

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:

Константинович А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Гусятинер Л. Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2020

Содержание отчёта.

[Раздел 1. Техника решения задач с использованием структурного программирования. 2](#_Toc59321267)

[1.1. Установка интерпретатора Python 3 и настройка окружения 2](#_Toc59321268)

[1.2. Техника работы в командной строке и среде IDLE 8](#_Toc59321269)

[1.3. Техника работы с линейными и разветвляющимися программами. 9](#_Toc59321270)

[1.4. Техника работы с циклическими программами, цикл while. 12](#_Toc59321271)

[1.5. Техника работы с числами. 16](#_Toc59321272)

[1.6. Техника работы со строками. 17](#_Toc59321273)

[1.7. Техника работы со списками. 18](#_Toc59321274)

[1.8. Техника работы с циклом for и генератором списков. 22](#_Toc59321275)

[1.9. Техника работы с функциями. 25](#_Toc59321276)

[1.10. Техника работы со словарями. 27](#_Toc59321277)

[1.11. Техника работы с множествами. 31](#_Toc59321278)

[1.12. Техника работы с кортежами. 33](#_Toc59321279)

[1.13. Техника работы с файлами. 35](#_Toc59321280)

[1.14. Техника работы с модулями. 38](#_Toc59321281)

[1.15. Техника работы с классами. 42](#_Toc59321282)

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

## Установка интерпретатора Python 3 и настройка окружения

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



Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.

2. Выбрать способ установки.



Рисунок 2. Установщик Python

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



Рисунок 3. Опции установки

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

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

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



Рисунок 4. Продвинутые опции установки

5. После успешной установки:



Рисунок 5. Сообщение об установке

## Техника работы в командной строке и среде IDLE

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



Рисунок 6. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

>>> exit()

или же

>>> quit()

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

[IDLE](https://docs.python.org/2/library/idle.html) - простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

>>>

Теперь напишите в подсказке следующий код:

>>> print("Hello, World")

Нажмите Enter .

>>> print("Hello, World")

Hello, World



Рисунок 7. Первая программа

## Техника работы с линейными и разветвляющимися программами.

**Задание 1.** Разработать программы по темам

- input

- print

- stdin, stdout, stderr модуля sys

- форматная строка и метод формат

**Листинг К4\_1/1.py:**

import sys

x = int(input())

sys.stdout.write(("{0}, {1}, {2}".format(x-1, x, x+1)))

**Задание 2.** Разработать программу для печати даты прописью

Пример ввода: 15.12.1983

Пример вывода: Пятнадцатое декабря одна тысяча девятсот восемьдесят третьего года

**Листинг К4\_2/1.py:**

def getDay(n):

days = ["", "Первое", "Второе", "Третье", "Четвертое", "Пятое", "Шестое", "Седьмое", "Восьмое", "Девятое", "Десятое", "Одиннадцатое",

"Двенадцатое", "Тринадцатое", "Четырнадцатое", "Пятнадцатое", "Шестнадцатое", "Семнадцатое", "Восемнадцатое", "Девятнадцатое", "Двадцатое",

"Двадцать первое", "Двадцать второе", "Двадцать третье", "Двадцать четвертое", "Двадцать пятое"

, "Двадцать шестое", "Двадцать седьмое", "Двадцать восьмое", "Двадцать девятое", "Тридцатое", "Тридцать первое"]

return days[n]

def getMonth(n):

months = ["", "Января", "Февраля", "Марта", "Апреля", "Мая", "Июня", "Июля", "Августа", "Сентября", "Октября", "Ноября",

"Декабря"]

return months[n]

def getYear(n):

result = ""

# get thousands (n <= 10000)

thousandNumbers = ["", "одна тысяча", "две тысячи", "три тысячи", "четыре тысячи", "пять тысяч", "шесть тысяч", "семь тысяч", "восемь тысяч",

"девять тысяч", "десять тысяч"]

lastThousands = ["", "одно тысячного", "двух тысячного", "трех тысячного", "четырех тысячного", "пяти тысячного",

