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:

:

Роголев В.А.
_____(подпись)

Гусятинер Л.Б.
_____(подпись)

_____(оценка)

Королев, 2020

Содержание отчёта

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

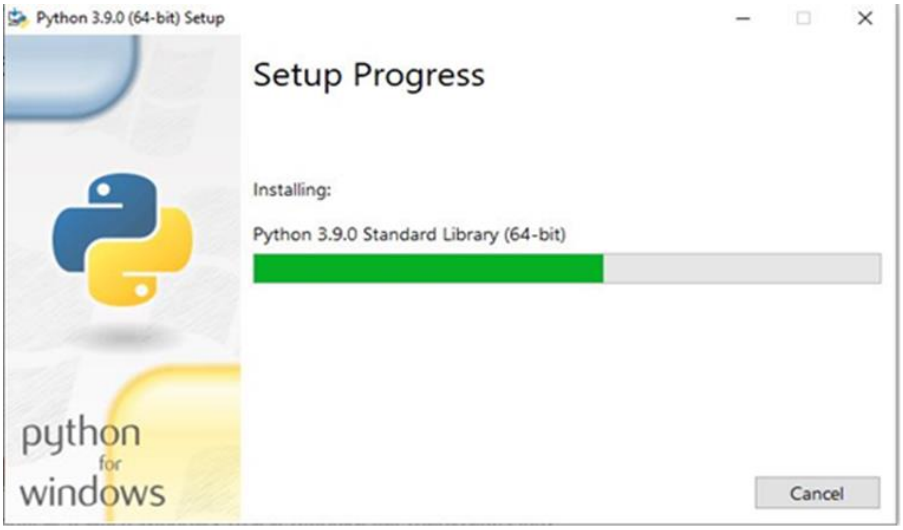

Рис. 2. Процесс установки

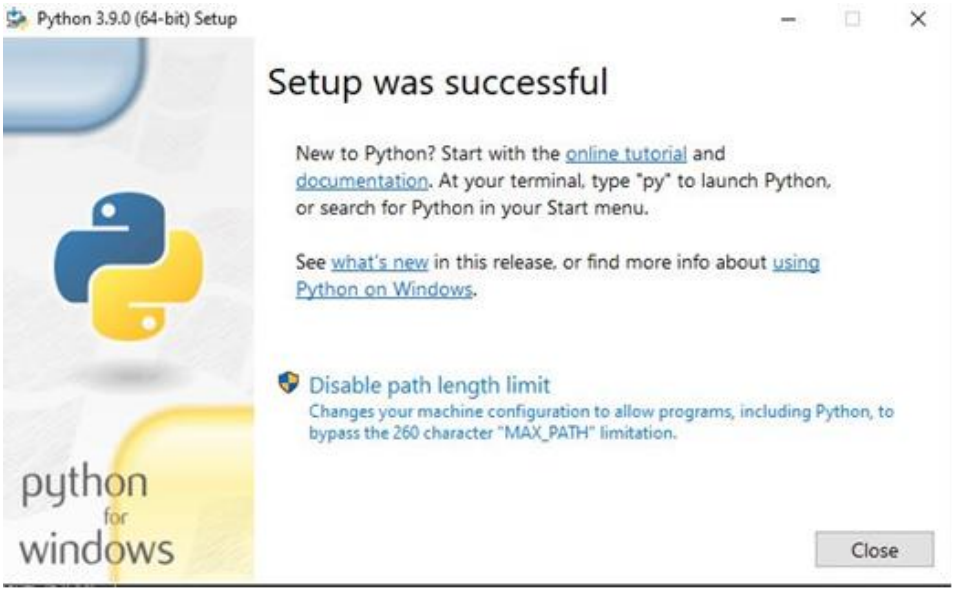

Рис. 3. Окно успешной установки

Процесс удаления описанный ниже нужен на случай, если была установлина 64 разрядная версия, а нужно 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.

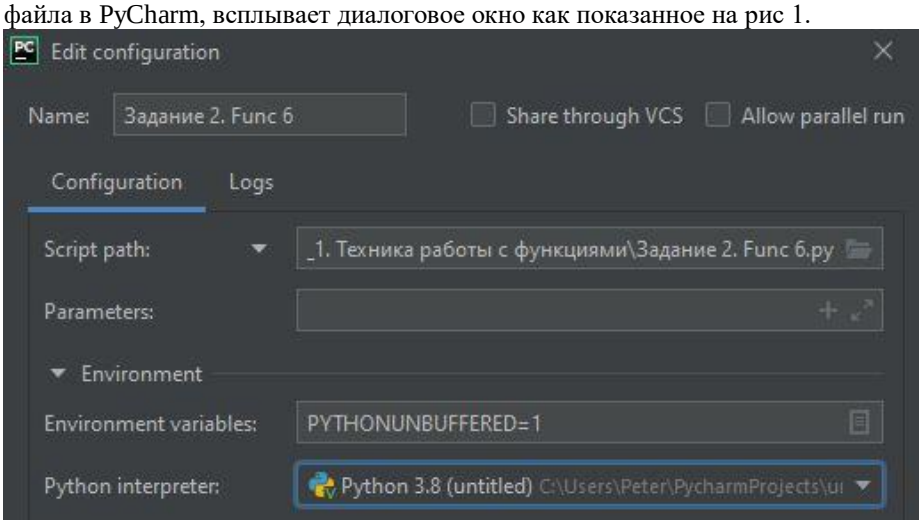

Рис. 1. Выбор интерпретатора

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

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

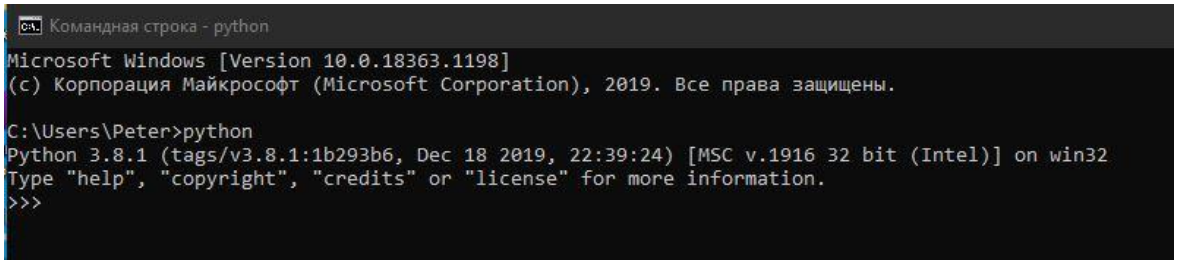

Рис. 2. Интерактивная оболочка python

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

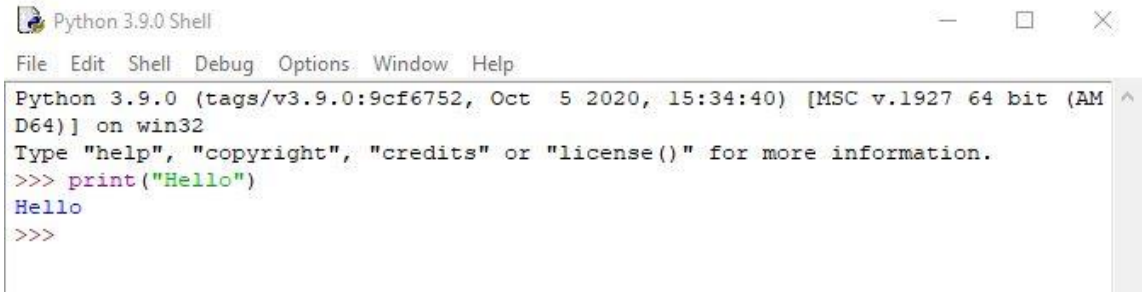

Рис. 3. Демонстрация работы IDEL

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 прост 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. С верху надо нажать кнопку file -> new file затем нажимаем снова file -> save as…

