1.3 Техника работы с линейными и разветвляющимися программами .....	9
1.4. Техника работы с циклическими программами _ цикл while .....	20
1.5. Техника работы с числами.....	27
1.6. Техника работы со строками .....	35
1.7. Техника работы со списками.....	39
1.8. Техника работы с циклом for и генераторами списков .....	44
1.9. Техника работы с функциями .....	50
1.10. Техника работы со словарями .....	53
1.11. Техника работы с множествами .....	59
1.12. Техника работы с кортежами .....	66
1.13. Техника работы с файлами .....	70
1.14. Техника работы с модулями .....	73
1.15. Техника работы с классами .....	87

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

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

Открываем браузер и переходим на страницу официального сайта python: <https://www.python.org/> на главной странице будет кнопка download нажимаем на неё и под надписью Download the latest version for windows.

Запускаем установочный файл

После того, как вы выбрали установочный файл и загрузили его, просто запустите его двойным нажатием на загруженный файл. Затем открывается диалоговое окно, которое представлено на рис. 1:

Запускаем загрузочный файл и проходим процесс установки .



Рис. 1. Установщик Python

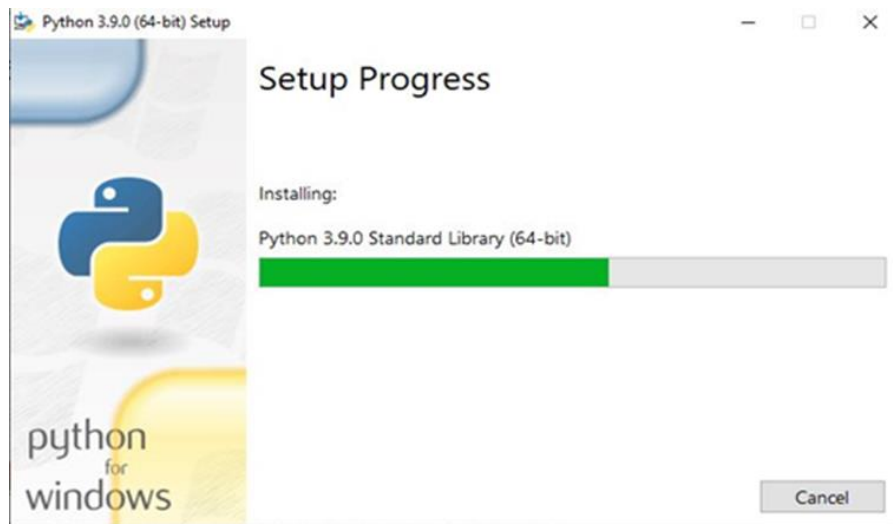


Рис. 2. Процесс установки

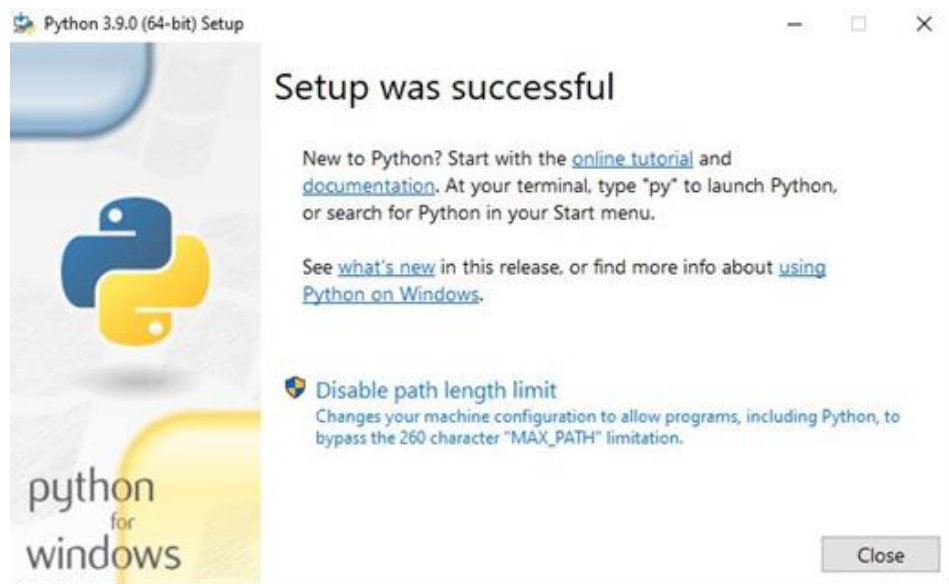


Рис. 3. Окно успешной установки

Процесс удаления описанный ниже нужен на случай, если вы установили 64 разрядную версию хотя вам надо было установить 32 разрядную версию python.

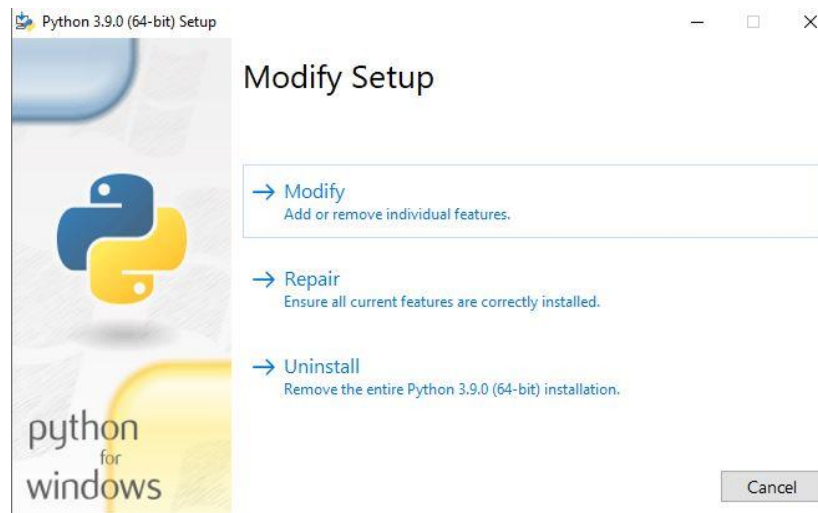


Рис. 4. Удаление python

Запускаем ранее использованный файл установки. Затем просто нажимаем на Uninstall и ждём завершения процесса удаления. На этом процесс удаления завершён.

Для **Ubuntu** установка проще. Просто открыв терминал вводим команду “sudo apt-get update”, а затем устанавливаем python “sudo apt-get install python”(Писать естественно без кавычек).

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

В PyCharm можно выбрать версию python когда вы в начале запускаете программу. При запуске файла в PyCharm, всплывает диалоговое окно как показанное на рис 1.

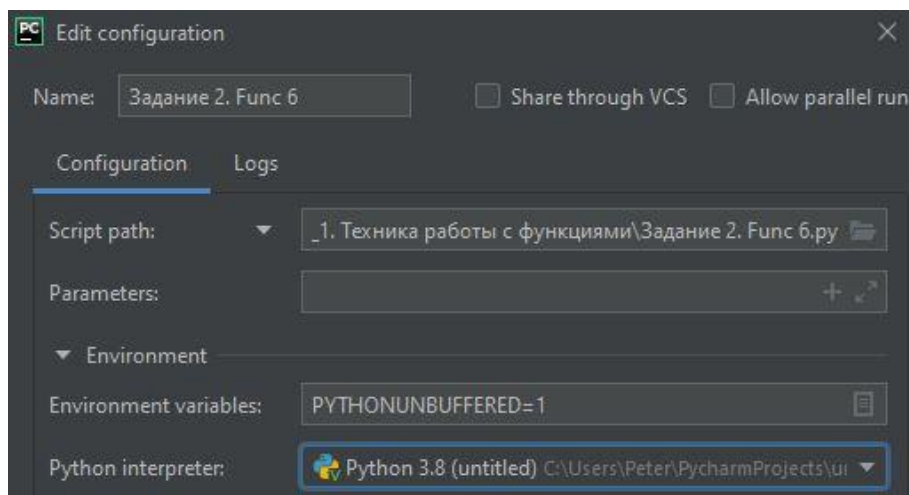


Рис. 1. Выбор интерпретатора

После того как выбрали версию языка как на рис. 1, затем нажимаем на внизу располагавшуюся кнопку run. После этого программа будет запущена на выбранной вами ранее версии python.

Выполняя команду “python” в вашем терминале, вы получаете интерактивную оболочку Python это показано на рис 2.

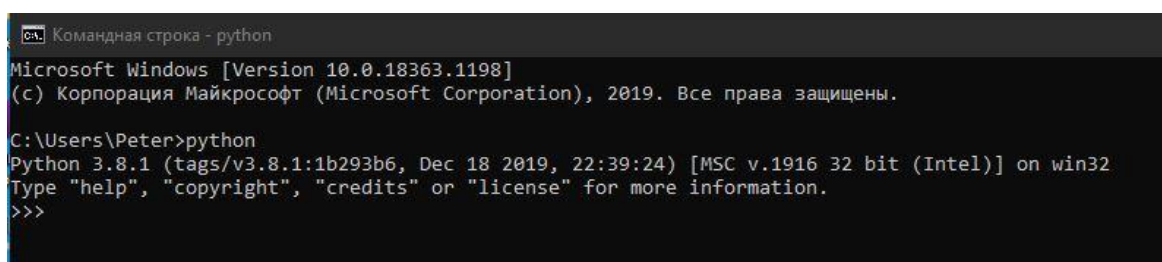


Рис. 2. Интерактивная оболочка python

[IDLE](#) - простой редактор для Python, который устанавливается вместе с Python. IDLE как правило находится в меню пуск. После того как вы нажмёте на IDLE в меню пуск то вы увидите окно показанное на рис 3.

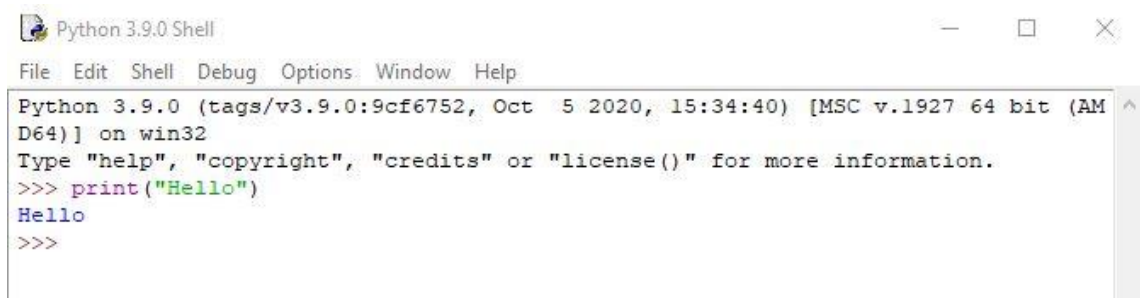


Рис. 3. Демонстрация работы IDLE

В оболочке есть подсказка из трех прямоугольных скобок:

```
>>>
```

Теперь напишите в подсказке следующий код рис. 3:

```
>>> print("Hello")
```

Нажмите Enter

```
>>> print("Hello")
```

```
Hello
```



Рис 4. Инициализация переменной

Также, если инициализировать переменную то можно вывести её просто напечатав название переменной и нажать на Enter как это показано на рис 4.

IDLE можно создавать файлы, где можно будет реализовать код. Затем их же можно и запустить. Сверху надо нажать кнопку file -> new file затем нажимаем снова file -> save as...