"шести тысячного", "семи тысячного", "восьми тысячного", "девяти тысячного", "десяти тысячного"]

thousands = n // 1000

# get hundreds

hundredsNumbers = ["", "сто", "двести", "триста", "четыреста", "пятьсот", "шестьсот", "семьсот", "восемьсот", "девятсот"]

lastHundreds = ["", "сотого", "двухсотого", "трехсотого", "четырехсотого", "пятисотого", "шестисотого", "семисотого", "восьмисотого", "девятсотого"]

hundreds = n % 1000 // 100

# get dozens and last number

dozensNumbers = ["", "", "двадцать", "тридцать", "сорок", "пятьдесят", "шестьдесят", "семьдесят", "восемьдесят", "девяносто"]

lastDozens = ["" ,"десятого", "двадцатого", "тридцатого", "сорокового", "пятидесятого", "шестидесятого", "семидесятого", "восьмидесятого", "девяностого"]

dozens = n % 100 // 10

oneDozenNumbers = ["", "одиннадцатого", "двенадцатого", "тринадцатого", "четырнадцатого", "пятнадцатого", "шестнадцатого", "семнадцатого",

"восемнадцатого", "девятнадцатого"]

lastNumber = n % 10

lastNumbers = ["", "первого", "второго", "третьего", "четвертого", "пятого", "шестого", "седьмого", "восьмого", "девятого"]

result += "года"

if lastNumber == 0:

result = lastDozens[dozens] + " " + result

elif dozens == 1:

result = oneDozenNumbers[lastNumber] + " " + result

else:

result = lastNumbers[lastNumber] + " " + result

result = dozensNumbers[dozens] + " " + result

if hundreds == 0 and dozens == 0 and lastNumber == 0:

result = lastThousands[thousands] + " " + result

else:

result = hundredsNumbers[hundreds] + " " + result

result = thousandNumbers[thousands] + " " + result

return result

def getDate(d, m, y):

return getDay(d) + " " + getMonth(m) + " " + getYear(y)

day, month, year = map(int, input().split("."))

print(getDate(day, month, year))

**Задание 3.** Разработать программу с меню для демонстрации работы с типами данных: список(list), словарь(dict), множество(set)

Меню -> выбор типа данных -> выбор метода -> краткая справка.

**Листинг К4\_2/2.py:**

list\_docs = ["", "Append(x): Add an item to the end of the list.",

"Extend(iterable): Extend the list by appending all the items from the iterable.",

"Insert(i, x): Insert an item at a given position. The first argument is the index of the element before which to insert."]

dict\_docs = ["", "Clear(): Removes all the elements from the dictionary.",

"Values(): Returns a list of all the values in the dictionary.",

"Update({key:value}): Updates the dictionary with the specified key-value pairs."]

set\_docs = ["", "Add(x): Adds an element to the set",

"Discard(x): Remove the specified item",

"Union(...): Return a set containing the union of sets"]

while(1):

print("Выберите структуру данных:")

print("1: List", "2: Dict", "3: Set", "0: Exit", sep='\n')

n = int(input())

if n == 0:

break

print("Выберите метод: ")

if n == 1:

print("1: Append", "2: Extend", "3: Insert", sep='\n')

k = int(input())

print(list\_docs[k])

elif n == 2:

print("1: Clear", "2: Values", "3: Update", sep='\n')

k = int(input())

print(dict\_docs[k])

elif n == 3:

print("1: Add", "2: Discard", "3: Union", sep='\n')

k = int(input())

print(set\_docs[k])

## 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".

**Листинг К5\_1/1.py:**

import math

from random import randrange

w = int(input())

r = w/2

n = 1000

k = 0

for i in range(n):

x = randrange(0, w)

y = randrange(0, w)

if ((x-r)\*\*2 + (y-r)\*\*2 <= r\*\*2):

k += 1

print("{} точек попало внутрь круга".format(k))

print("Отношение площадей круга и квадрата: {}".format(k/n))

print("Примерное число PI: {}".format(4\*k/n))

print("Разница между полученным и библиотечными числом pi: {}".format(abs(math.pi-4\*k/n)))

**Задание 2.** Придумать пример(ы) на использование break / continue /else.

**Листинг К5\_1/2.py:**

# print first 100 odd numbers

k = 0

x = 0

while True:

if k == 100:

break

if x % 2 == 0:

k += 1

x += 1

print(x)

continue

x += 1

**Задание 3**. Вычислить значение sin(x) с точностью до epsilon при помощи разложения в ряд. Построить блок-схему.

**Листинг К5\_2/1.py:**

import math

x = float(input("x: "))

epsilon = float(input("epsilon: "))

sign = 1

\_sum = 0

k = 3

\_next = (x\*\*k)/math.factorial(k)

\_sum = x - \_next

while(\_next > epsilon):

k += 2

\_next = (x\*\*k)/math.factorial(k)