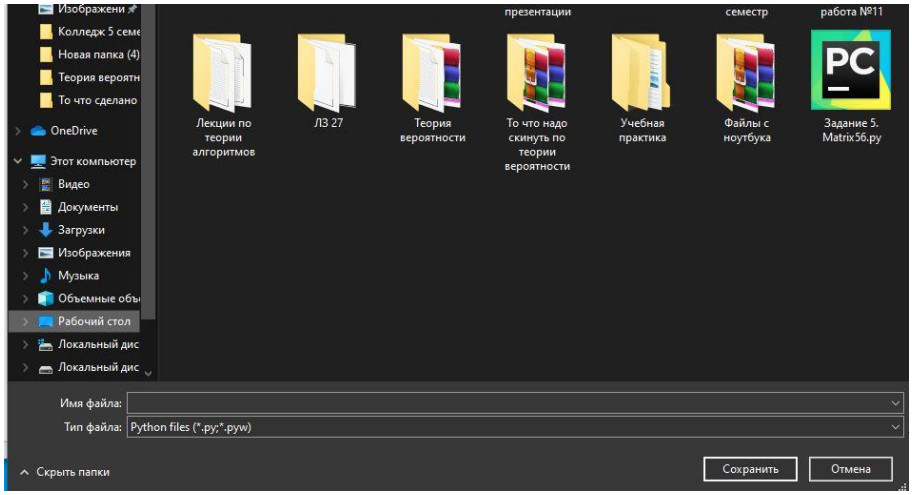

Рис. 5. Сохранение файла

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

Листинг 1. input.py

```
'''
Задание. Разработать программы по темам
- input
Функция input() в Python, ввод данных с клавиатура.
https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-
python/funktsija-input/
'''
# test.py
def str_to_num(line):
    """функция конвертирует строку в число"""
    line = line.strip()
    # если в строке только цифры
    if line.isdigit():
        return int(line)
    # если строка содержит точку или запятую
    elif '.' in line or ',' in line:
        # если из строки убрать точку или запятую
        # и при этом в строке останутся только цифры
        if any(line.replace(x, '').isdigit() for x in ['.', ',']):
            return float(line.replace(',', '.'))
    else:
        # ошибка
        print('Это не число!\n')
        return None

print('\nДля выхода из программы введите Ctrl+C')
print('Для окончания ввода цифр нажмите Enter\n')

nums = []
while True:
    inpt = input('Ожидается ввод числа или Enter:')
    if inpt == '':
        # Закончить ввод чисел
        break
    n = str_to_num(inpt)
    if n is not None:
        nums.append(n)

if nums:
    if len(nums) == 1:
        print('Вы ввели одну цифру: ', nums[0])
    else:
        print('\nВыберите действие:')
        print('  сложить цифры введите 1;')
        print('  умножить цифры введите 2.\n')

        rez = None
        while True:
            inpt = input('Введите 1 или 2:')
            inpt = inpt.strip()
            if inpt == '1':
                rez = sum(nums)
                print('Сумма введенных чисел:', rez)
            elif inpt == '2':
                rez = 1
                for i in nums:
                    rez *= i
                print('Произведение введенных чисел:', rez)
            else:
                print('Неправильное действие.\n')
```

```
            if rez is not None:
                break
else:
    print('Вы ничего не ввели.')
```

Листинг 2. print.py

```
'''
- print
Функция print() в Python, печатает объект.
https://docs-python.ru/tutorial/vstroennye-funktsii-interpretatora-
python/funktsija-print/
'''
print('Hello')
# Hello

print('Hello', 'how are you?')
# Hello how are you?

print('Hello', 'how are you?', sep='---')
# Hello---how are you?

print('Раз', end='=>')
print('Два')
# Раз=>Раз

print(11, 12, 13, 14, sep=';')
# 11;12;13;14
```

Листинг 3. std1.py, std2.py

```
'''
- stdin, stdout, stderr модуля sys
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

stdout = sys.stdout

try:
    sys.stdout = open('file.txt', 'w')
    print('text')
finally:
    # Закрываем file.txt
    sys.stdout.close()
    sys.stdout = sys.__stdout__
```

```
'''- stdin, stdout, stderr модуля sys
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, time
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)

# Строка будет печататься с задержкой, как в телетексте...
```

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

Листинг 4. dateParseToRussian.py

```
'''
- Задание 1. Разработать программу для печати даты прописью
Пример ввода: 15.12.1983
Пример вывода: Пятнадцатое декабря одна тысяча девятсот восемьдесят третьего
года
'''

def get_date(date):
    days = ['первое', 'второе', 'третье', 'четвёртое',
        'пятое', 'шестое', 'седьмое', 'восьмое',
        'девятое', 'десятое', 'одиннадцатое', 'двенадцатое',
        'тринадцатое', 'четырнадцатое', 'пятнадцатое', 'шестнадцатое',
        'семнадцатое', 'восемнадцатое', 'девятнадцатое', 'двадцатое',
        'двадцать первое', 'двадцать второе', 'двадцать третье',
        'двадацать четвёртое', 'двадцать пятое', 'двадцать шестое',
        'двадцать седьмое', 'двадцать восьмое', 'двадцать девятое',
        'тридцатое', 'тридцать первое']

    months = ['января', 'февраля', 'марта', 'апреля', 'мая', 'июня',
           'июля', 'августа', 'сентября', 'октября', 'ноября', 'декабря']

    date = date.split('.')
    return (days[int(date[0]) - 1] + ' ' + months[int(date[1]) - 1] + ' ' +
date[2] + ' года')

date = input()
print(get_date(date))
```

Листинг 5. Menu.py

```
'''
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
'''

options = int(input("Choose from Menu:\n"
             "list: 1.\n"
             "dict: 2.\n"
             "set:  3.\n"))

listMethods = ['list.append(x)      Добавляет элемент в конец списка\n',
'list.extend(L)   Расширяет список list, добавляя в конец все элементы списка
L\n',
'list.insert(i, x)      Вставляет на i-ый элемент значение x\n',
'list.remove(x)   Удаляет первый элемент в списке, имеющий значение x.
ValueError, если такого элемента не существует\n'
'list.pop([i])    Удаляет i-ый элемент и возвращает его. Если индекс не
указан, удаляется последний элемент\n'
'list.index(x, [start [, end]])     Возвращает положение первого элемента со
значением x (при этом поиск ведется от start до end)\n'
'list.count(x)    Возвращает количество элементов со значением x\n'
'list.sort([key=функция])     Сортирует список на основе функции\n'
'list.reverse()   Разворачивает список\n'
'list.copy()      Поверхностная копия списка\n'
'list.clear()     Очищает список\n']

dictMethods = ['dict.clear() - очищает словарь.\n',
```

```
'dict.copy() - возвращает копию словаря.\n',

'classmethod dict.fromkeys(seq[, value]) - создает словарь с ключами из seq и
значением value (по умолчанию None).\n',

'dict.get(key[, default]) - возвращает значение ключа, но если его нет, не
бросает исключение, а возвращает default (по умолчанию None).\n',

'dict.items() - возвращает пары (ключ, значение).\n',

'dict.keys() - возвращает ключи в словаре.\n',

'dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа
нет, возвращает default (по умолчанию бросает исключение).\n',

'dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь
пуст, бросает исключение KeyError. Помните, что словари неупорядочены.\n',

'dict.setdefault(key[, default]) - возвращает значение ключа, но если его
нет, не бросает исключение, а создает ключ с значением default (по умолчанию
None).\n',

'dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из
other. Существующие ключи перезаписываются. Возвращает None (не новый
словарь!).\n',

'dict.values() - возвращает значения в словаре.\n']

setMethods = ['len(s) - число элементов в множестве (размер множества).\n',
'x in s - принадлежит ли x множеству s.\n',
'set.isdisjoint(other) - истина, если set и other не имеют общих
элементов.\n',
'set == other - все элементы set принадлежат other, все элементы other
принадлежат set.\n',
'set.issubset(other) или set <= other - все элементы set принадлежат
other.\n',
'set.issuperset(other) или set >= other - аналогично.\n',
'set.union(other, ...) или set | other | ... - объединение нескольких
множеств.\n',
'set.intersection(other, ...) или set & other & ... - пересечение.\n',
'set.difference(other, ...) или set - other - ... - множество из всех
элементов set, не принадлежащие ни одному из other.\n',
'set.symmetric_difference(other); set ^ other - множество из элементов,
встречающихся в одном множестве, но не встречающиеся в обоих.\n',
'set.copy() - копия множества.\n',
'И операции, непосредственно изменяющие множество:\n',

'set.update(other, ...); set |= other | ... - объединение.\n',
'set.intersection_update(other, ...); set &= other & ... - пересечение.\n',
'set.difference_update(other, ...); set -= other | ... - вычитание.\n',
'set.symmetric_difference_update(other); set ^= other - множество из
элементов, встречающихся в одном множестве, но не встречающиеся в обоих.\n',
'set.add(elem) - добавляет элемент в множество.\n',
'set.remove(elem) - удаляет элемент из множества. KeyError, если такого
элемента не существует.\n',
'set.discard(elem) - удаляет элемент, если он находится в множестве.\n',
'set.pop() - удаляет первый элемент из множества. Так как множества не
упорядочены, нельзя точно сказать, какой элемент будет первым.\n',
'set.clear() - очистка множества.\n']

menuList = [listMethods, dictMethods, setMethods]

print(*menuList[options - 1])
```