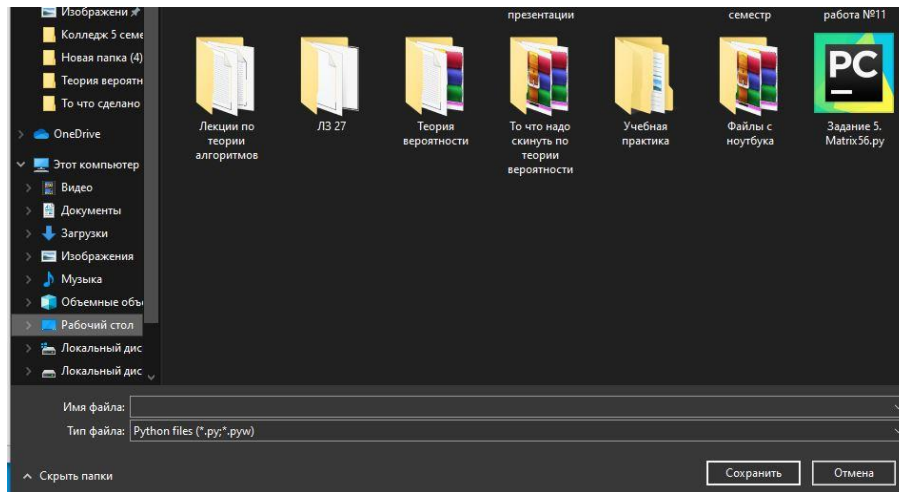


Рис. 5. Сохранение файла

Выбираем путь и в поле имя файла пишем названия файла нажимаем на кнопку сохранить и начинаем писать код пример на рис 5.



## 1.3 Техника работы с линейными и разветвляющимися программами

Приложения: input.py, print.py, std.py

### Листинг 1. input.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание. Разработать программы по темам
- input
Функция input() в Python, ввод данных с клавиатуры.
'''

name = input('Enter your name: ')
print('Hello, ', name)
```

### Листинг 2. print.py

```
'''
Выполнил: Митюшин Пётр. П1-18
- print
Функция print() в Python, печатает объект.
'''

import sys

print('Hello')
print("Hello")
objects = [1, 2, 3, 4, 5, 6, 7]
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=True) #
если True поток будет сброшен в указанный файл file принудительно.
Значение по умолчанию False
print(*objects, sep='; ')
print('Hello, {0}, {1}!'.format(input(), input()))
print('{:f}!\n'.format(3.14))
```

### Листинг 3. std.py, file.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
- stdin, stdout, stderr модуля sys
Объекты stdin, stdout, stderr модуля sys в Python.
'''

import sys
import time

stdout = sys.stdout
stderr = sys.stderr

lst = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

try:
    sys.stdout = open('file.txt', 'w')
    for i in lst:
        sys.stdout.write(i + '\n')
finally:
    sys.stdout.close()
    sys.stdout = stdout

def teleprint(*args, delay=0.05, str_join=' '):
    text = str_join.join(str(x) for x in args)
    n = len(text)
    for i, char in enumerate(text, 1):
        if i == n:
            char = f'{char}\n'
        sys.stdout.write(char)
        sys.stdout.flush()
        time.sleep(delay)

teleprint('Печать с задержкой!', 10, 12.5, 'Super!!!', delay=0.07)
```

## Листинг 4. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Печать даты прописью
'''

date = input().split('.')

day_of_the_month = {'01': "Первое", '02': "Второе", '03': "Третье",
'04': "Четвёртое", '05': "Пятое", '06': "Шестое", '07': "Седьмое",
'08': "Восьмое", '09': "Девятое", '10': "Десятое",
'11': "Одиннадцатое", '12': "Двенадцатое", '13': "Тринадцатое",
'14': "Четырнадцатое", '15': "Пятнадцатое", '16': "Шестнадцатое",
'17': "Семнадцатое", '18': "Восемнадцатое", '19': "Девятнадцатое",
'20': "Двадцатое", '21': "Двадцать первое", '22': "Двадцать второе",
'23': "Двадцать третье", '24': "Двадцать четвёртое", '25': "Двадцать
пятое", '26': "Двадцать шестое", '27': "Двадцать седьмое", '28':
"Двадцать восьмое", '29': "Двадцать девятое", '30': "Тридцатое", '31':
"Тридцать первое",
}

month = {'01': "января", '02': "февраля", '03': "марта", '04':
"апреля", '05': "мая", '06': "июня", '07': "июля", '08':
"августа", '09': "сентября", '10': "октября", '11': "ноября", '12':
"декабря",
}

day_in_years = {'1': "первого", '2': "второго", '3': "третьего", '4':
"четвёртого", '5': "пятого", '6': "шестого", '7': "седьмого", '8':
"восьмого", '9': "девятого",
}

year_list_dec =
['', 'десять', 'двадцать', 'тридцать', 'сорок', 'пятьдесят', 'шестьдесят', 'с
емьдесят', 'восемьдесят', 'девяносто']
year_list_dec2 = ['', 'десятого', 'двадцатого', 'тридцатого',
'сорокового', 'пятидесятого', 'шестидесятого', 'семидесятого',
'восемидесятого', 'девяностого']
year_list_unit = ['', 'первого', 'второго', 'третьего', 'четвёртого',
```

```

'пятого', 'шестого', 'седьмого', 'восьмого', 'девятого',
'одиннадцатого', 'двенадцатого', 'тринадцатого', 'четырнадцатого',
'пятнадцатого', 'шестнадцатого', 'семнадцатого', 'восемнадцатого',
'девятнадцатого']

year_list_hun =
['', 'сто', 'двести', 'триста', 'четыреста', 'пятьсот', 'шестьсот', 'семьсот',
'восемьсот', 'девятьсот']

year_list_hun2 = ['',
'сотого', 'двухсотого', 'трёхсотого', 'четырёхсотого', 'пятьсотого', 'шесть
сотого', 'семьсотого', 'восьмисотого', 'девятьсотого']

year_list_th = ['', 'тысяча', 'две тысячи', 'три тысячи']

year_list_th2 = {'1': 'тысячного', '2': 'двух тысячного', '3': 'трёх
тысячного'}

def definition_years(string):
if (len(string) == 1):
print(day_in_years[string[0]] + " года")
elif (len(string) == 2):
if string[1] == "0":
print(year_list_dec2[int(string[0])], "года")
elif string[0] == "1":
print(year_list_unit2[int(string[1])], "года")
else:
print(year_list_dec[int(string[0])], day_in_years[string[1]], "года")
elif (len(string) == 3):
if (string[0] != "0" and string[1] == "0" and string[2] == "0"):
print(year_list_hun2[int(string[0])], "года")
elif (string[0] != "0" and string[1] != "0" and string[2] == "0"):
print(year_list_hun[int(string[0])], year_list_dec2[int(string[1])],
"года")
else:
print(year_list_hun[int(string[0])], year_list_dec[int(string[1])],
day_in_years[string[2]], "года")
else:

```

```

if string[1] == "0" and string[2] == '0' and string[3] == '0':
print(year_list_th2[string[0]], "года")
elif string[1] != "0" and string[2] == '0' and string[3] == '0':
print(year_list_th2[string[0]], year_list_hun2[int(string[1])],
"года")
else:
print(year_list_th[int(string[0])], end=" ")# первая цифра в году
if (string[1] == string[2] == string[3]):
print(year_list_hun[int(string[1])], end=" ") #,
year_list_dec[int(string[2])], year_list_unit[int(string[3])],end=" ")
else:
print(year_list_hun[int(string[1])], end=" ")# вторая цифра в году
if string[1] != "0" and string[3] == '0':
print(year_list_dec2[int(string[2])], "года",end=" ")# третья цифра в
году
elif string[1] == "0" and string[3] == '0':
print(year_list_dec2[int(string[2])], "года")# третья цифра в году
elif string[1] != string[2] == string[3]:
print(year_list_unit2[int(string[3])], "года", end=" ")
elif string[2] == "1" and string[3] != "0":
print(year_list_unit2[int(string[3])], "года", end=" ")# четвёртая
цифра в году
else:
print(year_list_dec[int(string[2])], day_in_years[string[3]],"года",
end=" ")

date = input().split('.')

day = day_of_the_month[date[0]]
mon = month[date[1]]
print(day, mon, end=" ")
definition_years(date[2])

```

## Листинг 5. Задание2.py

'''

Выполнил: Митюшин Пётр. П1-18

*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

'''

```
def work_with_list():                                # Списки
    print("1. append")
    print("2. extend")
    print("3. insert")
    print("4. remove")
    print("5. pop")
    print("6. index")
    print("7. count")
    print("8. sort")
    print("9. reverse")
    print("10. copy")
    print("11. clear")
    print("exit")
    while (True):
        com = input()
        if com == "1":
            print("list.append(x)")
            print("Добавляет элемент в конец списка")
        elif com == "2":
            print("list.extend(L)")
            print("Расширяет список a, ", end="")
            print("добавляя в конец все элементы списка b", end="")
        elif com == "3":
            print("list.insert(i, x)")
            print("Вставляет на i-ый элемент значение x")
        elif com == "4":
            print("list.remove(x)")
            print("Удаляет первый элемент в списке, имеющий значение")
            print("x. ValueError, ", end="")
            print("если такого элемента не существует", end="")
        elif com == "5":
            print("list.pop([i])")
            print("Удаляет i-ый элемент и возвращает его. ", end="")
            print("Если индекс не указан, ", end="")
```

```

        print("удаляется последний элемент", end="")
elif com == "6":
    print("list.index(x, [start [, end]])")
    print("Возвращает положение первого элемента", end="")
    print("со значением x", end="")
    print("(при этом поиск ведется от start до end)", end="")
elif com == "7":
    print("list.count(x)")
    print("Возвращает количество элементов со значением x")
elif com == "8":
    print("list.sort([key=функция])")
    print("Сортирует список на основе функции")
elif com == "9":
    print("list.reverse()")
    print("Разворачивает список")
elif com == "10":
    print("list.copy()")
    print("Поверхностная копия списка")
elif com == "11":
    print("list.clear()")
    print("Очищает список")
elif com == "exit":
    main()

```

```

def work_with_dict():                                # Словари
    #d = {}
    print("1. clear")
    print("2. copy")
    print("3. fromkeys")
    print("4. get")
    print("5. items")
    print("6. keys")
    print("7. pop")
    print("8. popitem")
    print("9. setdefault")
    print("10. update")
    print("11. values")
    print("exit")

```

```

while (True):
    com = input()
    if com == "1":
        print("dict.clear()")
        print("Очищает словарь.")
    elif com == "2":
        print("dict.copy()")
        print("Возвращает копию словаря.")
    elif com == "3":
        print("classmethod dict.fromkeys(seq[, value])")
        print("Создает словарь с ключами из seq ", end="")
        print("и значением value (по умолчанию None).", end="")
    elif com == "4":
        print("dict.get(key[, default])")
        print("Возвращает значение ключа, ", end="")
        print("но если его нет, не бросает исключение, ", end="")
        print("а возвращает default (по умолчанию None).", end="")
    elif com == "5":
        print("dict.items()")
        print("Возвращает пары (ключ, значение).")
    elif com == "6":
        print("dict.keys()")
        print("Возвращает ключи в словаре.")
    elif com == "7":
        print("dict.pop(key[, default])")
        print("Удаляет ключ и возвращает значение.", end="")
        print("Если ключа нет, возвращает default", end="")
        print("по умолчанию бросает исключение).", end="")
    elif com == "8":
        print("dict.popitem()")
        print("Удаляет и возвращает пару ")
        print("(ключ, значение). Если словарь пуст, ", end="")
        print("бросает исключение KeyError. ", end="")
        print("Помните, что словари неупорядочены.", end="")
    elif com == "9":
        print("dict.setdefault(key[, default])")
        print("Возвращает значение ключа, ", end="")
        print("но если его нет, не бросает исключение,", end="")

```



```

        print(a создает ключ с значением default", end="")
        print("по умолчанию None).", end="")
elif com == "10":
    print("dict.update([other])")
    print("Обновляет словарь, добавляя пары (ключ, значение)")
    print("из other", end="")
    print("Существующие ключи перезаписываются.", end="")
    print("Возвращает None (не новый словарь!).", end="")
elif com == "11":
    print("dict.values()")
    print("Возвращает значения в словаре.")
elif com == "exit":
    main()

def work_with_set():
    # Множества
    #a = set()
    print("1. update")
    print("2. intersection_update")
    print("3. difference_update")
    print("4. symmetric_difference_update")
    print("5. add")
    print("6. remove")
    print("7. discard")
    print("8. pop")
    print("9. clear")
    print("10. copy")
    print("exit")
    while (True):
        com = input()
        if com == "1":
            print("set.update(other, ...); set |= other | ...")
            print("Объединение")
        elif com == "2":
            print("set.intersection_update(other, ...)")
            print("set &= other & ...")
            print("Пересечение")
        elif com == "3":
            print("set.difference_update(other, ...)")

```