\_sum += sign \* \_next

sign \*= -1

print(\_sum)

Блок схема:

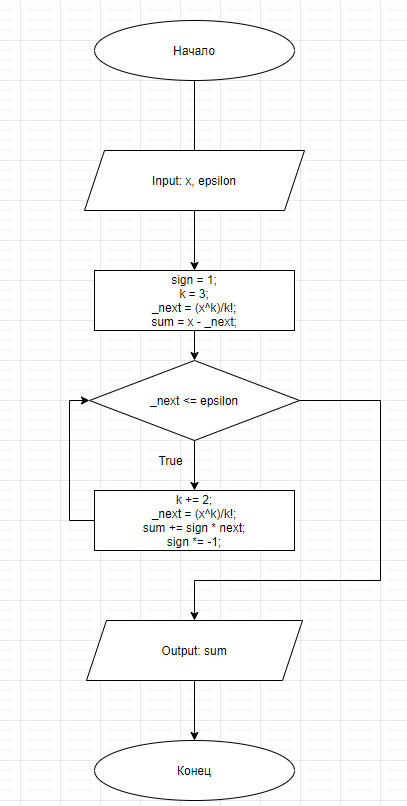


Рисунок 8. Блок схема алгоритма

**Задание 4.**

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.

**Листинг К5\_2/2.py:**

\_sum = 0

n = int(input())

while n != 0:

\_sum += n

n = int(input())

print(\_sum)

**Задание 5.** Разработать программу для нахождения наибольшего общего делителя.

**Листинг К5\_2/3.py:**

def nod (a, b):

return nod (b%a, a) if a else b

a, b = map(int, input().split())

print(nod(a,b))

**Задание 6.** С использованием результата задания 2 разработать программу для нахождения наименьшего общего кратного.

**Листинг К5\_2/4.py:**

def nod(a, b):

return nod(b%a, a) if a else b

def nok(numbers):

while len(numbers) != 1:

a = numbers.pop(0)

b = numbers.pop(0)

c = a\*b/nod(a,b)

numbers.insert(0, c)

return numbers[0]

numbers = []

n = int(input())

while n != 0:

numbers.append(n)

n = int(input())

print(nok(numbers))

**Задание 7.**

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.

**Листинг К5\_2/5.py:**

k = 0

curr = 1

n = int(input())

while k != n:

for i in range(curr):

print(curr, end=' ')

k += 1

if k == n:

break

curr += 1

## Техника работы с числами.

**Задание 1.** Составить и выполнить по 3 примера использования модулей для работы с дробными числами (fractions), для точных вычислений (decimal).

**Листинг К6\_1/1.py:**

####### Fractions library #######

import fractions

# 1 пример: арифметические операции по правилам дробей

a = fractions.Fraction("1/3")

b = fractions.Fraction("2/5")

print(a+b, a-b, a\*b, a/b)

# 2 пример: нахождение НОД

print(fractions.gcd(6,9))

# 3 пример: представление вещественного числа в виде дроби

print(fractions.Fraction("56.45"))

####### Decimal library #######

import decimal

# 1 пример: точные вычисления (0.1 + 0.1 + 0.1 = 0.3)

a = decimal.Decimal("0.1")

print(a + a + a)

# 2 пример: настройка Decimal через getcontext()

print(decimal.Decimal("3")/decimal.Decimal("7"))

decimal.getcontext().prec = 3 # 3 знака после запятой

print(decimal.Decimal("3")/decimal.Decimal("7"))

# 3 пример: представление в виде дроби

print(decimal.Decimal("3.14").as\_integer\_ratio())

## Техника работы со строками.

**Задание 1.** https://stepik.org/lesson/201702/step/5?unit=175778

С клавиатуры вводятся строки, последовательность заканчивается точкой.

Выведите буквы введенных слов в верхнем регистре, разделяя их пробелами.