1.4. Техника работы с циклическими программами _ цикл while

Листинг 1. Задание 1.py

```
while (1):
    print ('''(P) -> print
(R) -> repit
(B) -> break
''')
    i = input()
    if i == 'P' or i == 'p':
        print("Hello world")
    elif i == 'R' or i == 'r':
        print("Plz repit :)")
        continue
    elif i == 'B' or i == 'b':
        break
    else:
        print("Error")
```

Листинг 2. Задание 2.py

```
import math
import random
x = int(input("X = :>"))   #квадрат
r = x/2                #круг
i=c=0
while(i < 1000):
    px = random.randrange(0,x)
    py = random.randrange(0,x)
    if (py**2 <= r**2-px**2):
        c+=1
    i+=1
print ("В круге " + str(c) + " точек")
print ("Отношение площадей круга  и квадрата" + str(c/1000))
print ("Примерное число Pi :" + str(4*c/1000))
print ("Разница с <<библиотечным>> числом Pi :" + str(math.pi - 4*c/1000))
print("\nProcess returned 0 (0x0) execution time: 9.155 s")
```

Листинг 3. Задание 1.py

```
'''
Задание 1. Вычислить значение sin(x) с точностью до epsilon при помощи
разложения в ряд
'''
import math

def mysin(x,eps):
    n=1
    an=x
    s=0
    while(math.fabs(an)>eps):
         s+=an
         n+=1
         an*=-x*x/(2.*n-1.0)/(2.0*n-2.0);
    return s

x = float(input("X=:>"))
eps = float(input("E=:>"))
print(mysin(x,eps))
```

Листинг 4. Задание 2.py

```
'''
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.
Sample Input 1:
5
-3
8
4
0
Sample Output 1:
14

Sample Input 2:
0
Sample Output 2:
0
'''
s = 0
f = int(1)
while f != 0 :
    f = int(input())
    s = s + f
print(s)
```

Листинг 5. Задание3.py

```
'''
Задание 3.
Разработать программу для нахождения наибольшего общего делителя
'''
a = int(input("A => "))
b = int(input("B => "))
while (b):
    c = a % b;
    a = b;
    b = c;
print(a)
```

Листинг 6. Задание4.py

```
'''
Задание 4.
С использованием результата задания 2 разработать программу для нахождения
наименьшего
общего кратного
'''
a = int(input())
b = int(input())
A = a
B = b
while A % b != 0:
    A = A+a
while B % a != 0:
    B = B+b
if A < B :
    print(A)
else :
    print(B)
```

Листинг 7. Задание 5.py

```
'''
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 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
'''

```
n = int(input())
b = []
b.append(1)
l = len(b)+1
for d in range(1,n,1):
    for i in range(l):
        b.append(l)
    l +=1

print(*b[0:n])
```

1.5. Техника работы

Листинг 1. Задание 1.py

```
'''
Задание 1. Составить и выполнить по 3 примера использования модулей для
работы
с дробными числами (fractions), для точных вычислений (decimal).
'''
from fractions import Fraction
from decimal import Decimal, getcontext

print("\nFraction\n")# Fraction

print("Пример 1") # Пример 1
print(Fraction(33.33))
print(Fraction('33.33'))

print("Пример 2") # Пример 2
print(Fraction(153, 272))
print(Fraction(1, 2) + Fraction(3, 4))

print("Пример 3") # Пример 3
print(Fraction(1, 8) ** Fraction(1, 2))

print("\nDecimal\n")# Decimal

print("Пример 1") # Пример 1
number1 = Decimal("0.1")
number2 = Decimal("0.7")
print(number1 + number2)

print("Пример 2") # Пример 2
getcontext().prec = 2
print(Decimal('4.34') / 4)

getcontext().prec = 3
print(Decimal('4.34') / 4)

print("Пример 3") # Пример 3
getcontext().prec = 4  # установим точность округление
number = Decimal("2.1234123")
print(number.quantize(Decimal('1.000')))
2.123  #  округление до 3 чисел в дробной части
print(number.quantize(Decimal('1.00')))
2.12  #  округление до 2 чисел в дробной части
print(number.quantize(Decimal('1.0')))
2.1  #  округление до 1 числа в дробной части

'''
Задание 1. Составить и выполнить по 3 примера использования модулей для
работы
с дробными числами (fractions), для точных вычислений (decimal).
'''
from fractions import Fraction
from decimal import Decimal, getcontext

print("\nFraction\n")# Fraction

print("Пример 1") # Пример 1
print(Fraction(33.33))
print(Fraction('33.33'))

print("Пример 2") # Пример 2
print(Fraction(153, 272))
```

```
print(Fraction(1, 2) + Fraction(3, 4))

print("Пример 3") # Пример 3
print(Fraction(1, 8) ** Fraction(1, 2))

print("\nDecimal\n")# Decimal

print("Пример 1") # Пример 1
number1 = Decimal("0.1")
number2 = Decimal("0.7")
print(number1 + number2)

print("Пример 2") # Пример 2
getcontext().prec = 2
print(Decimal('4.34') / 4)

getcontext().prec = 3
print(Decimal('4.34') / 4)

print("Пример 3") # Пример 3
getcontext().prec = 4  # установим точность округление
number = Decimal("2.1234123")
print(number.quantize(Decimal('1.000')))
2.123  #  округление до 3 чисел в дробной части
print(number.quantize(Decimal('1.00')))
2.12  #  округление до 2 чисел в дробной части
print(number.quantize(Decimal('1.0')))
2.1  #  округление до 1 числа в дробной части
```

Листинг 2. Задание 2.py

```
Задание 2. Подготовить инструкцию по использованию модулей fractions,
decimal.

                              Модуль Fraction

Этот модуль пригодится в тех случаях, когда необходимо выполнить вычисления с
дробями, или когда результат должен быть выражен в формате дроби.

>>> from fractions import Fraction as frac

>>> print(Fraction(33.33))
2345390243441541/70368744177664

>>> print(Fraction('33.33'))
3333/100
Модуль Fraction особенно полезен, потому что он автоматически уменьшает
дробь. Выглядит это вот так:

>>> Fraction(153, 272)
Fraction(9, 16)
Кроме того, можно выполнять бинарные (двоичные) операции над дробью также
просто, как используеться int или float .
Просто добавить две фракции:

>>> Fraction(1, 2) + Fraction(3, 4)
Fraction(5, 4)
Также можно возвести дробь в степень:

>>> Fraction(1, 8) ** Fraction(1, 2)
0.3535533905932738

                              Модуль Decimal
```

Синтаксис
С помощью Decimal можно создавать десятичные числа.
Decimal обеспечивает поддержку правильного округления десятичной арифметики с плавающей точкой.