```

        print("set -= other | ...")
        print("Вычитание")
    elif com == "4":
        print("set.symmetric_difference_update(other); set ^=
other ")

        print("Множество из элементов")
        print("Встречающихся в одном множестве, ", end="")
        print("но не встречающиеся в обоих", end="")
    elif com == "5":
        print("set.add(elem)")
        print("Добавляет элемент в множество.")
    elif com == "6":
        print("set.remove(elem)")
        print("Удаляет элемент из множества. ", end="")
        Print("KeyError, если такого элемента не существует")
    elif com == "7":
        print("set.discard(elem)")
        print("Удаляет элемент, если он находится в множестве")
    elif com == "8":
        print("set.pop()")
        print("Удаляет первый элемент из множества.", end="")
        print("Так как множества не упорядочены", end="")
        print("Нельзя точно сказать, ", end="")
        print("какой элемент будет первым.", end="")
    elif com == "9":
        print("set.clear()")
        print("Очистка множества")
    elif com == "10":
        print("set.copy() ")
        print("Копия множества")
    elif com == "exit":
        main()

def main():
    print("1. list")
    print("2. dict")
    print("3. set")
    com = input('Enter your choice: ')

```

```
if com == "1":
    work_with_list()
elif com == "2":
    work_with_dict()
elif com == "3":
    work_with_set()

if __name__ == "__main__":
    main()
```

#### 1.4. Техника работы с циклическими программами \_ цикл while

Приложения: K5\_1 Задание 1.ру, Задание 2.ру. K5\_2 Задание 1.ру, Задание 2.ру, Задание 3.ру, Задание 4.ру, Задание 5.ру

##### Листинг 1. Задание 1.ру

```
'''
Выполнил: Слепов Андрей. П1-18
K5_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".
'''

import math
import random

NumberOfPoints = 1000

print("Введите сторону квадрата")
a = int(input())

Ssq = a
radius = a // 2
#Scr = math.pi() * radius1

points = []

for i in range(2):
    points.append([])
    for j in range(NumberOfPoints):
        elem = random.randint(-a/2, a/2)
```

```

        points[i].append(elem)

count = 0

for i in range(NumberOfPoints):
    if math.sqrt((points[0][i]**2 - 0) + (points[1][i]**2 - 0)) <=
radius:
        count += 1

area_ratio = count / NumberOfPoints
pi = area_ratio * 4
dif_pi = math.pi - pi
print("Кол-во точек попавших внутрь круга: ", count)
print("Отношение площади круга и квадрата: ", area_ratio)
print("приблизённое значение пи относительно наших вычислений", pi)
print("Разница между нашим числом пи и библиотечным", dif_pi)

```

## Листинг 2. Задание 2.ру

```
'''
Выполнил: Митюшин Пётр. П1-18
K5_1 Техника работы с циклическими программами _ цикл while
Напишите программу, которая считывает целые числа с консоли по одному
числу в строке.

Для каждого введённого числа проверить:
если число меньше 10, то пропускаем это число;
если число больше 100, то прекращаем считывать числа;
в остальных случаях вывести это число обратно на консоль в отдельной
строке.
'''

while True:
    num = int(input())
    if num < 10:
        continue
    elif num > 100:
        break
    else:
        print(num)

'''
Найти букву и вывести её номер
'''

string = input()
find_letter = input()
for i in range(string):
    if (string[i] == find_letter):
        print(i)
        break
else:
    print("Буква в строке не найдена")
```

### Листинг 3. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while

Задание 1. Вычислить значение sin(x) с точностью до epsilon при помощи
разложения в ряд
Построить блок-схему
'''

import math

x = sin(float(input("x: ")))
epsilon = float(input("epsilon: "))
sign = 1
summ = 0
k = 3
res = (x**k)/math.factorial(k)
summ = x - res
while(res > epsilon):
    k += 2
    res = (x**k)/math.factorial(k)
    summ += sign * res
    sign *= -1
print(summ)
```

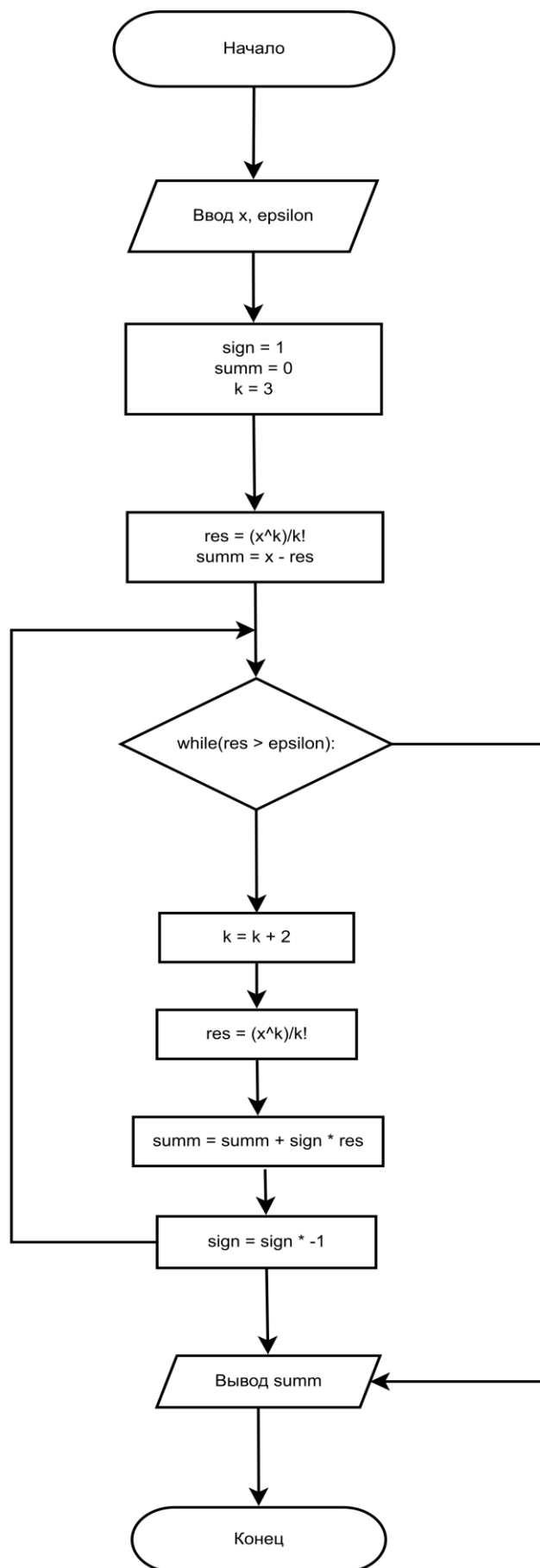


Рис 1. Блок-схема Задания 1



## Листинг 4. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K5_2 Техника работы с циклическими программами _ цикл while
Задание 2. 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.
'''

def main():
    a = int(input())
    summ = 0
    while a != 0:
        summ = summ + a
        a = int(input())
    print(summ)

if __name__ == "__main__":
    main()
```

## Листинг 5. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K5_2 Техника работы с циклическими программами _ цикл while
Задание 3. Разработать программу для нахождения наибольшего общего
делителя
'''

def gcd(n, m):
    while (n != 0 and m != 0):
        if (n >= m):
            n = n % m
        else:
            m = m % n

    return n + m
```

```
def main():
    a, b = map(int, input().split())
    print(gcd(a, b))

if __name__ == "__main__":
    main()
```

## Листинг 6. Задание4.py

'''

Выполнил: Митюшин Пётр. П1-18

K5\_2 Техника работы с циклическими программами \_ цикл while

Задание 4. С использованием результата задания 2 разработать программу для нахождения наименьшего общего кратного

'''

```
def main():
    a, b = map(int, input().split())
    m = a * b
    while (n != 0 and m != 0):
        if (a > b):
            a = a % b
        else:
            b = b % a
    print(m // (a + b))

if __name__ == "__main__":
    main()
```

## Листинг 7. Задание5.py

'''

Выполнил: Митюшин Пётр. П1-18

K5\_2 Техника работы с циклическими программами \_ цикл while

Задание 5. Напишите программу, которая выводит часть

последовательности 1 2 2 3 3 3 4 4 4 4 5 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.

```
'''
n = int(input())
x = 0
i = 0
j = 0
if n == 1:
    print(1)
else:
    while (i < n):
        i += 1
        j = 0
        while (j < i):
            if x == n:
                break
            else:
                print(i, end=' ')
                x += 1
            j += 1
```

## 1.5. Техника работы с числами.

### Листинг 1. Задание1.py

```
'''
Задание 1
Выполнил: Митюшин Пётр. П1-18
К6_1. Техника работы с числами
Составить и выполнить по 3 примера использования модулей для работы
с дробными числами (fractions), для точных вычислений (decimal).
'''

from fractions import *
from decimal import Decimal, ROUND_HALF_EVEN

number = Decimal("0.1")
number = number + number + number
print(number)

number = Decimal("0.69")
```

```

number = number * number
print(number)

a = Decimal("0.85843")
a = a.quantize(Decimal("1.00")) #quantize позволяет округлять числа
print(a)
b = Decimal("0.85843") + Decimal("0.69")
print(b.quantize(Decimal("1.00"), ROUND_HALF_EVEN))
print("-----")

a = Fraction(1, 7)
b = Fraction(1, 3)
print(a + b)
print(a - b)
print(a / b)
print(a % b)
print(a ** b)
a.limit_denominator()
print(Fraction('3.14159265359').limit_denominator(1000))
print(Fraction('3.14159265359').limit_denominator(100))
print(Fraction('3.14159265359').limit_denominator(10))
print(Fraction('3.14159265359').limit_denominator(1))
print(Fraction(1, 4))

```

## Инструкцию по использованию модулей fractions, decimal.

'''

Задание 2

Выполнил: Митюшин Пётр. П1-18

К6\_1. Техника работы с числами

Подготовить инструкцию по использованию модулей fractions, decimal.

'''

## Модуль fractions

Для начала в заголовке файла `from fractions import *`, `from decimal import *`.

Модуль `fractions` предоставляет поддержку рациональных чисел.

```
class fractions.Fraction(numerator=0, denominator=1)
class fractions.Fraction(other_fraction)
class fractions.Fraction(float)
class fractions.Fraction(decimal)
class fractions.Fraction(string)
```

Класс, представляющий собой рациональные числа. Экземпляр класса можно создать из пары чисел (числитель, знаменатель), из другого рационального числа, числа с плавающей точкой, числа типа `decimal.Decimal`, и из строки, представляющей собой число.

`Fraction.limit_denominator(max_denominator=1000000)` - ближайшее рациональное число со знаменателем не больше данного.

Также, помимо класса рациональных чисел, модуль `fractions` предоставляет функцию для нахождения наибольшего общего делителя.

`fractions.gcd(a, b)` - наибольший общий делитель чисел `a` и `b`.

## Модуль decimal

Ключевым компонентом для работы с числами в этом модуле является класс `Decimal`. Для его применения нам надо создать его объект с помощью конструктора. В конструктор передается строковое значение, которое представляет число:

Листинг 2.

```
from decimal import Decimal
number = Decimal("0.1")
```

После этого объект `Decimal` можно использовать в арифметических операциях:

Листинг 3.

```
from decimal import Decimal
number = Decimal("0.1")
number = number + number + number
print(number)
```

В операциях с Decimal можно использовать целые числа:

#### Листинг 4.

```
number = Decimal("0.1")
number = number + 2
```

Однако нельзя смешивать в операциях дробные числа float и Decimal:

#### Листинг 5.

```
number = Decimal("0.1")
number = number + 0.1    # здесь возникнет ошибка
```

С помощью дополнительных знаков мы можем определить, сколько будет символов в дробной части числа:

#### Листинг 6.

```
number = Decimal("0.10")
number = 3 * number
print(number)          # 0.30
```

Строка "0.10" определяет два знака в дробной части, даже если последние символы будут представлять ноль. Соответственно "0.100" представляет три знака в дробной части.