**Листинг К7\_1/1.py:**

s = input()

for c in s:

print(c.upper(), end=' ')

**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.

**Листинг К7\_1/2.py:**

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

**Задание 3**. https://stepik.org/lesson/201702/step/9?unit=175778

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

**Листинг К7\_1/3.py:**

bits = tuple(map(int, input().split('.')))

print(\*bits)

print(sum(bits))

**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.

**Листинг К7\_1/4.py:**

signals = {"!!" : "предупреждение", "@@" : "ошибка",

"//" : "информация", "\*\*" : "подробное сообщение"}

s = input()

while s != '.':

indicator = s[0:2]

if indicator in signals:

print(signals[indicator])

s = input()

## Техника работы со списками.

**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.

**Листинг К8\_1/1.py:**

numbers = list(map(int, input().split()))

k = 0

for i in range(1, len(numbers)-1):

if numbers[i-1] < numbers[i] > numbers[i+1]:

k += 1

print(k)

**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.

**Листинг К8\_1/2.py:**

numbers = list(map(int, input().split()))

k = 0

for i in range(len(numbers)):

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

if numbers[i] == numbers[j]:

k += 1

print(k)

**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

**Листинг К8\_1/3.py:**

import random

allEntry\_array = []

oneEntry\_array = []

n = int(input())

arrays = []

for i in range(n):

arrays.append([random.randint(1, 20) for x in range(30)])

print(arrays[i])

for c in arrays[0]:

inAll = True

if c not in oneEntry\_array:

oneEntry\_array.append(c)

for j in range(1, n):

inCurr = False

for k in arrays[j]:

if k not in oneEntry\_array:

oneEntry\_array.append(k)

if k == c:

inCurr = True

if inAll:

inAll = inCurr

if inAll and c not in allEntry\_array:

allEntry\_array.append(c)

print(allEntry\_array)

print(oneEntry\_array)

**Задание 4**. 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.

**Листинг К8\_2/1.py:**

import random

def bubbleSort(A):

l = len(A)

for i in range(l-1):

isSorted = True

for j in range(l-i-1):

if A[j] > A[j+1]:

A[j], A[j+1] = A[j+1], A[j]

isSorted = False

if isSorted:

break

A = [random.randint(1,100) for x in range(20)]

print(\*A)

bubbleSort(A);

print(\*A)

**Задание 5**. 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.

**Листинг К8\_2/2.py:**

import random

def selectSort(A):

l = len(A)

c = l-1

for i in range(l-1):

k = A.index(max(A[:c+1]))

A[k], A[c] = A[c], A[k]

c -= 1

print(\*A)

A = [random.randint(1,100) for x in range(20)]

selectSort(A)

**Задание 6.** 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).

**Листинг К8\_2/3.py:**

import random

# binary find position

def upper\_bound(A, k):

r = len(A)-1

l = 0

while l+1 < r:

m = (l+r) // 2

if A[m] > k:

r = m

else:

l = m

if k < A[l]:

return l

elif k <= A[r]:

return r

else:

return r+1

def insertSort(A):

c = 2

l = len(A)

if A[0] > A[1]:

A[0], A[1] = A[1], A[0]

for i in range(2,l):

A.insert(upper\_bound(A[:c], A[i]), A[i]) #add element to sorted part

A.pop(i+1) #delete this element from previous part

c += 1

print(\*A)

A = [random.randint(1,100) for x in range(20)]

insertSort(A)

## Техника работы с циклом for и генератором списков.

**Задание 1.** (Л.Б.) Для проведения конкурса проектов в ККМТ формируются группы из 4х участников: coder, writer, tester, designer, программирующих на одном и том же языке. Каждый студент может программировать только на одном языке и занимать только одну позицию.

Дан текстовый файл, содержащий перечень студентов с указанием языка и позиции (каждый студент с новой строки)

Требуется

1. Получить список студентов с указанием языка и позиции

2. Сформировать список всевозможных команд

3. Вывести список команд с указанием состава и названия команды

**Листинг К9\_1/1.py:**

def getAllLanguages(members):

founded = []

for member in members:

if member[2] not in founded:

founded.append(member[2])

return founded

def printTeams(teams):

k = 1

for team in teams:

print("Команда " + str(k))

print("Coder: " + team[0][0] + " " + team[0][1])

print("Designer: " + team[1][0] + " " + team[1][1])

print("Tester: " + team[2][0] + " " + team[2][1])

print("Writer: " + team[3][0] + " " + team[3][1])

k += 1

print()

f = open("data.txt")

students = [x.split() for x in f]

#team = [coder, designer, tester, writer]

teams = []

for language in getAllLanguages(students):

#coders = findMembers(students, language, "coder")

coders = [x for x in students if x[2] == language and x[3] == "coder"]

designers = [x for x in students if x[2] == language and x[3] == "designer"]

testers = [x for x in students if x[2] == language and x[3] == "tester"]

writers = [x for x in students if x[2] == language and x[3] == "writer"]

for coder in coders:

for designer in designers:

for tester in testers:

for writer in writers:

teams.append([coder, designer, tester, writer])

print([coder, designer, tester, writer])

printTeams(teams)