```
>>> from decimal import Decimal
>>> number1 = Decimal("0.1")
>>> number2 = Decimal("0.7")

>>> print(number1 + number2)
0.8
```

Decimal, в отличие от float, имеет ряд преимуществ:

работает так же, как школьная арифметика;
десятичные числа представлены точно (в отличие от float, где такие числа как 1.1 и 5.12 не имеют точного представления);
точность десятичного модуля Decimal можно изменять (с помощью getcontext().prec);

Контекст

Базовые параметры Decimal можно посмотреть в его контексте, выполнив функцию getcontext():

```
>>> from decimal import getcontext
>>> getcontext()
Context(prec=3, rounding=ROUND_HALF_EVEN, Emin=-999999, Emax=999999,
        capitals=1, clamp=0, flags=[Inexact, Rounded],
traps=[InvalidOperation,
        DivisionByZero, Overflow])
```

Точность

Контекстом в Deciaml можно управлять, устанавливая свои значения. Например, для того, чтобы управлять точностью Decimal, необходимо изменить параметр контекста prec (от англ. precision – точность):

```
>>> from decimal import Decimal, getcontext
>>> getcontext().prec = 2
>>> print(Decimal('4.34') / 4)
1.1

>>> getcontext().prec = 3
>>> print(Decimal('4.34') / 4)
1.08
```

Округление

Округление осуществляется с помощью метода quantize(). В качестве первого аргумента – объект Decimal, указывающий на формат округления:

```
>>> from decimal import Decimal
>>> getcontext().prec = 4  # установим точность округление
>>> number = Decimal("2.1234123")

>>> print(number.quantize(Decimal('1.000')))
2.123  #  округление до 3 чисел в дробной части

>>> print(number.quantize(Decimal('1.00')))
2.12  #  округление до 2 чисел в дробной части

>>> print(number.quantize(Decimal('1.0')))
2.1  #  округление до 1 числа в дробной части
```

```
Важно: если точность округления установлена в 2 , а формат округления
Decimal('1.00'), возникнет ошибка:

>>> print(number.quantize(Decimal('1.000')))

Traceback (most recent call last):
  File "<pyshell#78>", line 1, in <module>
    print(number.quantize(Decimal('1.00')))
decimal.InvalidOperation: [<class 'decimal.InvalidOperation'>]
Чтобы избежать ее, необходимо поменять точность округления, как было сделано
в примере выше:

>> getcontext().prec = 4
>>> print(number.quantize(Decimal('1.000')))
2.123
Помимо первого параметра, quantize() принимает в качестве второго параметра
стратегию округления:

ROUND_CEILING – округление в направлении бесконечности (Infinity);
ROUND_FLOOR – округляет в направлении минус бесконечности (- Infinity);
ROUND_DOWN – округление в направлении нуля;
ROUND_HALF_EVEN – округление до ближайшего четного числа. Число 4.9
округлится не до 5, а до 4 (потому что 5 – не четное);
ROUND_HALF_DOWN – округление до ближайшего нуля;
ROUND_UP – округление от нуля;
ROUND_05UP – округление от нуля (если последняя цифра после округления до
нуля была бы 0 или 5, в противном случае к нулю).
>>> from decimal import Decimal, ROUND_CEILING

>>> number = Decimal("0.029")
>>> print(number.quantize(Decimal("1.00"), ROUND_CEILING))
0.03

Полезные методы Decimal

sqrt() – вычисляет квадратный корень из десятичного числа;
exp() – возвращает e^x (показатель степени) десятичного числа;
ln() – используется для вычисления натурального логарифма десятичного числа;
log10() – используется для вычисления log (основание 10) десятичного числа;
as_tuple() – возвращает десятичное число, содержащее 3 аргумента, знак (0 для
+, 1 для -), цифры и значение экспоненты;
fma(a, b) – "fma" означает сложить, умножить и добавить. Данный метод
вычисляет (num * a) + b из чисел в аргументе. В этой функции округление num *
a не выполняется;
copy_sign() – печатает первый аргумент, копируя знак из второго аргумента.
```

Листинг 3. Задание 1.txt

```
Задание 1. Подготовить инструкцию по использованию модулей math и cmath.

                                Модули math и cmath.

Первый дает вам доступ к гиперболическим, тригонометрическим и
логарифмическим функциям для действительных чисел,
а последний позволяет работать с комплексными числами. Также все возвращаемые
значения - это float.

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.

```
import math

def getsin(x):

    multiplier = 1
    result = 0

  for i in range(1,20,2):
        result += multiplier*pow(x,i)/math.factorial(i)
        multiplier *= -1

    return result


getsin(math.pi/2) # returns 1.0
getsin(math.pi/4) # returns 0.7071067811865475
```

Еще одна полезная функция в модуле math - gcd(x, y), которая дает вам наибольший общий делитель
(GCD) двух чисел x и y. Когда x и y оба не равны нулю, эта функция возвращает наибольшее
положительное целое число, которое делит как x, так и y. Вы можете косвенно использовать
его для вычисления наименьшего общего кратного двух чисел, используя следующую формулу:

```
gcd(a, b) x lcm(a, b) = a x b
```

Вот несколько арифметических функций, которые предлагает Python:

```
import math

math.ceil(1.001)    # returns 2
math.floor(1.001)   # returns 1
math.factorial(10)  # returns 3628800
math.gcd(10,125)    # returns 5

math.trunc(1.001)   # returns 1
math.trunc(1.999)   # returns 1
```

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 asin(x), acos(x) и atan(x).

Математический модуль обеспечивает функцию hypot(a, b) для вычисления длины гипотенузы.

```
import math

math.sin(math.pi/4)    # returns 0.7071067811865476
math.cos(math.pi)      # returns -1.0
math.tan(math.pi/6)    # returns 0.5773502691896257
math.hypot(12,5)       # returns 13.0

math.atan(0.5773502691896257) # returns 0.5235987755982988
math.asin(0.7071067811865476) # returns 0.7853981633974484
```

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 asinh(x), acosh(x) и atanh(x), которые могут быть использованы
для вычисления обратных соответствующих гиперболических значений.

```
import math

math.sinh(math.pi)    # returns 11.548739357257746
math.cosh(math.pi)    # returns 11.591953275521519
math.cosh(math.pi)    # returns 0.99627207622075

math.asinh(11.548739357257746)   # returns 3.141592653589793
math.acosh(11.591953275521519)   # returns 3.141592653589793
math.atanh(0.99627207622075)     # returns 3.141592653589798
```

Так как math.pi равно примерно 3.141592653589793, когда мы использовали asinh()
для значения, возвращаемого sinh(math.pi), мы получили нашу pi обратно.

Степень и логарифмические функции

Вероятнее всего, вы чаще всего сталкиваетесь со степенями и логарифмами,
чем с гиперболическими или тригонометрическими функциями. К счастью, модуль math предоставляет множество функций, которые помогут нам вычислить логарифмы.

Вы можете использовать log(x, [base]) для вычисления log заданного числа x для данной базы.
Если вы оставите необязательный аргумент базы, log x будет вычисляться до базы e.
Здесь e - математическая константа, значение которой равно 2.71828182 ....

и к ней можно получить доступ с использованием math.e.
Кстати, Python также позволяет вам получить доступ к другой константе pi, используя math.pi.

Если вы хотите рассчитать значения логарифма base-2 или base-10, использование log2(x) и log10(x) вернет более точные результаты, чем log(x, 2) и log(x, 10). Имейте в виду, что функция log3(x) отсутствует, поэтому вам нужно будет использовать log(x, 3) для вычисления значений логарифма базы-3.
То же самое касается всех других баз.

Если значение, логарифм которого вы вычисляете, очень близко к 1, вы можете использовать log1p(x). 1p в log1p означает 1 плюс.
Поэтому log1p(x) вычисляет log(1 + x), где x близок к нулю.
Однако результаты более точны с log1p(x).

Вы также можете рассчитать значение числа x, возведённого в степень y, используя pow(x, y).
Перед вычислением степени эта функция преобразует оба аргумента в тип float.
Если вы хотите, чтобы конечный результат был вычислен в точных целых степенях,
вы должны использовать встроенную функцию pow() или оператор **.