Объекты Decimal имеют метод `quantize()`, который позволяет округлять числа. В этот метод в качестве первого аргумента передается также объект Decimal, который указывает формат округления числа:

#### Листинг 7.

```
from decimal import Decimal
number = Decimal("0.444")
number = number.quantize(Decimal("1.00"))
print(number)          # 0.44
number = Decimal("0.555678")
print(number.quantize(Decimal("1.00"))) # 0.56
number = Decimal("0.999")
print(number.quantize(Decimal("1.00"))) # 1.00
```

Используемая строка "1.00" указывает, что округление будет идти до двух знаков в дробной части.

По умолчанию округление описывается константой `ROUND_HALF_EVEN`, при котором число округляется в большую сторону, если оно нечетное, а предыдущее перед ним больше 4. Например:

### Листинг 8.

```
from decimal import Decimal, ROUND_HALF_EVEN
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN))# 10.02
```

```
number = Decimal("10.035")
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN))# 10.04
```

Можно переопределить, используя одну из следующих констант:

**ROUND\_HALF\_UP:** округляет число в сторону повышения, если после него идет число 5 или выше

**ROUND\_HALF\_DOWN:** округляет число в сторону повышения, если после него идет число больше 5

### Листинг 9.

```
number = Decimal("10.026")
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))# 10.03
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))# 10.02
```

**ROUND\_05UP** - округляет только 0 до единицы, если после него идет 5

### Листинг 10.

```
number = Decimal("10.005")
print(number.quantize(Decimal("1.00"), ROUND_05UP)) # 10.01
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_05UP))# 10.02
```

**ROUND\_CEILING** - округляет число в большую сторону вне зависимости от того, какое число идет после него

### Листинг 11.

```
number = Decimal("10.021")
print(number.quantize(Decimal("1.00"), ROUND_CEILING))# 10.03
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_CEILING))# 10.03
```

**ROUND\_FLOOR** - не округляет число вне зависимости от того, какое число идет после него.

### Листинг 12.

```
number = Decimal("10.021")
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))# 10.02
number = Decimal("10.025")
```

```
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))# 10.02
```

### Листинг 13. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К6_2. Техника работы с числами
'''

import cmath

x = int(input())
print(cmath.polar(x))

r = int(input())
phi = int(input())
print(cmath.rect(r, phi))
print(cmath.exp(x))

y = int(input())
print(cmath.log(x, y))
print(cmath.log10(x))
print(cmath.sqrt(x))
print(cmath.acos(x))
print(cmath.asin(x))
print(cmath.atan(x))
print(cmath.cos(x))
print(cmath.sin(x))
print(cmath.tan(x))
print(cmath.acosh(x))
print(cmath.asinh(x))
print(cmath.atanh(x))
print(cmath.cosh(x))
print(cmath.sinh(x))
print(cmath.tanh(x))
print(cmath.isfinite(x))
print(cmath.isinf(x))
print(cmath.isnan(x))
print(cmath.pi)
print(cmath.e)
```



## Листинг 14. Задание2.ру

```
'''
Выполнил: Митюшин Пётр. П1-18
К6_2. Техника работы с числами
'''

import math

x = int(input())
print(math.ceil(x))

y = int(input())
print(math.copysign(x, y))
print(math.fabs(x))

print(math.factorial(x))
print(math.floor(x))
print(math.fmod(x, y))
print(math.frexp(x))

i = int(input())
print(math.ldexp(x, i))
print(math.fsum(1, 2, 3, 4, 5, 6))
print(math.isfinite(x))
print(math.isinf(x))
print(math.isnan(x))
print(math.modf(x))
print(math.trunc(x))
print(math.exp(x))
print(math.expm1(x))
print(math.log(x, y))
print(math.log1p(x))
print(math.log10(x))
print(math.log2(x))
print(math.pow(x, y))
print(math.sqrt(x))
print(math.acos(x))
print(math.asin(x))
print(math.atan(x))
```

```
print(math.atan2(y, x))
print(math.cos(x))
print(math.sin(x))
print(math.tan(x))
print(math.hypot(x, y))
print(math.degrees(x))
print(math.radians(x))
print(math.cosh(x))
print(math.sinh(x))
print(math.tanh(x))
print(math.acosh(x))
print(math.asinh(x))
print(math.atanh(x))
print(math.erf(x))
print(math.erfc(x))
print(math.gamma(x))
print(math.lgamma(x))
print(math.pi)
print(math.e)
```

## 1.6. Техника работы со строками

### Листинг 1. Задание1.py

```
"""
Выполнил: Митюшин Пётр. П1-18
K7_1. Техника работы со строками
С клавиатуры вводятся строки, последовательность заканчивается точкой.
Выведите буквы введенных слов в верхнем регистре, разделяя их
пробелами.
"""

print_string = ""
elem = input()
while (elem != "."):
    for i in range(0, len(elem)):
        if (i < len(elem) - 1):
            print_string += elem[i].upper() + ' '
        else:
            print_string += elem[i].upper()
    print(print_string)
    print_string = ""
    elem = input()
```

### Листинг 2. Задание2.py

```
"""
Выполнил: Митюшин Пётр. П1-18
K7_1. Техника работы со строками
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 согдиться для логина. Если строка состоит из одного слова,
менять ничего не нужно.
"""

string = str(input())
string = string.replace(' ', '_')
print(string)
```

### Листинг 3. Задание3.py

```
"""
Выполнил: Митюшин Пётр. П1-18
K7_1. Техника работы со строками
Уберите точки из введенного IP-адреса. Выведите сначала четыре числа
через пробел,
а затем сумму получившихся чисел.
"""

n = input().split('.')
per = int(n[0]) + int(n[1]) + int(n[2]) + int(n[3])
c = int(n[0]), int(n[1]), int(n[2]), int(n[3])
print(n[0], n[1], n[2], n[3])
print(per)
```

### Листинг 4. Задание4.py

```
"""
Выполнил: Митюшин Пётр. П1-18
K7_1. Техника работы со строками
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.
"""

string = input()
while (string != "."):
    if (string[0] == "!" and string[1] == "!"):
        print("предупреждение")
```

```

elif (string[0] == "@" and string[1] == "@"):
    print("ошибка")
elif (string[0] == "/" and string[1] == "/"):
    print("информация")
elif (string[0] == "*" and string[1] == "*"):
    print("подробное сообщение")
string = input()

```

Инструкцию по использованию. Форматирования строк “По старому”.

### Листинг 5.

```

'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''

name = "Peter"
print('Hello, %s' % name) # Вывод: "Hello, Peter"
a = 16
print('a: %d' %a) # Вывод: 16

```

%d или %s – Означает, где именно заменить значение имени, представленного в виде строки. %d – для вывода чисел, %s – для вывода строки.

### Листинг 6.

```

'''
Выполнил: Митюшин Пётр. П1-18
'''

errno = 50159747054
print('%x' % errno) # Вывод: 'badc0ffee'

%x для конвертации значения int в строку и представить его в качестве
шестнадцатеричного числа.

```

### Листинг 7.

```

'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''

```

```

er = 5
name = 'Andrey'
print('Hey %s, there is a 0d% error!' % (name, er)) # 'Hey Andrey, there
is a 5 error!'

```

Вывод нескольких переменных.

## Форматирование строк “По новому” (str.format).

Форматирование строк обрабатывается вызовом .format() в строке.

### Листинг 8.

```

'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''

print('Hello, {}'.format(name)) # Вывод: 'Hello, Bob'
print('Hey {name}, there is a 0x{errno:x} error!'.format(name=name,
errno=errno)
# Вывод: 'Hey Bob, there is a 0xbadc0ffee error!'

```

## Интерполяция строк. f-Строки.

Python 3.6 Добавил новый подход форматирования строк под названием форматированные строчные литералы, или “f-строки”. Этот новый способ форматирования строк позволяет вам использовать встроенные выражения Python внутри строковых констант.

### Листинг 9.

```

'''
Выполнил: Митюшин Пётр. П1-18
'''

name = 'Peter'
print(f'Hello, {name}!')
# Вывод: 'Hello, Peter!'

a = 5
b = 10
print(f'Five plus ten is {a + b} and not {2 * (a + b)}.')

```

```
def greet(name):
    return f"Hello, {name}!"
print(greet('Bob'))
```

Передавать значения можно так же и в функцию. После передачи значений, переданные строки соединяются и составляют итоговую строку.

#### Листинг 10.

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''

Name = 'Peter'
print(f"Hey {name}, there's a {er:#x} error!")
# Вывод: "Hey Peter, there's a 0xbadc0ffee error!"
```

### Шаблонные строки (Стандартная библиотека Template Strings)

Template strings – это более простой и менее мощный механизм, но в ряде случаев он может быть именно тем, что вам нужно.

#### Листинг 11.

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''

from string import Template
t = Template('Hey, $name!')
print(t.substitute(name=name)) # Вывод: 'Hey, Andrey!'

templ_string = 'Hey $name, there is a $error error!'
print(Template(templ_string).substitute(name=name, error=hex(errno)))
# Вывод: 'Hey Andrey, there is a 0xbadc0ffee error!'
templ_string = 'Hey $name, there is a $error error!'
```

## 1.7. Техника работы со списками

#### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''

lst = input().split()
count = 0
for i in range(2, len(lst)):
    if (int(lst[i - 2]) < int(lst[i - 1]) > int(lst[i])):
        count += 1
print(count)
```

## Листинг 2. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''

count = 0
lst = list(map(int, input().split()))
for i in range(len(lst)):
    for j in range(i+1, len(lst)):
        if lst[i] == lst[j]:
            count += 1

print(count)
```

## Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''

import random

def search_equal_items(lst):
    print(list(set(lst)))

lst = []
n = int(input())
for i in range(n):
```



```

lst.clear()
m = random.randint(5, 10)
for j in range(m):
    elem = random.randint(1, 50)
    lst.append(elem)
search_equal_items(lst)

```

#### Листинг 4. Задание1.py

```

'''
Выполнил: Слепов Андрей
K8_2. Техника работы со списками
Задание 1. Array112. Дан массив A размера N.
Упорядочить его по возрастанию методом сортировки
простым обменом («пузырьковой» сортировкой):
просматривать массив, сравнивая его соседние элементы
(A0 и A1, A1 и A2 и т. д.) и меняя их местами,
если левый элемент пары больше правого; повторить описанные
действия N - 1 раз. Для контроля за выполняемыми действиями
выводить содержимое массива после каждого просмотра.
Учесть, что при каждом просмотре количество анализируемых
пар можно уменьшить на 1.
'''

def bubble_sort(arr):
    for i in range(len(a) - 1):
        for j in range(len(a) - i - 1):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
            print(*a)
    return a

a = [int(i) for i in input().split()]
print(*bubble_sort(a))

```

#### Листинг 5. Задание2.py

```

'''
Выполнил: Слепов Андрей
K8_2. Техника работы со списками
Задание 2. Array113. Дан массив A размера N.
Упорядочить его по возрастанию методом сортировки простым
выбором: найти максимальный элемент массива и поменять его
местами с последним (N-1 м) элементом; выполнить описанные
действия N - 1 раз, каждый раз уменьшая на 1 количество
анализируемых элементов и выводя содержимое массива.
'''

```

```
'''

def sort_sel(arr):
    i = 0
    while i < len(arr) - 1:
        m = i
        j = i + 1
        while j < len(arr):
            if arr[j] < arr[m]:
                m = j
            j += 1
        arr[i], arr[m] = arr[m], arr[i]
        i += 1
    return arr
a = [int(i) for i in input().split()]
print(*sort_sel(a))
```