**Задание 2.** 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.

**Листинг К9\_2/1.py:**

import random

n = int(input())

a = [random.randint(1, 20) for x in range(n)]

b = [a[i] for i in range(1,len(a),2)]

print(len(b))

print(\*b)

**Задание 3**. 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.

**Листинг К9\_2/2.py:**

import random

n = int(input())

a = [random.randint(1, 20) for x in range(n)]

print(\*a)

b = [a[i] for i in range(0,len(a),2)]

for i in range(1, len(a), 2):

b.append(a[i])

print(len(b))

print(\*b)

**Задание 4.** 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.

**Листинг К9\_2/3.py:**

import random

n = int(input())

a = [random.randint(1, 20) for x in range(n)]

b = [sum(a[:k+1]) for k in range(len(a))]

print(\*a)

print(\*b)

**Задание 5**. 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).

**Листинг К9\_2/4.py:**

import random

M, N = list(map(int, input().split()))

a = list(map(int, input().split()))

mtrx = [[a[m] for k in range(N)] for m in range(M)]

print(\*a)

print(\*mtrx)

**Задание 6.** Matrix56. Дана матрица размера M x N (N — четное число). Поменять местами левую и правую половины матрицы.

**Листинг К9\_2/5.py:**

import random

M, N = list(map(int, input().split()))

a = [[random.randint(1,9) for k in range(N)] for j in range(M)]

#print mtrx (non reversed)

for j in a:

print(j)

print()

#reverse mtrx

for k in range(M):

for j in range(N // 2):

indx = (N // 2 + N % 2) + j

a[k][j], a[k][indx] = a[k][indx], a[k][j]

#print reversed mtrx

for j in a:

print(j)

**Задание 7.** Matrix88. Дана квадратная матрица порядка M. Обнулить элементы матрицы, лежащие ниже главной диагонали. Условный оператор не использовать.

**Листинг К9\_2/6.py:**

import random

M = int(input())

a = [[random.randint(1,9) for k in range(M)] for j in range(M)]

for j in a:

print(j)

print()

k = 1

for k in range(1, M):

for j in range(k):

a[k][j] = 0

k += 1

for j in a:

print(j)

## Техника работы с функциями.

**Задание 1.** 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.

**Листинг К10\_1/1.py:**

def sumRange(A, B):

sum = 0

for k in range(A, B+1):

sum += k

return sum

A, B, C = list(map(int, input().split()))

print(sumRange(A, B))

print(sumRange(B, C))

**Задание 2.** 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.

**Листинг К10\_1/2.py:**

def isSquare(k):

n = 0

b = 0

res = False

while b < k:

b = n\*\*2

if b == k:

res = True

break

n += 1

return res

a = map(int, input().split())

for j in a:

print(isSquare(j), end = ' ')

**Задание 3**. 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.

**Листинг К10\_1/3.py:**

def sortInc3(X):

X.sort()

a = list(map(int, input().split()))

b = list(map(int, input().split()))

sortInc3(a)

sortInc3(b)

print(a)

print(b)

**Задание 4.** 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.

**Листинг К10\_2/1.py:**

a = [int(x) for x in iter(input, '.')]

b = list(map(lambda x: print(x\*\*2, end = ' '), a[::-1]))

**Задание 5.** 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.

**Листинг К10\_2/2.py:**

a = list(map(int, input().split()))

print([a[i] for i in filter(lambda x: x % 2 == 1, range(len(a)))])

**Задание 6.** 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.

**Листинг К10\_2/3.py:**

a, b, p = list(map(int, input().split()))

list(map(lambda x: print(x\*\*p, end = ' '), range(a, b+1)))

## Техника работы со словарями.

**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.

**Листинг К11\_1/1.py:**

\_dict = {}

text = input().split()

for word in text:

if word in \_dict:

\_dict[word] += 1

else:

\_dict.update({word : 1})

print(\_dict)

**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 чиcло 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.

**Листинг К11\_1/2.py:**

commands = {'execute' : 'X', 'write' : 'W', 'read' : 'R'}

filesPermissions = {}

n = int(input())

for i in range(n):

file = input().split()

filesPermissions.update({file[0] : file[1:]})

for i in range(int(input())):

cmnd, file = input().split()

cmnd = commands[cmnd]

if cmnd not in filesPermissions[file]:

print("Access denied")

else:

print("OK")