Вы также можете вычислить квадратный корень любого заданного числа x, используя sqrt(x),
но то же самое можно также сделать, используя pow(x, 0.5).

```
import math

math.exp(5)                       # returns 148.4131591025766
math.e**5                         # returns 148.4131591025765

math.log(148.41315910257657)      # returns 5.0
math.log(148.41315910257657, 2)   # returns 7.213475204444817
math.log(148.41315910257657, 10) # returns 2.171472409516258

math.log(1.0000025)               # returns 2.4999968749105643e-06
math.log1p(0.0000025)             # returns 2.4999968750052084e-06

math.pow(12.5, 2.8)               # returns 1178.5500657314767
math.pow(144, 0.5)                # returns 12.0
math.sqrt(144)                    # returns 12.0
```

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 - abs(z), a phi - phase(z).

Аналогично, вы можете преобразовать комплексное число в полярной форме в прямоугольную форму с помощью rect(r, phi).
Комплексное число, возвращаемое этой функцией, равно r * (math.cos (phi) + math.sin (phi) * 1j).

```
import cmath

cmath.polar(complex(1.0, 1.0)) # returns (1.4142135623730951,
0.7853981633974483)

cmath.phase(complex(1.0, 1.0)) # returns 0.7853981633974483

abs(complex(1.0, 1.0)) # returns 1.4142135623730951
```

Модуль cmath также позволяет использовать регулярные математические функции со сложными числами.
Например, вы можете вычислить квадратный корень из комплексного числа, используя sqrt(z) или его косинус, используя cos(z).

```
import cmath

cmath.sqrt(complex(25.0, 25.0)) # returns
(5.49342056733905+2.2754493028111367j)

cmath.cos(complex(25.0, 25.0))      # returns
(35685729345.58163+4764987221.458499j)
```

Комплексные числа имеют множество приложений,
таких как моделирование электрических цепей, динамика жидкости и анализ сигналов.
Если вам нужно работать над любой из этих вещей, модуль cmath не разочарует вас.

1.6. Техника работы со строками

Листинг 1. Задание 1.py

```
'''
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.
'''
a = list()
for i in iter(input, '.'):
    a.append(i.upper())
for i in a:
    print(*i)
```

Листинг 2. Задание 2.py

```
'''
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
'''
login = input()
if not ' ' in login:
```

```
    print(login)
else:
    print(login.replace(' ', '_'))
```

Листинг 3. Задание 3.py

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

Листинг 4. Задание 4.py

```
'''
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:
    a = input()
    if a == ".":
        break
```

```
    elif "@@" in a:
        print("ошибка")
    elif "!!" in a:
        print("предупреждение")
    elif "//" in a:
        print("информация")
    elif "**" in a:
        print("подробное сообщение").
```

Листинг 5. Задание 1.py

```
'''
Задание 1. Подготовить сравнительную инструкцию по использованию
форматирования строк
'''
print("             1 Форматирование строк “По старинке” (оператор %)")
print("name = \"PYH\"")
name = "PYH"
print("print('Hello, %s' %name)")
print("Output: Hello, %s" %name)
print()

print("Вывод в шестнадцатиричного числа")
print("errno = 50159747054")
errno = 50159747054
print("print('%x' % errno)")
print("Output: %x" % errno)
print("_____________________________________")
print()


print("             2 Форматирование строк “По новому” (str.format)")
print("name = \"PYH\"")
name = "PYH"
print("print('Hello, {}'.format(name))")
print("Output:", 'Hello, {}'.format(name))
print()

print("Или")
print("errno = 50159747054")
errno = 50159747054
print("print(")
print("\t'Hey {name}, there is a 0x{errno:x} error!'.format(""")")
print("\t\tname=name, errno=errno")
print("\t)")
print(")")
print(
    'Output: Hey {name}, there is a 0x{errno:x} error!'.format(
        name=name, errno=errno
    )
)
print("_____________________________________")
print()


print("             3 Интерполяция строк / f-Строки (Python 3.6+)")
print("name = \"PYH\"")
name = "PYH"
print("print(f'Hello, {name}!')")
print(f'Output: Hello, {name}!')
print()

print("a = 5")
a = 5
print("b = 10")
b = 10
```

```
print("print(f'Five plus ten is {a + b} and not {2 * (a + b)}.')")
print(f'Five plus ten is {a + b} and not {2 * (a + b)}.')
print("_____________________________________")
print()


print("            4 Шаблонные строки (Стандартная библиотека Template
Strings)")
print("from string import Template")
from string import Template
print("name = \"PYH\"")
name = "PYH"
print("t = Template('Hey, $name!')")
t = Template('Hey, $name!')
print("print(t.substitute(name=name))")
print(t.substitute(name=name))
print()

print("templ_string = 'Hey $name, there is a $error error!'")
templ_string = 'Hey $name, there is a $error error!'

print("print(")
print("     Template(templ_string).substitute(")
print("           name=name, error=hex(errno)")
print("     )")
print(")")
print(
    Template(templ_string).substitute(
        name=name, error=hex(errno)
    )
)
print("_____________________________________")
print()
```

1.7. Техника работы со списками

Листинг 1. Задание 1.py

```
'''
Задание 1.
https://pythontutor.ru/lessons/lists/problems/more_than_neighbours/
Задача «Больше своих соседей»
Дан список чисел. Определите, сколько в этом списке элементов, которые больше
двух
своих соседей, и выведите количество таких элементов. Крайние элементы списка
никогда
не учитываются, поскольку у них недостаточно соседей.
'''

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

Листинг 2. Задание 2.py

```
'''
Задание 2. https://pythontutor.ru/lessons/lists/problems/num_equal_pairs/
Задача «Количество совпадающих пар»
Дан список чисел. Посчитайте, сколько в нем пар элементов, равных друг другу.
Считается, что любые два элемента, равные друг другу образуют одну пару,
которую
необходимо посчитать.
'''
```

```
n = [int(i) for i in input().split()]
a = 0
for i in range(len(n)):
    for j in range(len(n)):
        if n[i] == n[j]:
            a += 1
    a -=1
print(a/2)
```

Листинг 3. Задание 3.py

```
'''
Задание 3. (Л.Б.)
Дано N списков целых чисел (N вводится с клавиатуры, сами списки заполняются
случайным образом). Требуется сформировать
- список, содержащий уникальные значения, попадающие в каждый из N списков
- список, содержащий уникальные значения, попадающие хотя бы в один из N
списков
Решение без использования set - дополнительный бонус
'''

import random
n = int(input())
mtrx = [[random.randint(0, n) for i in range(n)] for j in range(n)]

listever =[]
listall =[]


for i in range(n):
        for j in range(n):
                if mtrx[i][j] not in listall:
                        listall.append(mtrx[i][j])


for t in listall:
      c = 0
      for i in range(n):
            if t in mtrx[i]:
                  c += 1
      if (c == n):
            listever.append(t)

print("Матрица:", *mtrx, sep='\n ')
print("Список значений, попадающих в каждый из N списков:\n", *listever)
print("Список значений, попадающие хотя бы в один из N списков:\n", *listall)
```

Листинг 4. Задание1.py

```
'''
К8_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.
'''
from random import randint

n = int(input())
a = []
```

```
for i in range(n):
    a.append(randint(1,100))
print('  ' + str(a))


for i in range(n-1):
    for j in range(n-i-1):
        if a[j] > a[j+1]:
            c = a[j]
            a[j] = a[j+1]
            a[j+1] = c
        print(str(i) + ' ' + str(a))

print('  ' + str(a))
```

Листинг 5. Задание 2.py

```
'''
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.
'''
from random import randint

n = int(input())
a = []
for i in range(n):
    a.append(randint(1, 100))

for i in range(len(a) - 1):
        m = i
        j = i + 1
        while j < len(a):
            if a[j] < a[m]:
                m = j
            j = j + 1
        c = a[i]
        a[i] = a[m]
        a[m] = c
        print(a)
print("->" + str(a))
```

Листинг 6. Задание 3.py

```
'''
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).
'''
from random import randint

n = int(input())
a = []
for i in range(n):
    a.append(randint(1,100))
print("->" + str(a))
for i in range(len(a)):
```

```
        j = i - 1
        key = a[i]
        while a[j] > key and j >= 0:
            a[j + 1] = a[j]
            j -= 1
        a[j + 1] = key
        print(a)
print("->" + str(a))
```

1.8. Техника работы с циклом for и генераторами списков

Листинг 1. Задание 1.py