## Листинг 6. Задание3.py

```
'''
Выполнил: Слепов Андрей
K8_2. Техника работы со списками
Задание 3. Array114. Дан массив A размера N. Упорядочить
его по возрастанию методом сортировки простыми вставками:
сравнить элементы A0 и A1 и, при необходимости меняя их
местами, добиться того, чтобы они оказались упорядоченными
по возрастанию; затем обратиться к элементу A2 и
переместить его в левую (уже упорядоченную) часть массива,
сохранив ее упорядоченность; повторить этот процесс для
остальных элементов, выводя содержимое массива после
обработки каждого элемента (от 1-го до N-1 го).
'''
```

```
def insertion(arr):
    for i in range(len(arr)):
        j = i - 1
        key = arr[i]
        while arr[j] > key and j >= 0:
```

```
        arr[j + 1] = arr[j]
        j -= 1
    arr[j + 1] = key
    return arr
```

```
a = [int(i) for i in input().split()]
print(*insertion(a))
```

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

### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_1. Техника работы с циклом for и генераторами списков
'''

import codecs

def find_name_command(find_command, group_command, lst):
    nc = 0
    index = 0
    find_command += '\r\n'
    for i in lst:
        #print(i)
        if (nc == 1 and enter_in_string(i)):
            break
        elif (nc == 1 and not enter_in_string(i)):
            print(i)
            group_command.append(i)
        if (find_command == i and nc == 0):
            group_command.append(i)
            nc += 1

def enter_in_string(string1):
    nc = 0
    for i in range(len(string1)):
        if (string1[i] == ":"):
            return False
    return True

with codecs.open("Конкурс проектов.txt", "r", "utf_8_sig") as file:
    lst = [i for i in file.readlines()]

students_and_position = [i for i in lst if not enter_in_string(i)]
possible_comm = [i for i in lst if enter_in_string(i)]

find_command = input()
group_command = []
```

```

find_name_command(find_command, group_command, lst)

print(students_and_position)
print(possible_comm)
print(group_command)

```

## Листинг 2. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 1. Array55. Дан целочисленный массив А размера N (<= 15).
Переписать в новый целочисленный
массив В все элементы с нечетными порядковыми номерами (1, 3, ...) и
вывести размер
полученного массива В и его содержимое. Условный оператор не
использовать.
'''

def main():
    a = list(map(int, input().split()))
    b = []
    for i in range(1, len(a), 2):
        b.append(int(a[i]))
    print("Размер массива: ", len(b))
    print(*b)

if __name__ == "__main__":
    main()

```

## Листинг 3. Задание2.py

```

'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 2. Array57. Дан целочисленный массив А размера N. Переписать в
новый целочисленный массив В
того же размера вначале все элементы исходного массива с четными
номерами,
а затем — с нечетными:
A[0], A[2], A[4], A[6], ..., A[1], A[3], A[5], ... .
Условный оператор не использовать.
'''

```

```

a = []
n = int(input())
for i in range(n):
    elem = int(input())
    a.append(elem)
b = []
for i in range(1, len(a), 2):
    b.append(a[i])

for i in range(0, len(a), 2):
    b.append(a[i])
print(*b)

if __name__ == "__main__":
    main()

```

#### Листинг 4. Задание3.py

```

'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 3. Array58. Дан массив A размера N. Сформировать новый массив
В того же размера по
следующему правилу: элемент В[К] равен сумме элементов массива А с
номерами от 0 до К.
'''

def fillingZero(lst, b, n):
    b.append(lst[0])
    for i in range(1, n):
        b.append(0)

def main():
    b = []
    lst = []
    summ = 0
    n = int(input())
    for i in range(n):

```

```

        elem = float(input())
        lst.append(elem)
    fillingZero(lst, b, n)
    #b[0] = lst[0]
    for k in range(1, n):
        for j in range(k+1):
            b[k] += lst[j]
    print(b)
if __name__ == "__main__":
    main()

```

### Листинг 5. Задание4.py

```

'''
Выполнил: Митюшин Пётр. П1-18
K9_2. Техника работы с циклом for и генераторами списков
Задание 4. Matrix3. Даны целые положительные числа М, N и набор из М
чисел. Сформировать
матрицу размера М x N, у которой в каждом столбце содержатся все числа
из исходного
набора (в том же порядке).
'''

n = int(input())
m = int(input())
mas = []
arr = []
for i in range(m):
    elem = int(input())
    arr.append(elem)

for i in range(m):
    mas.append([])
    for j in range(n):
        mas[i].append(arr[i])

print(mas)

```

### Листинг 6. Задание5.py

```

'''

```

Выполнил: Митюшин Пётр. П1-18

К9\_2. Техника работы с циклом for и генераторами списков

Задание 5. Matrix56. Дана матрица размера М х N (N – четное число).

Поменять местами

левую и правую половины матрицы.

```
'''

def swapMatrix(mas, a, b):
    for i in range(m):
        temp = mas[i][a]
        mas[i][a] = mas[i][b]
        mas[i][b] = temp

n = int(input())
m = int(input())
mas = []
for i in range(m):
    mas.append([])
    for j in range(n):
        elem = float(input())
        mas[i].append(elem)

for j in range(n//2):
    swapMatrix(mas, j, (n // 2) + j)
print(mas)
```

## Листинг 7. Задание6.py

```
'''

Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 6. Matrix88. Дана квадратная матрица порядка М. Обнулить
элементы матрицы,
лежащие ниже главной диагонали. Условный оператор не использовать.
'''

m = int(input())
mas = []
for i in range(m):
```



```

mas.append([])
for j in range(m):
    el = int(input())
    mas[i].append(el)

index_app = 0
for i in range(m):
    for j in range(m):
        if (j < index_app):
            #print(index_app)
            mas[i][j] = 0
    index_app += 1

for i in range(m):
    for j in range(m):
        print(mas[i][j], end=" ")
    print()

```

## 1.9. Техника работы с функциями

### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K10_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, b, c):
    if a > b:
        return 0
    elif (a < b):
        summ = 0
        for i in range(a, b):
            summ += i
        for i in range(b, c+1):
            summ += i
        return summ

a = int(input())
b = int(input())
c = int(input())
print(SumRange(a, b, c))
```

### Листинг 2. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K10_1. Техника работы с функциями
Задание 3. Func10. Описать функцию IsSquare(K) логического типа,
возвращающую True,
если целый параметр K (> 0) является квадратом некоторого целого
числа, и False
в противном случае. С ее помощью найти количество квадратов в наборе
из 10 целых
положительных чисел.
```

```
'''
import math

def IsSquare(K):
    result = math.sqrt(K)
    return (result % 1 == 0)

array = []
for i in range(11):
    elem = int(input())
    array.append(elem)
for i in range(11):
    print(IsSquare(array[i]))
```

### Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K10_1. Техника работы с функциями
Задание 4. Func33. Описать функцию SortInc3(X), меняющую содержимое
списка X из трех
вещественных элементов таким образом, чтобы их значения оказались
упорядоченными по
возрастанию (функция возвращает None). С помощью этой функции
упорядочить по
возрастанию два данных списка X и Y.
'''

def filling(X):
    for i in range(3):
        elem = int(input())
        X.append(elem)

def SortInc3(X):
    for i in range(3):
        for j in range(3):
            if (X[i] < X[j]):
                temp = X[i]
                X[i] = X[j]
                X[j] = temp
```

```

        return X
X = []
Y = []
filling(X)
filling(Y)
print(SortInc3(X))
print(SortInc3(Y))

```

#### Листинг 4. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
K10_2. Техника работы с функциями
Задание 2. Использовать map, lambda
Квадраты в обратном порядке. Числа вводятся до точки. Через пробел
выведите эти числа в
обратном порядке, возводя их в квадрат.
Sample Input:
'''

lst = [i for i in map(int, iter(input, "."))]
p = list(map(lambda i: print(i ** 2, end=" "), lst[::-1]))

```

#### Листинг 5. Задание2.py

```

'''
Выполнил: Митюшин Пётр. П1-18
K10_2. Техника работы с функциями
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.
'''

n = int(input())
lst = [int(input()) for i in range(n)]
b = list(filter(lambda x: x % 2 != 0, lst))
print(b)

```

## Листинг 6. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K10_2. Техника работы с функциями
Задание 4. Использовать lambda, map.
Быстрая инициализация. Программа получает на вход три числа через
пробел — начало и конец
диапазона, а также степень, в которую нужно возвести каждое число из
диапазона. Выведите
числа получившегося списка через пробел.
'''
lst = list(map(int, input().split()))
p = list(map(lambda i: print(i ** lst[2], end=" "), range(lst[0],
lst[1]+1)))
print(*p)
```

## 1.10. Техника работы со словарями

### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K11_1. Техника работы со словарями
Задание 1. Задача «Номер появления слова»
Условие. В единственной строке записан текст. Для каждого слова из
данного текста
подсчитайте, сколько раз оно встречалось в этом тексте ранее.
Словом считается последовательность непробельных символов идущих
подряд, слова разделены
одним или большим числом пробелов или символами конца строки.
'''
counter = {}
lst = input().split()
for i in lst:
    if i not in counter:
        counter[i] = 0
    else:
```

```

        counter[i] += 1
    print(counter[i], end=" ")

```

## Листинг 2. Задание2.py

```

'''
Выполнил: Слепов Андрей. П1-18
K11_1. Техника работы со словарями
Задание 2.Задача «Права доступа»
Условие. В файловую систему одного суперкомпьютера проник вирус,
который сломал контроль
за правами доступа к файлам. Для каждого файла известно, с какими
действиями можно к
нему обращаться:
запись W,
чтение R,
запуск X.
В первой строке содержится число N – количество файлов содержащихся в
данной файловой
системе. В следующих N строчках содержатся имена файлов и допустимых с
ними операций,
разделенные пробелами. Далее указано число M – количество запросов к
файлам. В последних
M строках указан запрос вида Операция Файл. К одному и тому же файлу
может быть применено
любое количество запросов.
Вам требуется восстановить контроль над правами доступа к файлам (ваша
программа для
каждого запроса должна будет возвращать ОК если над файлом выполняется
допустимая
операция, или же Access denied, если операция недопустима.
'''

permission = {}
n = int(input())
for i in range(n):
    s = input().split()
    permission[s[0]] = s[1:]

```

```

for i in range(int(input())):
    perm, file = input().split()
    if perm == 'read':
        perm = 'R'
    if perm == 'write':
        perm = 'W'
    if perm == 'execute':
        perm = 'X'
    if perm in permission[file]:
        print('OK')
    else:
        print('Access denied')

```

### Листинг 3. Задание3.py

```

'''
Выполнил: Слепов Андрей. П1-18
K11_1. Техника работы со словарями
Задание 3. Задача «Самое частое слово»
Условие. Дан текст: в первой строке задано число строк, далее идут
сами строки.
Выведите слово, которое в этом тексте встречается чаще всего. Если
таких слов несколько,
выведите то, которое меньше в лексикографическом порядке.
'''

dictt = {}
max = 0

for i in range(int(input())):
    for word in input().split():
        if word in dictt:
            dictt[word] += 1
        else:
            dictt[word] = 1
        if dictt[word] > max:
            max = dictt[word]

for key, value in sorted(dictt.items()):
    if value == max:

```

```
print(key)
break
```

#### Листинг 4. Задание1.py

```
'''
Выполнил: Слепов Андрей. П1-18
K11_2. Техника работы со словарями
Задание 1. Телефонная книга. Этап 1. Коля устал запоминать телефонные
номера и заказал у Вас
программу, которая заменила бы ему телефонную книгу. Коля может
послать программе
два вида запросов: строку, содержащую имя контакта и его номер,
разделенные пробелом,
или просто имя контакта. В первом случае программа должна добавить в
книгу новый номер,
во втором - вывести номер контакта. Ввод происходит до символа точки.
Если введенное
имя уже содержится в списке контактов, необходимо перезаписать номер.
'''

phone_num = {}
for string in i(input, "."):
    for i in range(len(string)):
        if string[i] == " ":
            name = string[0:i]
            number = string[i+1:len(string)]
            phone_num[name] = number
    if (string.isalpha()):
        print(phone_num[string])
```

#### Листинг 5. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K11_2. Техника работы со словарями
Задание 2. Телефонная книга. Этап 2. Коля понял, что у многих из
его знакомых есть несколько
телефонных номеров и нельзя хранить только один из них. Он попросил
доработать Вашу
программу так, чтобы можно было добавлять к существующему контакту
новый номер или даже
```



несколько номеров, которые передаются через запятую. По запросу телефонного номера должен выводиться весь список номеров в порядке добавления, номера должны разделяться запятой. Если у контакта нет телефонных номеров, должна выводиться строка "Не найдено".