**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.

**Листинг К11\_1/3.py:**

\_dict = {}

for i in range(int(input())):

for word in input().split():

if word in \_dict:

\_dict[word] += 1

else:

\_dict.update({word : 1})

\_max = 0

for k in \_dict:

\_max = max(\_max, \_dict[k])

print(min(list(filter(lambda x: \_dict[x] == \_max, \_dict.keys()))))

**Задание 4**. 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.

**Листинг К11\_2/1.py:**

tel\_book = {}

cmnd = input().split()

while cmnd != ['.']:

if len(cmnd) == 2:

tel\_book.update({cmnd[0] : cmnd[1]})

else:

print(tel\_book[cmnd[0]])

cmnd = input().split()

**Задание 5.** 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".

**Листинг К11\_2/2.py:**

def addTelephone(t\_book, contact, tel, comma):

if comma:

t\_book[contact] += ", " + tel

else:

t\_book[contact] = tel

tel\_book = {}

cmnd = input().split()

while cmnd != ['.']:

if len(cmnd) == 1:

if cmnd[0] in tel\_book:

print(tel\_book[cmnd[0]])

else:

print("Не найдено")

else :

if cmnd[0] not in tel\_book:

tel\_book.update({cmnd[0] : ""})

addTelephone(tel\_book, cmnd[0], cmnd[1].replace(',',''), 0)

if len(cmnd) > 2:

for tel in cmnd[2:]:

addTelephone(tel\_book, cmnd[0], tel.replace(',',''), 1)

else:

for tel in cmnd[1:]:

addTelephone(tel\_book, cmnd[0], tel.replace(',',''), 1)

cmnd = input().split()

## Техника работы с множествами.

**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.

**Листинг К12\_1/1.py:**

print(len(set(map(int, input().split()))))

**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.

**Листинг К12\_1/2.py:**

a = set(input().split())

b = set(input().split())

print(len(a.intersection(b)))

**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.

**Листинг К12\_1/3.py:**

a = set(input().split())

b = set(input().split())

print(\*sorted(a.intersection(b)))

**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.

**Листинг К12\_1/4.py:**

n = int(input())

s = set()

for i in range(n):

line = input().split()

for word in line:

s.add(word)

print(len(s))

**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.

**Листинг К12\_1/5.py:**

n = int(input())

s = set()

students = []

for i in range(n):

k = int(input())

tmp = set()

for i in range(k):

language = input()

s.add(language)

tmp.add(language)

students.append(tmp)

allKnownLanguages = students[0]

for student in students[1:]:

allKnownLanguages = allKnownLanguages.intersection(student)

print(len(allKnownLanguages))

for language in sorted(allKnownLanguages):

print(language)

print(len(s))

for language in sorted(s):

print(language)

**Задание 6**. 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.

**Листинг К12\_2/1.py:**

knownWords = set()

text = set()

for i in range(int(input())):

knownWords.add(input().lower())

for i in range(int(input())):

line = input().split()

for word in line:

text.add(word.lower())

print(knownWords)

print(text)

d = text.difference(knownWords)

for word in d:

print(word)

## Техника работы с кортежами.

**Задание 1.** https://stepik.org/lesson/193753/step/4?unit=168148

Вывести чётные

Необходимо вывести все четные числа на отрезке [a; a \* 10].

**Листинг К13\_1/1.py:**

a = int(input())

a += a % 2

print(tuple(x for x in range(a, a\*10 + 2, 2)))

**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.

**Листинг К13\_1/2.py:**

a = int(input())

b = int(input())

for k in range(a, b, -1):

print(k)

**Задание 3.** (Л.Б.) В каждой строке файла хранится информация о пунктах и их координатах относительно некоторого центра.

Требуется

1. Прочесть 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

Пример входного файла

Москва 0 0

Ивантеевка 20 15

Щёлково 10 30

Пушкино 15 5

**Листинг К13\_1/3.py:**

import math

def sortCities(cities):

l = len(cities)

for i in range(l-1):

for j in range(l-i-1):

length1 = cities[j][1]\*\*2 + cities[j][2]\*\*2

length2 = cities[j+1][1]\*\*2 + cities[j+1][2]\*\*2

if length1 > length2:

cities[j+1], cities[j] = cities[j], cities[j+1]

cities = []

for line in open("data.txt"):

city, x, y = line.split()

x = int(x)

y = int(y)

cities.append((city, x, y))