```
'''
Задание 1. 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.
'''
import random

n = int(input())
a = []
b = []
for i in range(n):
    a.append(random.randint(1, 100))
print("Start:>" + str(a))
for i in range(1, len(a), 2):
    b.append(a[i])
print("Fin:>" + str(b))
print(len(b))
```

Листинг 2. Задание 2.py

```
'''
Задание 2. Array57. Дан целочисленный массив A размера N.
Переписать в новый целочисленный массив B 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.
'''

import random

n = int(input())
a = []
b = []
for i in range(n):
    a.append(random.randint(1, 100))
print("Start:>" + str(a))
for i in range(0, n, 2):
    b.append(a[i])
for i in range(1, n, 2):
    b.append(a[i])

print("Fin:>" + str(b))
```

Листинг 3. Задание 3.py

```
'''
Задание 3. Array58. Дан массив A размера N. Сформировать новый массив B
того же размера по следующему правилу: элемент B[K] равен сумме
элементов массива A с номерами от 0 до K.
'''
import random

n = int(input())
a = []
b = []
for i in range(n):
    a.append(random.randint(1, 100))
print(a)
s = 0
for i in range(len(a)):
    s += a[i]
    b.append(s)
```

```
print(b)
```

Листинг 4. Задание 4.py

```
'''
Задание 4. Matrix3. Даны целые положительные числа M, N и набор из M чисел.
Сформировать матрицу размера M x N, у которой в каждом столбце содержатся
все числа из исходного набора (в том же порядке).
'''
import random
m = int(input())
n = int(input())
a = [random.randint(1, 100) for i in range(m)]
print(a)
mtrx = []
for i in range(m):
    list = []
    for j in range(n):
        list.append(a[i])
    mtrx.append(list)
[print(' '.join([str(j) for j in i])) for i in mtrx]
```

Листинг 5. Задание 5.py

```
'''
Задание 5. Matrix56. Дана матрица размера M x N (N — четное число). Поменять
местами
левую и правую половины матрицы.
'''
import random
m = int(input())
n = int(input())
matrix =[[random.randint(1, 100) for j in range(n)] for i in range(m)]
print('\n'.join(map(str,matrix)))
for i in range(m):
    for j in range(n//2):
        c = matrix[i][j]
        matrix[i][j] = matrix[i][j-n//2]
        matrix[i][j-n//2] = c
print()
print('\n'.join(map(str,matrix)))
'''
print()
for i in range(m):
    print( "{:s}".format(str(mtrx[i])),sep = '\n')
'''
```

Листинг 6. Задание 6.py

```
'''
Задание 6. Matrix88. Дана квадратная матрица порядка M.
Обнулить элементы матрицы, лежащие ниже главной диагонали.
Условный оператор не использовать.
'''
import random
m = int(input())
n = int(input())
matrix =[[random.randint(1, 10) for j in range(n)] for i in range(m)]
print('\n'.join(map(str,matrix)))
for i in range(1,m):
    for j in range(i):
        matrix[i][j] = 0
print()
print('\n'.join(map(str,matrix)))
```

1.9. Техника работы с функциями

Листинг 1. Задание 2.py

```
'''
```

```
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):
      sym = 0
      for i in range(a, b+1):
            sym += i
      return sym

a = int(input())
b = int(input())
c = int(input())

print(SumRange(a, b))
print(SumRange(b, c))
```

Листинг 2. Задание 3.py

```
'''
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):
      sqrtK = math.sqrt(K)
      if (sqrtK == int(sqrtK)):
            return True
      else:
            return False
a=[]
for x in input().split():
        a.append(int(x))
c = 0
for i in a:
      c += IsSquare(i)
print(c)
```

Листинг 3. Задание 4.py

```
'''
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 SortInc3(a):
        for j in range(2):
                for i in range(2):
                        if (a[i] > a[i+1]):
                                c = a[i]
                                a[i] = a[i+1]
                                a[i+1] = c

X=[]
Y=[]
for x in input().split():
        X.append(float(x))
for y in input().split():
        Y.append(float(y))
```

```
SortInc3(X)
print(X)
SortInc3(Y)
print(Y)
```

Листинг 4. Задание1.py

```
'''
Задание 2. https://stepik.org/lesson/201702/step/13?unit=175778
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:
5
16
20
1
9
.
'''
a = [int(x)**2 for x in iter(input, '.')]
b = list(map(lambda x: print(x, end=' '), a[::-1]))
```

Листинг 5. Задание2.py

```
'''
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("N:"))
a = []
for i in range(n):
    a.append(int(input()))
a = list(filter(lambda x: x%2, a))
print("A=>", *a, sep=" ")
print("Размер A=>", len(a))
```

Листинг 6. Задание3.py

```
'''
Задание 4. Использовать lambda, map.
https://stepik.org/lesson/239422/step/2?unit=211833
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.
Sample Input:
5 12 3
Sample Output:
125 216 343 512 729 1000 1331 1728
'''
a=[]
for i in input().split():
    a.append(int(i))
arr = list(map(lambda x: x**a[2], range(a[0], a[1]+1)))
print(*arr)
```

1.10. Техника работы со словарями

Листинг 1. Задание 1.py

```
'''
Задание 1. https://pythontutor.ru/lessons/dicts/problems/occurency_index/
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.
'''

count = {}
arr = input().split()
for i in arr:
    if i not in count:
        count[i] = 0
    else:
        count[i] += 1
    print(count[i], end=" ")
```

Листинг 2. Задание 2.py

```
'''
Задание 2. https://pythontutor.ru/lessons/dicts/problems/permissions/
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 колличество запросов.
Вам требуется восстановить контроль над правами доступа к файлам
(ваша программа для каждого запроса должна будет возвращать
OK если над файлом выполняется допустимая операция,
или же Access denied, если операция недопустима.
'''

FPerm= dict()
for i in range(int(input("N=>"))):
    list = input(f"{i+1} ").split()
    FPerm[list[0]] = list[1:]

for i in range(int(input("N=>"))):
    Com, name = input(f"{i+1} ").split()
    Com = {'execute' : 'X', 'write' : 'W', 'read' : 'R'}
    if Com in FPerm[name]:
        print('OK')
    else:
        print('Access denied')
```

Листинг 3. Задание 3.py

```
'''
Задание 3. https://pythontutor.ru/lessons/dicts/problems/most_frequent_word/
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.
'''

tdict = list()
for i in range( int(input()) ):
    word = input().split()
    for C in word:
        tdict.append(C)

D = dict()
for k in tdict:
    D[k] = D.get(k, 0) + 1

maxi = max(D.values())
for K, Val in sorted(D.items()):
    if (Val == maxi):
        print(K)
        break
```

Листинг 4. Задание 1.py

```
'''
Задание 1. https://stepik.org/lesson/243394/step/4?unit=215740
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.

Sample Input:
Ben 89001234050
Alice 210-220
Alice
Alice 404-502
Nick +16507811251
Ben
Alex +4(908)273-22-42
Alice
Nick
Robert 51234047129
Alex
.

Sample Output:
210-220
89001234050
404-502
+16507811251
+4(908)273-22-42
'''

D = {}
while True:
    a = input()
    if a == '.':
        break
    a = a.split()
    if len(a) == 2:
        D[a[0]] = D[1]
    else:
```

```
        print(D[a[0]])
```
Листинг 5. Задание 2.py
```
'''
Задание 2. https://stepik.org/lesson/243394/step/8?unit=215740
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".

Sample Input:
Ben 89001234050, +70504321009
Alice 210-220
Alice
Alice 404-502, 894-005, 439-095
Nick +16507811251
Ben
Alex +4(908)273-22-42
Alice
Nick
Robert 51234047129, 92174043215
Alex
Robert
.