```
'''
data = input()
phone_book = {}
while data != '.':
    data = data.replace(',','').split()
    if len(data) == 1:
        name = ''.join(data)
        if name in phone_book:
            print(', '.join(phone_book[name]))
        else:
            print("Не найдено" )
    else:
        name, number = data[0],data[1:]
        phone_book[name] = phone_book.get(name,[]) + number
    data = input()
```

## Листинг 6. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
K11_2. Техника работы со словарями
Задание 3. Телефонная книга. Этап 3. Коле очень понравилась Ваша
программа, однако он стал
замечать, что иногда в его телефонную книгу попадают номера в
некорректном формате.
Чтобы не сохранять недействительные номера, он попросил Вас
обрабатывать только номера,
соответствующие критериям:
- номер должен начинаться либо с +7, либо с 8 и состоять из 11 цифр.
- блоки цифр могут разделяться пробелами или дефисами.
- вторая, третья и четвертая цифры могут помещаться в скобки.
Если программа встречает некорректный номер, она должна его
проигнорировать. В обратном
```

случае она должна привести номер к виду +7 (900) 800-70-60 и запомнить. Остальной

функционал программы остается без изменений.

```
'''
```

```
def check_number(n):
```

```
    symbols = '+- ()1234567890' # То что должно содержаться в номере
    телефона
```

```
    count_sumb = 0
```

```
    if n.startswith('+7') or n.startswith('8'): # Начинается ли строка
    с указанного префикса нужно для того что бы
```

```
        for counter in n:
```

```
            if counter not in symbols: # Проверка на то что не
    содержит ли строка символы которые не должны присутствовать в номерах
    телефона
```

```
                return False
```

```
            if counter in '1234567890':
```

```
                count_sumb += 1 # Кол-во цифр в номере
```

```
            if count_sumb != 11: # Если длина номера телефона больше или
    меньше 11
```

```
                return False
```

```
            else:
```

```
                return True
```

```
        else:
```

```
            return False
```

```
def mod_number(n):
```

```
    n = n.replace('+', '').replace('(', '').replace(')', '').replace(' ',
    ',').replace('-', '') # Все символы +, (, ), ' ', -, заменяются на
    на пустую строку
```

```
    number = '+7 (' + n[1:4] + ') ' + n[4:7] + '-' + n[7:9] + '-' +
    n[9:] # Формируем правильный номер телефона
```

```
    return number
```

```
phonebook = {}
```

```
for counter in iter(input, '.'): # Считываем до поступления на вход
    точки
```

```
    if ' ' in counter: # Нужно для того, чтобы, если введут только имя
    то его номер надо вывести
```

```

        if counter.split()[0] not in phonebook: # Имя которое надо
записать в телефонную книгу
            phonebook[counter.split()[0]] = [] # Добавляем к ключу
которое считается за имя и по этому ключу добавляем словарь
            phones = counter.split(maxsplit = 1)[1]
            for phone in phones.split(', '): # На случай, если несколько
номеров телефонов и номера телефонов введены через запятую
                if check_number(phone): # Проверка на корректный номер
телефона

phonebook[counter.split()[0]].append(mod_number(phone)) # Добавляем
телефонный номер в добавленный ранее список
    else:
        if counter.split()[0] not in phonebook or
phonebook[counter.split()[0]] == []: # Проверка на то есть ли в строке
номер телефона
            print('Не найдено')
        else:
            print(*phonebook[counter.split()[0]], sep = ', ') # Вывод
номера телефона

```

## 1.11. Техника работы с множествами

### Листинг 1. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Задача «Количество различных чисел»
Условие. Дан список чисел. Определите, сколько в нем встречается
различных чисел.
'''
s = set(input().split())
print(len(s))

```

### Листинг 2. Задание2.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Задача «Количество совпадающих чисел»
Условие. Даны два списка чисел. Посчитайте, сколько чисел содержится
одновременно как
в первом списке, так и во втором.
'''

```

```
'''
n = set(input().split())
m = set(input().split())
p = 0
for elem in n:
    for j in m:
        if elem is j:
            p += 1
print(p)
```

### Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Задача «Пересечение множеств»
Условие. Даны два списка чисел. Найдите все числа, которые входят как
в первый,
так и во второй список и выведите их в порядке возрастания.
'''

n = set(map(int, input().split()))
m = set(map(int, input().split()))
lst = []
fin = set()
for i in n:
    for j in m:
        if (i == j):
            lst.append(i)
print(*sorted(lst))
```

### Листинг 4. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4.
Задача «Количество слов в тексте»
Условие. Дан текст: в первой строке записано число строк, далее идут
сами строки.
Определите, сколько различных слов содержится в этом тексте.
Словом считается последовательность непробельных символов идущих
подряд, слова разделены
```

одним или большим числом пробелов или символами конца строки.

```
'''
def fillSetWords(string, set_words):
    start_index = 0
    for i in range(len(string)):
        if (string[i] == ' '):
            end_index = i
            set_words.add(string[start_index:end_index])
            start_index = i + 1

set_words = set()
n = int(input())
for i in range(n):
    string = input().split()
    for i in range(len(string)):
        set_words.add(string[i])
print((len(set_words)))
```

## Листинг 5. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. https://pythontutor.ru/lessons/sets/problems/polyglotes/
Задача «Полиглоты»
Условие. Каждый из некоторого множества школьников некоторой школы
знает некоторое
количество языков. Нужно определить сколько языков знают все
школьники, и сколько языков
знает хотя бы один из школьников.
В первой строке задано количество школьников. Для каждого из
школьников сперва записано
количество языков, которое он знает, а затем - названия языков, по
одному в строке.
В первой строке выведите количество языков, которые знают все
школьники. Начиная со
второй строки - список таких языков. Затем - количество языков,
которые знает хотя бы
один школьник, на следующих строках - список таких языков. Языки нужно
выводить в
```

лексикографическом порядке, по одному на строке.

```
'''
languages = set()
n = int(input())
nc = 0
knows_all_length = True
for i in range(n):
    m = int(input())
    if (m == 1 and nc < 1):
        string = input()
        print(1)
        knows_all_length = False
        nc += 1
        print(string)
    else:
        for c_length in range(m):
            string = input()
            languages.add(string)

if knows_all_length:
    print(len(languages))
    languages = sorted(languages)
    for i in languages:
        print(i)
print(len(languages))
languages = sorted(languages)
for i in languages:
    print(i)
```

## Листинг 6. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
Простейшая система проверки орфографии может быть основана на
использовании списка известных слов.
Если введённое слово не найдено в этом списке, оно помечается как
"ошибка".
Попробуем написать подобную систему.
```

На вход программе первой строкой передаётся количество *d* известных нам слов, после чего

на *d* строках указываются эти слова.

Затем передаётся количество *l* строк текста для проверки, после чего *l* строк текста.

Выведите уникальные "ошибки" в произвольном порядке. Работу производите без учёта регистра.

```
'''
lst_words = set()
lst_string = set()

m = int(input())
for i in range(m):
    lst_words.update(input().lower().split())

n = int(input())
for i in range(n):
    lst_string.update(input().lower().split())

lst_string.difference_update(lst_words)
for i in lst_string:
    print(i)
```

## Листинг 7. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. (Л.Б.) Сессия
В файле disc.txt хранится перечень дисциплин, выносимых на сессию,
например,
Теория алгоритмов
МДК.01.01
Основы экономики
...
В файле session.txt хранятся сведения о результатах сессии, например,
Грушников; П2-18; Теория алгоритмов; 5
Константинович; П2-18; Теория алгоритмов; 5
...'''
```

Студент считается сдавшим сессию, если у него сданы все предметы и нет оценки "2".

Студент считается "отличником", если у него все пятерки

Требуется сформировать множества:

- студентов, сдавших сессию
- студентов-отличников
- дисциплин, по которым нет задолженностей

Результат вывести в файл output.txt

```
'''
```

```
import codecs
```

```
f_input1 = codecs.open("session.txt", "r", "utf_8_sig")
```

```
f_input2 = codecs.open("disc.txt", "r", "utf_8_sig")
```

```
lst = f_input1.readlines() #Из файла session.txt читаем и добавляем в список lst
```

```
session = [i.strip().split('; ') for i in lst] #Удаляем ';' ' из list
```

```
lst = f_input2.readlines()
```

```
disc = [i.strip() for i in lst] #Формирует список строк и благодаря функции strip с обоих концов которой устранены указанные символы.
```

```
Формируется список из предметов
```

```
f_input1.close()
```

```
f_input2.close()
```

```
names_students = []
```

```
for data_students in session:
```

```
    if data_students[0] not in names_students:
```

```
        names_students.append(data_students[0]) #Добавляем имя с
```

```
список имён
```

```
excellent = set() #Отличники
```

```
sdali = set() #Множество сдавших сессию
```

```
predmet = {*disc} #Название предмета
```

```
for i in range(len(names_students)):
```

```
    count_5 = 0 #Кол-во отличников. Так же каждая прокрутка цикла
```

```
обнуляет этот счётчик и для другого студента начинает считать заново
```

```
    count_2 = 0 #Кол-во двоек нужно для того чтобы узнать кто не
```



```

закрыл ссесию
    for j in range(len(session)):
        if(session[j][0] == names_students[i]): # Поиск имени студента
            if(session[j][3] == '5'):
                count_5 += 1
            elif(session[j][3] == '2'):
                count_2 += 1
            predmet.discard(session[j][2]) # Удаляет указанный
элемент из множества
        if(count_5 == len(disc)):
            excellent.add(names_students[i]) #Добавляет фамилии отличников
        if(count_2 == 0):
            sdali.add(names_students[i]) # Усли кол-во двоек равно нулю то
студент сдал ссесию (мне бы так)
output_file = open("out.txt", "w")
output_file.write('Сдали:\n'+'\n'.join(names_students)) #Имена тех кто
сдали
output_file.write('\n=====
=====\\n')
output_file.write('Отличники:\n'+'\n'.join(excellent)) #Имена
отличников
output_file.write('\n=====
=====\\n')
output_file.write('Предметы по которым нет
задолженности:\n'+'\n'.join(predmet)) #Предметы по которым нет
задолженностей
output_file.close()

```

## 1.12. Техника работы с кортежами

### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
Вывести чётные
Необходимо вывести все четные числа на отрезке [a; a * 10].
'''
n = int(input())
k = (n + 1) // 2 * 2
print(tuple(range(k, n * 10 + 1, 2)))
```

### Листинг 2. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2.
Убывающий ряд.
С клавиатуры вводятся целые числа  $a > b$ . Выведите убывающую
последовательность чисел
по одному числу в строке.
'''
n = int(input())
m = int(input())
for i in tuple(range(n, m, -1)):
    print(i)
```

### Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. (Л.Б.) В каждой строке файла хранится информация о пунктах
и их координатах
относительно некоторого центра.
Требуется
1. Прочсть файл в список кортежей
2. Найти диаметр множества точек, то есть расстояние между наиболее
удалёнными точками.
Указать наиболее удалённые пары
```