\_max = [(0,0), (0,0), 0]

equals = []

l = len(cities)

for i in range(l-1):

for j in range(i+1, l):

x1, x2 = cities[i][1], cities[j][1]

y1, y2 = cities[i][2], cities[j][2]

l1 = math.sqrt(x1\*\*2 + y1\*\*2)

l2 = math.sqrt(x2\*\*2 + y2\*\*2)

if l1 == l2:

equals.append(((x1,y1), (x2, y2)))

length = math.sqrt((x1-x2)\*\*2 + (y1-y2)\*\*2)

if length > \_max[2]:

\_max[0] = (x1, y1)

\_max[1] = (x2, y2)

\_max[2] = length

print("Diameter: " ,\_max[0], \_max[1])

print("Equals: ", \*equals)

sortCities(cities)

print("Sorted by length from the center: ", \*cities)

## Техника работы с файлами.

**Задание 1**. http://ptaskbook.com/ru/tasks/text.php

Text5. Дана строка S и текстовый файл. Добавить строку S в конец файла.

**Листинг К14\_1/1.py:**

s = input() + '\n'

f = open("data.txt", "r+")

fileText = []

#get to the end of file

for line in f:

...

f.write(s)

f.close()

**Задание 2.** http://ptaskbook.com/ru/tasks/text.php

Text12. Дана строка S и текстовый файл. Заменить в файле все пустые строки на строку S.

**Листинг К14\_1/2.py:**

s = input() + '\n'

f = open("data.txt", "r+")

fileText = []

for line in f:

if line != '\n':

fileText.append(line)

else:

fileText.append(s)

f.close()

f = open("data.txt", "r+")

for line in fileText:

f.write(line)

f.close()

**Задание 3**. http://ptaskbook.com/ru/tasks/text.php

Text20. Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.

**Листинг К14\_1/3.py:**

import re

f = open("data.txt", "r+")

fileText = []

for line in f:

newLine = re.sub(r' +', r' ', line)

fileText.append(newLine)

f.close()

f = open("data.txt", "w+")

for line in fileText:

f.write(line)

f.close()

**Задание 4.** http://ptaskbook.com/ru/tasks/text.php

Text44. Дан текстовый файл, каждая строка которого изображает целое число, дополненное слева и справа несколькими пробелами. Вывести количество этих чисел и их сумму.

**Листинг К14\_1/4.py:**

f = open("data.txt", "r")

\_sum = 0

for line in f:

n = int(line.strip())

print(n)

\_sum += n

print(\_sum)

**Задание 5**. http://ptaskbook.com/ru/tasks/text.php

Text53. Дан текстовый файл. Создать символьный файл, содержащий все знаки препинания, встретившиеся в текстовом файле (в том же порядке).

**Листинг К14\_1/5.py:**

f = open("data.txt", "r")

out = open("output.txt", "w")

p\_marks = ['.', ',', '?', '!', ':', ';']

for line in f:

for c in line:

if c in p\_marks:

out.write(c + ' ')

f.close()

out.close()

**Задание 6**. (Л.Б.)

При разработке курсовых проектов студентами 3 курса программистов ККМТ выбираются различные направления, например, "графика", "базы данных".. и предпочтения по языкам и средам "Си++", "Delphi"...

В каждой строке текстового файла хранятся следующие сведения о курсовых проектах:

Фамилия Имя Отчество; Группа; Год; Тема; Направления (список через запятую);

Языки и среды (список через запятую)

Например,

Иванов Иван Иванович;П1-21;2023;Картинки в базе;графика;Pascal,Lazarus

...

Программа должна читать входной файл и выдавать на экран ответы на вопросы

1. Какое направление встречается чаще всего

2. Какие языки и среды появились в дипломах в 2017 г.

**Листинг К14-2/1.py:**

def getPopularCourses(students):

coursesDict = {}

result = []

for student in students:

courses = [x.strip() for x in student[3].split(',')]

for course in courses:

if course not in coursesDict:

coursesDict.update({course : 1})

else:

coursesDict[course] += 1

\_max = 1

for course in coursesDict:

if \_max < coursesDict[course]:

\_max = coursesDict[course]

for course in coursesDict:

if coursesDict[course] == \_max:

result.append(course)

return result

students = []

f = open("data.txt", "r")

for line in f:

students.append(line.split(';'))

popularCourses = getPopularCourses(students)

print("Чаще всего встречается: ")

for course in popularCourses:

print(course)

coursesIn2017 = set()

for student in students:

if student[2] == "2017":

for course in student[4].split(','):

coursesIn2017.add(course.strip())

print("Среды и языки 2017 года: ")

for elem in coursesIn2017:

print(elem)