Sample Output:
210-220
89001234050, +70504321009
210-220, 404-502, 894-005, 439-095
+16507811251
+4(908)273-22-42
51234047129, 92174043215
'''
a = [i for i in iter(input, '.')]
d = {}
for i in a:
    tel = i.split(' ', 1)
    if len(tel) == 2:
        d[tel[0]] = tel[1] if not d.get(tel[0]) else d[tel[0]] + ', ' +
tel[1]
    else:
        print(d.get(i, 'Не найдено'))
```
Листинг 6. Задание3.py
```
'''
Задание 3. https://stepik.org/lesson/243394/step/13?unit=215740
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.

Sample Input:
Ben 89001234050, +7 050 432 10-09
Alice 404-502, 894053212-65, 439-095
Nick +1(650)781 12-51
Ben
Alex +4(908)273-22-421, 8 (908) 273-22-42
Alice
Nick
Robert 51234047129, 89174043215
Alex
Robert
.

Sample Output:
+7 (900) 123-40-50, +7 (050) 432-10-09
+7 (940) 532-12-65
Не найдено
+7 (908) 273-22-42
+7 (917) 404-32-15
'''

```
def GPhone(number):
    for i in [" ", "-", "+", "(", ")"]:
        number = number.replace(i, '')
    if len(number) == 11 and (number[0] == '8' or number[0] == '7'):
        number = f'+7 ({number[1:4]}) {number[4:7]}-{number[7:9]}-
{number[9:]}'
        return number
    else:
        return False

line, phonebook= [', '.join(i.split(maxsplit=1)).split(', ') for i in
iter(input, '.')], {}
for i in line:
    if len(i) > 1:
        for j in range(1, len(i)):
            if GPhone(i[j]):
                phonebook.setdefault(i[0], []).append(GPhone(i[j]))
    elif len(i) == 1:
        print(*phonebook.get(i[0], ["Не найдено"]), sep=', ')
```

1.11. Техника работы с множествами

Листинг 1. Задание1.py

```
'''
Задание 1. https://pythontutor.ru/lessons/sets/problems/number_of_unique/
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

```
'''
Задание 2.
https://pythontutor.ru/lessons/sets/problems/number_of_coincidental/
Задача «Количество совпадающих чисел»
Условие. Даны два списка чисел. Посчитайте, сколько чисел содержится
одновременно как
в первом списке, так и во втором.
'''

str1 = set(input().split())
str2 = set(input().split())
p = 0
for i in str1:
    for j in str2:
        if i is j:
            p += 1
print(p)
```

Листинг 3. Задание3.py

```
'''
Задание 3. https://pythontutor.ru/lessons/sets/problems/sets_intersection/
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.
'''
str1 = set(map(int, input().split()))
str2 = set(map(int, input().split()))
lst = []
fin = set()
for i in str2:
    for j in str3:
        if (i == j):
            lst.append(i)
print(*sorted(lst))
```

Листинг 4. Задание4.py

```
'''
Задание 4. https://pythontutor.ru/lessons/sets/problems/number_of_words/
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.
'''

words = set()
for i in range(int(input())):
    words.update(input().split())
```

```
print(len(words))
```

Листинг 5. Задание5.py

```
'''
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.
'''

Lang = set()
n = int(input())
f = 0
KAL = True
for i in range(n):
    m = int(input())
    if (m == 1 and f == 0):
        str1 = input()
        print(1)
        KAL = False
        f += 1
        print(str1)
    else:
        for cLang in range(m):
            str1 = input()
            Lang.add(str1)

if KAL:
    print(len(Lang))
    Lang = sorted(Lang)
    for i in Lang:
        print(i)
print(len(Lang))
Lang = sorted(Lang)
for i in Lang:
    print(i)
```

Листинг 6. Задание1.py

```
'''
Задание 1. https://stepik.org/lesson/3380/step/3?unit=963
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.
'''

m = int(input())
Dict = [input().lower() for i in range(m)]
n = int(input())
data = [input() for i in range(n)]

unknown = []
for l in data:
    words = l.lower().split(' ')
    unknown += [Word for Word in words if Word not in Dict]

print(*set(unknown), sep='\n')
```

1.12. Техника работы с кортежами

Листинг 1. Задание1.py

```
'''
Задание 1. https://stepik.org/lesson/193753/step/4?unit=168148
Вывести чётные
Необходимо вывести все четные числа на отрезке [a; a * 10].
Sample Input:
2
Sample Output:
(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
'''

n = int(input())
if (n%2 == 1):
    print(tuple(range (n+1, n*10 + 1, 2)))
else:
    print(tuple(range (n, n*10 + 1, 2)))
```

Листинг 2. Задание 2.py

```
'''
Задание 2. https://stepik.org/lesson/193753/step/5?unit=168148
Убывающий ряд.
С клавиатуры вводятся целые числа a > b. Выведите убывающую
последовательность чисел
по одному числу в строке.
Sample Input:
-2
-8
Sample Output:
-2
-3
-4
-5
-6
-7
'''

a = int(input())
b = int(input())
for i in range(a, b, -1):
    print(i)
```

1.14. Техника работы с модулями

Листинг 1. Задание1.py

```
'''
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

str1 = input()
str2 = "234523"
Dq = collections.deque(str1)
Dq.append(str2)
print(Dq)
Dq.extend('ehwr')
print(Dq)
Dq.extendleft('ab')
print(Dq)
```

```
print()
print("Dq.index('a', 1) =>", Dq.index('a', 1))
print("Dq.pop() =>", Dq.pop())
print(Dq)
print("Dq.popleft() =>", Dq.popleft())
print(Dq)
print()
Dq.reverse()
print(Dq)
Dq.rotate(1)
print(Dq)
Dq.rotate(2)
print(Dq)
Dq.rotate(-2)
print(Dq)
Dq.rotate(-1)
print(Dq)
```

Листинг 2. Задание2.py

```
'''
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)
sorted(cnt.elements())
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)
print(cnt1)
cnt1.update(cnt)
print(cnt1)
print()
print("#print(cnt + cnt1)")
print(cnt + cnt1)
print("#print(cnt - cnt1)")
print(cnt - cnt1)
print("#print(cnt & cnt1)")
print(cnt & cnt1)
print("#print(cnt | cnt1)")
print(cnt | cnt1)
print()
print(cnt.items())
print(cnt.values())
cnt.clear()
str1 = ""
c = collections.Counter()
with open("text.txt", "r") as file:
    for i in file:
        str1 += i
c = collections.Counter(str1).most_common(len(str1))
print(c)
cn = collections.Counter()
with open("text.txt", "r") as file:
    for i in file:
        Words = re.findall(r'\w+', file.read())
cn = collections.Counter(Words).most_common(len(Words))
```

```
print(cn)
```

Листинг 3. Задание 1.py

```
'''
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 Counter
from collections import defaultdict

print("<Counter>")
count = Counter([1, 2, 3, 2])
dict(count)
print(count)
print(sorted(count.elements()))
count.update('100')
print(count)
print()

print("<defaultdict>")
s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
d = defaultdict(list)
for k, val in s:
    d[k].append(val)
print(sorted(d.items()))
s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
d = {}
for k, val in s:
    d.setdefault(k, []).append(val)
print(d)
s = 'mississippi'
d = defaultdict(int)
for k in s:
    d[k] += 1
print(sorted(d.items()))
```

Листинг 4. Задание 2.py

```
'''
Задание 2. Контейнерные типы данных модуля collections.
Класс OrderedDict() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-python/klass-
ordereddict-modulja-collections/
'''

from collections import OrderedDict
W = OrderedDict.fromkeys('Hello, world!')
W.move_to_end('H')
print(''.join(W.keys()))
W.move_to_end('H', last=False)
print(''.join(W.keys()))
W.popitem(last=True)
print(''.join(W.keys()))
```

Листинг 5. Задание 1.py

```
'''
Модуль sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/
Задание 1. Функция argv модуля sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/funktsija-argv-
modulja-sys/
'''
```

```
import sys

print("Количество аргументов:>" + str(sys.argv))
print()
print("Поседний аргумент:>" + str(sys.argv[-1]))
print()
print("Все аргументы:>" + str(len(sys.argv)))
for i in range(len(sys.argv)):
      print(sys.argv[i], end=" ")
```

Листинг 6. Задание 2.py