3. Сформировать список пар городов, имеющих одинаковое расстояние до центра

4. Отсортировать список одним из методов, реализованных в предыдущих работах

Результаты вывести на экран

'''

```
import codecs
```

```
def BubbleSort(vals):
```

```
    for i in range(len(vals)):
```

```
        for j in range(i+1, len(vals)):
```

```
            if (vals[i] < vals[j]):
```

```
                temp = vals[i]
```

```
                vals[i] = vals[j]
```

```
                vals[j] = temp
```

```
    return vals
```

```
fin = codecs.open("Zadamie 3.txt", 'r', "utf_8_sig")
```

```
d_coord = dict()
```

```
d_hyp = dict()
```

```
print()
```

```
for string in fin:
```

```
    l = string.split()
```

```
    d_coord[l[0]] = tuple(l[1:])
```

```
    d_hyp[l[0]] = (int(d_coord[l[0]][0])**2 + int(d_coord[l[0]][1])**2) ** 0.5
```

```
fin.close()
```

```
vals_sort = BubbleSort(list(d_hyp.values()))
```

```
numb_city = len(d_coord)
```

```
for i in range(numb_city):
```

```
    for city in d_hyp:
```

```
        if d_hyp[city] == vals_sort[i]:
```

```
            print(f"{i+1} |", end = ' ')
```

```
            print(city + ":", *d_coord[city], end = '\n    ')
```

```
            print(f"До центра: {int(vals_sort[i])} км\n")
```

```

        break

count = 0
for city_i in d_hyp:
    for city_j in d_hyp:
        if d_hyp[city_i] == d_hyp[city_j] and city_i != city_j:
            print(f"{d_hyp[city_i]} == {d_hyp[city_j]}")
            count += 1
if count == 0:
    print("Пар городов, имеющих одинаковое расстояние до центра нет")

```

#### Листинг 4. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Класс namedtuple() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-python/klass-namedtuple-modulja-collections/
'''

from collections import namedtuple

print("Пример 1\n")
Point = namedtuple('Point', ['x', 'y'])

x, y = int(input()), int(input())
p = Point(x, y)
print(p[0] - p[1])
print(p[0] * p[1])
print(p[0] / p[1])
print(p)

print("Пример 2\n")
lst = [1, 2]
p = Point._make(lst)
print(p)

```

```

print("Primer 3\n")
Home = namedtuple('home', ['color', 'width', 'height'])
p = Home("#ffff0000", "50", "150")
print(p.color)

print("Primer 4\n")
Persona = namedtuple('Person', 'name age group')
Peter = Persona(name='Peter', age='18', group='P1-18')
Andrey = Persona(name='Andrey', age='18', group='P1-18')
for i in [Peter, Andrey]:
    print(i)

print(Peter._replace(name="icefantik"))

print("Primer 5\n")
Point3D = namedtuple('Point3D', Point._fields + ('z',))
print(Point3D(10, 20, 30))

Point2D = namedtuple('Point2D', Point._fields)
print(Point2D(10, 20))

print("Primer 6\n")
Group = namedtuple("Group", ['name', 'age', 'group'])
for i in range(1):
    s_name = input()
    s_age = input()
    s_group = input()
    Group.__doc__ += str(i)
    Group.name.__doc__ = s_name
    Group.age.__doc__ = s_age
    Group.group.__doc__ = s_group

print("Primer 7")
lst = []
Group2 = namedtuple('Group2', ['name', 'age', 'group'])
for i in range(1, 2):
    s_name = input()
    s_age = input()

```

```

s_group = input()
lst.append(Group2(name=s_name, age=s_age, group=s_group))

for i in lst:
    print("Name: {0}, age: {1}, group: {2}".format(i.name, i.age,
i.group))

```

### 1.13. Техника работы с файлами

#### Листинг 1. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
Text5. Дана строка S и текстовый файл. Добавить строку S в конец
файла.
'''

string = input()
with open("file.txt", "a") as file:
    file.write(string)

```

## Листинг 2. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2.
Text12. Дана строка S и текстовый файл. Заменить в файле все пустые
строки на строку S.
'''

string = input()
with open("Task2.txt", "r") as read_file:
    with open("Task2_1.txt", "w") as write_file:
        for i in read_file.readlines():
            print(i)
            if i != "\n":
                write_file.write(i)
            else:
                write_file.write(string+"\r")
```

## Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3.
Text20. Дан текстовый файл. Заменить в нем все подряд идущие пробелы
на один пробел.
'''

with open("Text20.txt", "r") as f_read:
    with open("Text20_1.txt", "w") as f_write:
        nc = 0
        final_string = ""
        for string in f_read.readlines():
            nc = 0
            for j in range(len(string)):
                if (string[j] == " " and nc == 0):
                    nc += 1
                    final_string += string[j]
                elif (string[j] != " "):
                    nc = 0
                    final_string += string[j]
            #elif (nc > 1):
```

```
f_write.write(final_string)
print(final_string)
```

#### Листинг 4. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4.
Text44. Дан текстовый файл, каждая строка которого изображает целое
число, дополненное слева и справа
несколькими пробелами. Вывести количество этих чисел и их сумму.
'''

lst = []
summ = 0
count = 0
formation = ""
amount_of_num = 0
with open("Text 44.txt", "r") as f_read:
    #lst = [i for i in f_read.readlines()]
    for i in f_read.read():
        if (i == ' ' or i == "\n"):
            if formation != '':
                lst.append(int(formation))
                formation = ""
            continue
        elif ('0' <= i and i <= '9' or i == '-'):
            formation += i
            count += 1

print(len(lst))
print(sum(lst))
```

#### Листинг 5. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5.
Text53. Дан текстовый файл. Создать символьный файл, содержащий все
знаки препинания,
```



```

встретившиеся в текстовом файле (в том же порядке).
'''
punct_marks = []
with open("Text 53.txt", "r") as file:
    for i in file.read():
        if i == '.' or i == ',' or i == ';' or i == '!' or i == '?' or
i == '-':

            punct_marks.append(i)
        elif i == '(' or i == ')' or i == '[' or i == ']' or i == '{'
or i == '}':

            punct_marks.append(i)
        elif i == '"' or i == '<<' or i == '>>' or i == ':' or i ==
'/' or i == "'":

            punct_marks.append(i)

with open("TextPunct.txt", "w") as Punctfile:
    for i in punct_marks:
        Punctfile.write(i)

```

## 1.14. Техника работы с модулями

### Листинг 1. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Контейнерные типы данных модуля collections.
https://docs-python.ru/standart-library/modul-collections-python/
Класс deque() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-
python/klass-deque-modulja-collections/
'''

import collections

lst = []
length = 0
with open("text.txt", "r") as file:

```

```

        for i in file.readline():
            lst.append(i)
            length += 1

dq = collections.deque(lst)

dq.reverse()
print(dq)

with open("text.txt", "w") as file:
    for i in range(length):
        file.write(dq[i])

n = 3
array = [40, 35, 30, 25, 20]
d = collections.deque(array)
summ_d = sum(d)
for i in array:
    summ_d += i - d.popleft()
    d.append(i)
print(summ_d / n)
to_check = input()
item_found = False
iterables = ['ABC', 'D', 'EF']
iterators = collections.deque(iterables)
for item in iterators:
    if to_check == item:
        print("Item found")
        item_found = True
if not item_found:
    print("Item not found")

```

## Листинг 2. Задание2.py

'''

Выполнил: Митюшин Пётр. П1-18

Задание 2. Контейнерные типы данных модуля collections.

Класс Counter() модуля collections в Python.

```

https://docs-python.ru/standart-library/modul-collections-  
python/klass-counter-modulja-collections/
'''
import collections
import re

cnt = collections.Counter(a=4, b=2, c=0, d=-2)
print(cnt)
print(*cnt.elements())

ct = collections.Counter("abbbaaaccacccascd")
s = set(ct)
print(ct.most_common(len(s)))

cnt1 = collections.Counter(a=3, b=6, c=6, d=5)
cnt1.subtract(cnt) # вычитает элементы текущего счетчика
print(cnt1)

cnt1.update(cnt) # складывает элементы текущего счетчика
print(cnt1)

print(cnt + cnt1) #Сложить два счетчика
print(cnt - cnt1) #Вычитание счетчиков
print(cnt & cnt1) #Пересечение счетчиков
print(cnt | cnt1) #Объединение счётчиков

print(cnt.items())
print(cnt.values())
cnt.clear()

string = ""
c = collections.Counter()
with open("text1.txt", "r") as file:
    for i in file:
        string += i

c = collections.Counter(string).most_common(len(string))
print(c)

```

```

cn = collections.Counter()
with open("text1.txt", "r") as file:
    for i in file:
        words = re.findall(r'\w+', file.read())    #findall
используется для поиска всех непересекающихся совпадений в шаблоне

cn = collections.Counter(words).most_common(len(words))
print(cn)

```

### Листинг 3. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Контейнерные типы данных модуля collections.
https://docs-python.ru/standart-library/modul-collections-python/
Класс defaultdict() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-python/klass-defaultdict-modulja-collections/
'''

from collections import defaultdict
import random

print("1.-----")
lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009),
      ("Alice", 404-502), ("Nick", 16507811251),
      ("Robert", 51234047129), ("Alice", 894-455), ("Alice", 439-
495)]
d = defaultdict(list)
for i, elem in lst:
    d[i].append(elem)

print(d.items())

print("2.-----")
lst_number = [('a', 1), ('b', 2), ('c', 3), ('d', 4) , ('e', 5) ,
              ('f', 6), ('a', 2), ('a', 3)]
d_slov = {}
for i, elem in lst_number:

```

```

    d_slov.setdefault(i, []).append(elem**2)

print(sorted(d_slov.items()))

print("3.-----")
l = {}
for i, elem in lst:
    n = random.randint(1, 100)
    l.setdefault(i, []).append(n)

print(l)

print("4.-----")
string = "Hello world"
d = defaultdict(int)
for k in string:
    d[k] += 1

print(d)

print("5.-----")
lst = ["qwertyui", "asdfghjkl", "zxcvbnm", "q"]
d = defaultdict(int)
for i in lst:
    d[i] += len(i)
print(d.items())

print("6.-----")
lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009),
("Alice", 404-502), ("Nick", 16507811251),
      ("Robert", 51234047129), ("Alice", 894-455), ("Alice", 439-
495)]
d = defaultdict(set)
index = 0
for i, elem in lst:
    if (index % 2 == 0):
        d[i].add(elem)
    index += 1

```

```

print(d.items())

print("7.-----")
for elem in lst:
    print(elem)

print("8.-----")
for i, elem in lst:
    print(i, elem)

```

#### Листинг 4. Задание2.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Контейнерные типы данных модуля collections.
Класс OrderedDict() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-python/klass-orderreddict-modulja-collections/
'''

from collections import *

c = Counter()
items = []
n = int(input())
for i in range(n):
    string = input()
    items.append(string)

for i in items:
    c[i] += 1
print(c)

defdict = defaultdict(list)
for i in range(n):
    defdict[i].append(i)

print(defdict)

d = OrderedDict.fromkeys('abcd')

```

```

d.move_to_end('b') # добавляет элемент из строки в конец
print(''.join(d.keys())) #print(d.keys())

d.move_to_end('a')
print(''.join(d.keys()))

d.popitem('a') # удаляем элемент
d.popitem('b')
print(''.join(d.keys()))

d.move_to_end('d', last=True) #переносим d в перед
print(''.join(d.keys()))
d.move_to_end('d', last=False) #переносим d в перед
print(''.join(d.keys()))

```

## Листинг 5. Задание1.py

```

'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Функция argv модуля sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/funktsija-argv-modulja-sys/
'''
import sys

print(len(sys.argv))
print(sys.argv[1])

for i in range(len(sys.argv)):
    print(sys.argv[i], end=" ")

w = None
print(len(sys.argv))
if len(sys.argv) > 1:
    if sys.argv[1] in ("-h", "-help"):
        print("Manual")
        word = 0
    else:
        word = sys.argv[0]

```

### Листинг 6. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Имя используемой OS.
https://docs-python.ru/standart-library/modul-sys-python/imja-
ispolzuemoj-os/'''
import sys
import os

if sys.platform.startswith('linux'):
    print("This is linux {0}".format(os.name))
else:
    print("This is not linux. This is {0}".format(os.name))
    print(sys.getwindowsversion())
```

### Листинг 7. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Различные сведения о версии Python.
https://docs-python.ru/standart-library/modul-sys-python/razlichnye-
svedeniya-versii/'''
import sys

string = sys.version_info
print(string)
print(sys.copyright)
print("API C languages: ", sys.api_version)
print(sys.version)
print("Hex version: ", sys.hexversion)
```

### Листинг 8. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4. Каталоги и пути интерпретатора Python.
https://docs-python.ru/standart-library/modul-sys-python/katalogi-
puti-interpretatora/
```



```
'''
import sys
import os

print(sys.prefix)
print(sys.base_prefix)
print(sys.exec_prefix)
print(sys.base_exec_prefix)
print(sys.executable)
```