```
'''
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('win32'):
    print(f"This is Windows :> {os.name}")
    print(sys.getwindowsversion())
else:
    print(f"WTF?. This is :> {os.name}")
    print(sys.getwindowsversion())
```

Листинг 7. Задание3.py

```
'''
Задание 3. Различные сведения о версии Python.
https://docs-python.ru/standart-library/modul-sys-python/razlichnye-
svedenija-versii/
'''
import sys
str1 = sys.version_info
print(str1)
print()
print(sys.copyright)
print()
print("API C languages: ", sys.api_version)
print()
print(sys.version)
print()
print("Hex version: ", sys.hexversion)
```

Листинг 8. Задание4.py

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

import sys
import os

print()
print(sys.prefix)
path_dir = "."
path_const = path_dir
n = int(input("N < 30 :>"))
for i in range(0, n):
```

```
    path = path_dir + "/Папка " + str(i)
    os.mkdir(path)
    path_const = "./Папка " + str(i+1)
    os.mkdir(path_const)
    path_dir += "/Папка " + str(i)
    print(os.getcwd())
print()
print(sys.base_prefix)
print(sys.exec_prefix)
print(sys.base_exec_prefix)
print(sys.executable)
```

Листинг 9. Задание 5.py

```
'''
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.txt", "r")
    s = input()
    print("str: ", s)
finally:
    sys.stdin.close()
    sys.stdin = stdin

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

Листинг 10. Задание 6.py

```
'''
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 ("-exit" in sys.argv) or ("-e" in sys.argv):
            sys.exit(0)
```

Листинг 11. Задание 1-11.py

```
'''
Задание 1. Вывод текущей директории
Задание 2. Создание папки
Задание 3. Изменение директории
Задание 4. Создание вложенных папок
Задание 5. Создание файлов
Задание 6. Переименование файлов
Задание 7. Перемещение файлов
Задание 8. Список файлов и директорий
Задание 9. Удаление файлов
Задание 10. Удаление директорий
Задание 11. Получение информации о файлах
'''
import os
```

```
# Задание 1. Вывод текущей директории
print("Derectory:", os.getcwd())
print()
# Задание 2. Создание папки
if not os.path.isdir("Folder"):
    os.mkdir("Folder")
# Задание 3. Изменение директории
os.chdir("./Folder")
# Задание 4. Создание вложенных папок
if not os.path.isdir("new_folder"):
    os.mkdir("new_folder")
os.chdir("new_folder")
print("Derectory:", os.getcwd())
print()
os.chdir("..")
path_dir = os.getcwd()
n = int(input("Count of subfolder:> "))
for i in range(n):
    path = path_dir + "/Folder " + str(i)
    os.mkdir(path)
    path_dir += "/Folder " + str(i)
print()
#Задание 5. Создание файлов
file = open("file.txt", "w")
file.write("OOO text file OOO")
file.close()

#Задание 6. Переименование файлов
file = open("newfile.txt", "w")
file.write("OOO text newfile OOO")
file.close()
Second_name = input("Rename:> ")
os.rename("newfile.txt", Second_name)
print()


#Задание 7. Перемещение файлов
file = open("First_name.txt", "w")
file.close()
os.replace("First_name.txt", "new_folder/First_name.txt")

#Задание 8. Список файлов и директорий
for dirpath, dirnames, filenames in os.walk("."):
      for dirname in dirnames:
            print("Katalog: ", os.path.join(dirpath, dirname))
      for filename in filenames:
            print("Files: ", os.path.join(dirpath, filename))
print()

#Задание 9. Удаление файлов
File_name = input("Enter name for Del:> ")
f = open(File_name, "w")
f.close()
os.remove(File_name)

#Задание 10. Удаление директорий
os.mkdir('')
print("This derectory: ", os.getcwd())
print("All folder and files: ", os.listdir())
del_name = input("Enter folder aand files for Del:>")
os.rmdir(del_name)
print()

#Задание 11. Получение информации о файлах
```

```
f = open("text.txt", "w")
f.write('Hello, World')
f.close()
print(os.stat("text.txt"))
os.remove("text.txt")
print()
```

1.15. Техника работы с классами

Листинг 1. Задание 1-8.py

```
'''
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. Создание класса
class TestClass:
        x = 20
# Задание 2. Создание объекта
p = TestClass()
print(p.x)
#Задание 3. Функция init
class DataHero:
        # Задание 5. Параметр self
        def __init__(self, name, rank):
                self.name = name
                self.rank = rank
                self.STR = self.rank//4 + 10
                self.DEX = self.rank//4 + 10
                self.INT = self.rank//4 + 10
                self.HP = self.rank//4 + 10
                self.Health = self.HP*self.DEX*10
                self.MP = self.INT*10
        # Задание 4. Методы объектов
        def Enter(self):
                print("Hi Hero! what are you doing?")
        def DataHero(date):
                print(f"++++++++++++++++++++++ ")
                print(f"Name: {date.name}")
                print(f"Level: {date.rank}")
                print(f"Health: {date.HP}")
                print(f"Mana: {date.MP}")
                print(f"++++++++++++++++++++++ ")
                print()
        def UnderDataHero(date):
                print(f"++++++++++++++++++++++ ")
                print(f"STR: {date.STR}")
                print(f"DEX: {date.DEX}")
                print(f"INT: {date.INT}")
                print(f"HP: {date.HP}")
                print(f"++++++++++++++++++++++ ")
                print()

hero1 = DataHero("Hagemanto", 20)

hero1.DataHero()
hero1.UnderDataHero()

# Задание 6. Изменение свойств объекта
hero1.Enter()
hero1.rank = 99960
hero1 = DataHero(hero1.name,hero1.rank)
hero1.DataHero()
hero1.UnderDataHero()
# Задание 7. Удалить свойства объекта
```

```
del hero1.MP
# Задание 8. Удаление объектов
del hero1
```

Листинг 2. Задание 1-5.py

```
'''
Задание 1. Создание классов
Задание 2. Создание экземпляров класса
Задание 3. Доступ к атрибутам
Задание 4. Встроенные атрибуты класса
Задание 5. Уничтожение объектов (сбор мусора)
'''

#1. Создание классов
class DataHero:
        objcount = 0
        def __init__(self, name, rank):
                self.name = name
                self.rank = rank
                DataHero.objcount += 1
                self.STR = self.rank//4 + 10
                self.DEX = self.rank//4 + 10
                self.INT = self.rank//4 + 10
                self.HP = self.rank//4 + 10
                self.Health = self.HP*self.DEX*10
                self.MP = self.INT*10
        #Методы класса
        #Количество объектов класса
        def ObjCount(self):
                print(f"Count objects: {DataHero.objcount}")
        def Enter(self):
                print("Hi Hero! what are you doing?")
        def DataHero(date):
                print(f"++++++++++++++++++++++ ")
                print(f"Name: {date.name}")
                print(f"Level: {date.rank}")
                print(f"Health: {date.HP}")
                print(f"Mana: {date.MP}")
                print(f"++++++++++++++++++++++ ")
                print()
        def UnderDataHero(date):
                print(f"++++++++++++++++++++++ ")
                print(f"STR: {date.STR}")
                print(f"DEX: {date.DEX}")
                print(f"INT: {date.INT}")
                print(f"HP: {date.HP}")
                print(f"++++++++++++++++++++++ ")
                print()

#2. Создание экземпляров класса
hero1 = DataHero("Hagemanto", 25)   #Лысый плащ
hero2 = DataHero("Oni_Saibōgu", 19)     #Генос

print(hero1.__doc__)
print(DataHero.__doc__)

#3. Доступ к атрибутам
hero1.Enter()
hero1.DataHero()

print(f"Count create hero: {DataHero.objcount}")

hero1.age = 9999
hero1.DataHero()
```

```
del hero2.rank
del hero2
print()

#4. Встроенные атрибуты класса
print("doc: ", DataHero.__doc__)
print("name: ", DataHero.__name__)
print("module: ", DataHero.__module__)
print("bases: ", DataHero.__bases__)
print("dict: ", DataHero.__dict__)

#5. Уничтожение объектов (сбор мусора)
hero3 = DataHero("musor",1)
del hero3
```