### Листинг 9. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. Объекты stdin, stdout, stderr модуля sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/obekty-stdin-stdout-stderr-modulja-sys/
'''

import sys
import time

stdin = sys.stdin
try:
    sys.stdin = open("text_zadania5.txt", "r")
    s = input()
    print("odna stroka: ", s)
finally:
    sys.stdin.close()
    sys.stdin = stdin

try:
    sys.stdin = open("text_zadania5.txt", "r")
    for i in sys.stdin:#s = input()
        print(i, end="")
finally:
    sys.stdin.close()
    sys.stdin = stdin
```

### Листинг 10. Задание6.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 6. Функция exit() модуля sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/funktsija-exit-modulja-sys/
'''
```

```
import sys

if len(sys.argv) > 1:
    if sys.argv[1] in ("-exit", "-e"):
        sys.exit(0)
```

### Листинг 11. Задание1.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 1. Вывод текущей директории
'''

import os
print("Текущая деректория:", os.getcwd())
```

### Листинг 12. Задание2.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 2. Создание папки
'''

if not os.path.isdir("Митюшин и Слепов"): # Проверка на отсутствие
#папки с таким названием
    os.mkdir("Митюшин и Слепов")
```

### Листинг 13. Задание3.py

```
'''
Выполнил: Слепов Андрей. П1-18

Задание 3. Изменение директории
'''
```

```

import os

os.chdir("../")
if not os.path.isdir("Новая папка"): # Проверка на отсутствие папки с
таким названием
    os.mkdir("Новая папка")

os.chdir("Новая папка")

print("Текущая директория изменилась на :", os.getcwd())

```

#### Листинг 14. Задание4.py

```

'''
Выполнил: Слепов Андрей. П1-18
Задание 4. Создание вложенных папок
'''

import os

os.chdir("..")
path_dir = os.getcwd()
n = int(input("Введите количество вложенных папок, которые вы хотите
создать: "))
for i in range(n):
    path = path_dir + "/Папка " + str(i)
    os.mkdir(path)
    path_dir += "/Папка " + str(i)

```

#### Листинг 15. Задание5.py

```

'''
Выполнил: Слепов Андрей. П1-18
Задание 5. Создание файлов
'''

file = open("MyFirstFile.txt", "w")
file.write("Файл – именованная область данных на носителе
информации.")
file.close()

```

### Листинг 16. Задание6.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 6. Переименование файлов
'''

import os

path = input("Введите путь, где будет создан этот файл: ")
First_name = input("Введите имя файла: ")

os.chdir(path)

file = open(First_name + ".txt", "w")
file.close()

Second_name = input("Как Вы хотите переименовать файл: ")
os.rename(First_name + ".txt", Second_name)
```

### Листинг 17. Задание7.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 7. Перемещение файлов
'''

import os

f = open("First_name.txt", "w")
f.close()

os.replace("First_name.txt", "../First_name.txt")
```

### Листинг 18. Задание8.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 8. Список файлов и директорий
'''

import os
```

```

# распечатать все файлы и папки
for dirpath, dirnames, filenames in os.walk("."):
    # перебрать каталоги
    for dirname in dirnames:
        print("Каталог:", os.path.join(dirpath, dirname))
    # перебрать файлы
    for filename in filenames:
        print("Файл:", os.path.join(dirpath, filename))

# os.walk() — это генератор дерева каталогов.
# Он будет перебирать все переданные составляющие.
# Здесь в качестве аргумента передано значение «.», которое обозначает
верхушку дерева.
# Метод os.path.join() был использован для объединения текущего пути с
именем файла/папки.

```

### Листинг 19. Задание9.py

```

'''
Выполнил: Слепов Андрей. П1-18
Задание 9. Удаление файлов
'''

import os

File_name = input("Введите имя файла: ")
f = open(File_name + ".txt", "w")
f.close()
os.remove(File_name + ".txt")

```

### Листинг 20. Задание10.py

```

'''
Выполнил: Слепов Андрей. П1-18
Задание 10. Удаление директорий
'''

import os

print("Все папки и файлы: ", os.listdir())

```

```
del_name = input("Введите имя папки которую Вы хотите удалить: ")
os.rmdir(del_name)
```

## Листинг 21. Задание11.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 11. Получение информации о файлах
'''

import os

f = open("text.txt", "w")
f.write('Hello, World!')
f.close()

print(os.stat("text.txt"))

os.remove("text.txt")
```

## 1.15. Техника работы с классами

### Листинг 1. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Создание класса
'''

class ColorSet:
    black = "#000000"
```

### Листинг 2. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Создание объекта
'''

class ColorSet:
    black = "#000000"
    white = "#ffffff"

cl = ColorSet()
print(ColorSet.black)
```

### Листинг 3. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Функция init
'''

class VisualElements:
    def __init__(self, color, color_background):
        self.color = color
        self.color_background = color_background
    black = "#000000"
    white = "#ffffff"

unit = VisualElements("#000000", color_background = "#ffffff")
print(unit.color)
print(unit.color_background)
```

## Листинг 4. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4. Методы объектов
'''

class CreateCharacter:
    max_speed = 100
    def __init__(self, name, race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
Player.print_haracteristic()
```

## Листинг 5. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. Параметр self
'''

class Enemy:
    def __init__(self, name, enemy_health = 30, damage = 5):
        self.name = "enemy"
        self.enemy_health = enemy_health
        self.damage = damage
    def hit(self, damage): self.enemy_health -= damage
```



```

class CreateCharacter:
    max_speed = 100
    def __init__(self, name, race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def hit(self, damage): self.health -= damage

Player = CreateCharacter("Pers", 'Human', 100)
enemy = Enemy("eewre")
Player.hit(enemy.damage)
enemy.hit(Player.damage)
print(Player.health)
print(enemy.enemy_health)

```

## Листинг 6. Задание6.py

'''

Выполнил: Митюшин Пётр. П1-18

Задание 6. Изменение свойств объекта

'''

```

class CreateCharacter:
    max_speed = 100
    def __init__(self, name, race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_characteristic(self):
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
Player.print_characteristic()
Player.name = "Peter"

```

```
print("-----")
Player.print_haracteristic()
```

### Листинг 7. Задание7.py

```
'''
Задание 7. Удалить свойства объекта
'''

class CreateCharacter:
    max_speed = 100
    def __init__(self, name, race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
del Player.armor
try:
    Player.print_haracteristic()
except AttributeError:
    print("Attribute was removed")
```

### Листинг 8. Задание8.py

```
'''
Задание 8. Удаление объектов
'''

class CreateCharacter:
    max_speed = 100
    def __init__(self, name, race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
```

```

        self.armor = armor
    def print_haracteristic(self):
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
del Player
try:
    Player.print_haracteristic()
except NameError:
    print("Object player was removed")

```

## Листинг 9. Задание1.py

```

'''
K16_1. Техника работы с классами.
Задание 1. Создание класса
Задание 2. Создание объекта
Задание 3. Функция init
Задание 4. Методы объектов
Задание 5. Параметр self
Задание 6. Изменение свойств объекта
Задание 7. Удалить свойства объекта
Задание 8. Удаление объектов
'''

#Задание 1
print("Задание 1")
class Class_Wolf:
    count_wolf = 0
    def __init__(self, age):
        self.name = "Wolf"
        self.color = "grey"
        self.age = age
        self.rem_atr = "Нужно удалить"
        Class_Wolf.count_wolf += 1
    def holw(): print("W0000000000000000000!!!")
    def how_much_wolf(self):
        print("Боет {0} волк(ов)".format(Class_Wolf.count_wolf))

```

```

#Задание 2
wolf = Class_Wolf(10)

#Задание 3
print("\nЗадание 3")
Class_Wolf.holw()
wolf.how_much_wolf()
print("Вывод сколько воют волков не через функцию класса %d" %
Class_Wolf.count_wolf)

print(hastate(wolf, "rampart"))
dilator(wolf, "rampart")
print(hastate(wolf, "rampart"))

atr = input("Введите имя атрибута: ")
print("Этот атрибут %s существует" %art)
print("Значение атрибута: %s", getter(wolf, art))
estate(wolf, art, 10)
print(getter(wolf, art))

#Задание 4
print("\nЗадание 4")

print("doc: ", Class_Wolf.__doc__)
print("name: ", Class_Wolf.__name__)
print("module: ", Class_Wolf.__module__)
print("bases: ", Class_Wolf.__bases__)
print("dict: ", Class_Wolf.__dict__)

#Задание 5
print("\nЗадание 5")

wolf1 = Class_Wolf(10)
wolf2 = wolf1
wolf3 = wolf1
print(id(wolf1), id(wolf2), id(wolf3))
del wolf

```

```

del wolf1
del wolf2
del wolf3
try:
    print(id(wolf1), id(wolf2), id(wolf3))
except Name Error:
    print("Удаление экземпляров прошло успешно")

```

## Листинг 10. Задание1.py

```

'''
K16_3. Техника работы с классами.
Задание 1. Наследование класса
Задание 2. Переопределение методов
Задание 3. Популярные базовые методы
Задание 4. Приватные методы и атрибуты класса
'''

#Задание 1
print("Задание 1")

class Parant:
    a = 10
    def set_attr(self, attr): Parant.a = attr ** 2
    def parent_method(self): print("Вызов метода родителя")
    def get_attr(self): print("attr: ", Parant.a)
    def power(self, x, y): print(x * y)
    def method(self): print("HELLO")

class Child(Parant):
    def child_method(self): print("Вызов метода класса наследника")
    def power(self, x, y): print(x ** y)
    def method(self): print("HELLO")

c = Child()
c.child_method()
c.parent_method()
c.set_attr(4)
c.get_attr()

#Задание 2
print("\nЗадание 2")

```

```

class Parant:
    a = 10
    def power(self, x, y): print(x * y) #Задание 2
    def method(self): print("HELLO")#Задание 2

class Child(Parant):
    def power(self, x, y): print(x ** y)#Задание 2
    def method(self): print("HELLO")#Задание 2

c.power(10, 4)
c.method()

#Задание 3
print("\nЗадание 3")
class ClassName:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self): return "Vector ({0}, {1})".format(self.x,
self.y)

    def __mul__(self, other): return ClassName(self.x * other.x,
self.y * other.y)

v1 = ClassName(3, 10)
v2 = ClassName(5, 5)
print(v1 * v2)

#Задание 4
print("\nЗадание 4")
class Counter:
    sec_count = 0
    x = 2
    def print_power(self):
        self.x *= 2
        self.__y = 100
        print(self.x)

```

```

    def count(self):
        self.sec_count -= 1
        print(self.sec_count)
coun = Counter()
coun.count()
coun.count()
print(coun.sec_count)

```

```

coun.print_power()
coun.print_power()
print(coun.x)

```

### Листинг 11. Задание1.py

```

'''
Модуль класса
'''

#Задание 3
class ClassSize:
    def __init__(self, width, height):
        self.width = width
        self.height = height
'''

ОСНОВНОЙ КОД
'''

import modul

#Задание 4
rect = modul.ClassSize(17, 28)
print(rect.width)

#Задание 1
class MyClass:
    def __init__(self, cont): self.cont = cont
    def __truediv__(self, other): # Деление с остатком
        line = "-" * len(other.cont)
        return "\n".join([self.cont, line, other.cont])

spam = MyClass("spam")
string = MyClass("hello!")

```

```
print(spam / string)
```

```
#Задание 2
```

```
class MultiPoint:
    countHello = 0
    def __init__(self, name):
        self.name = name
        MultiPoint.countHello += 1
    def printHi(self): print("Hello " + self.name)
    def printCount(self): print(MultiPoint.countHello)
```

```
sq = MultiPoint("Peter")
sq.printHi()
sq.printCount()
```