## Техника работы с модулями.

**Задание 1.** Класс deque() модуля collections в Python.

**Листинг К15\_1/1.py:**

from collections import deque

dq = deque("abc") #deque(['a', 'b', 'c'])

dq.append("d") #deque(['a', 'b', 'c', 'd'])

dq.appendleft("k") #deque(['k', 'a', 'b', 'c', 'd'])

dq1 = dq.copy() #deque(['k', 'a', 'b', 'c', 'd'])

dq1.clear() #deque([])

dq.append("a") #deque(['k', 'a', 'b', 'c', 'd', 'a'])

dq.count("a") #2

dq.extend("abc")#deque(['k', 'a', 'b', 'c', 'd', 'a', 'a', 'b', 'c'])

dq.extendleft("cd") #deque(['d', 'c', 'k', 'a', 'b', 'c', 'd', 'a', 'a', 'b', 'c'])

dq.pop() #'c'

dq.popleft() #'d'

dq.reverse() #deque(['b', 'a', 'a', 'd', 'c', 'b', 'a', 'k', 'c'])

dq.rotate(2) #deque(['k', 'c', 'b', 'a', 'a', 'd', 'c', 'b', 'a'])

**Задание 2**. Класс Counter() модуля collections в Python.

**Листинг К15\_1/2.py:**

from collections import Counter

cnt = Counter("aaaabbbccd") #Counter({'a': 4, 'b': 3, 'c': 2, 'd': 1})

dict(cnt) #{'a': 4, 'b': 3, 'c': 2, 'd': 1}

list(cnt.elements())#['a', 'a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'd']

cnt.most\_common(3) #[('a', 4), ('b', 3), ('c', 2)]

cnt1 = Counter(a=3, b=2, c=1)

cnt.subtract(cnt1) # Counter({'a': 1, 'b': 1, 'c': 1, 'd': 1})

cnt1.update(Counter("aabc")) # Counter({'a': 5, 'b': 3, 'c': 2})

print(cnt1['a']) #5

del(cnt1['b']) #Counter({'a': 5, 'c': 2})

**Задание 3**. Класс defaultdict() модуля collections в Python.

**Листинг К15\_2/1.py:**

from collections import defaultdict

# Подсчёт кол-ва вхождений слов в тексте

text = "Some text Some text Some text text text text"

d = defaultdict(int)

for word in text.split():

d[word] += 1

print(d.items()) #dict\_items([('Some', 3), ('text', 6)])

a = [("Товар1", "Много"), ("Товар2", "Мало")]

d = defaultdict(lambda: "Отсутствует")

for k, v in a:

d[k] = v

for i in range(1, 4):

print("Товар"+str(i)+":" , d["Товар"+str(i)])

#Товар1: Много

#Товар2: Мало

#Товар3: Отсутствует

**Задание 4**. Класс OrderedDict() модуля collections в Python.

**Листинг К15\_2/2.py:**

from collections import OrderedDict

# Упорядоченный вывод таблицы Хаффмана(при использовании обычного словаря

# порядок элементов случайный)

h\_table = [('a', '01'), ('b', '10'), ('c', '11'), ('d', '101')]

d = OrderedDict()

for k,v in h\_table:

d.update({k:v})

#Упорядоченный вывод

for elem in d:

print(elem + ': ', d[elem])

# При декодировании удобнее использовать словарь

# Например: for c in s: print(d[c], end=’’)

**Задание 5**. Функция argv модуля sys в Python.

**Листинг К15\_3/1.py:**

import sys

args = sys.argv

args[0] # Путь к файлу

args[1:] # Пользовательские аргументы

**Задание 6.** Имя используемой OS.

**Листинг К15\_3/2.py:**

import sys

if sys.platform.startswith('linux'):

# специфические операции для linux

...

elif sys.platform.startswith('win32'):

ver = sys.getwindowsversion();

print(ver.major) # для меня: 10

# специфические операции для windows

...

**Задание 7.** Различные сведения о версии Python.

**Листинг К15\_3/3.py:**

import sys

#sys.version\_info(major=3, minor=8, micro=0, releaselevel='final', serial=0)

print(sys.version\_info)

# hexversion гарантированно увеличивается

if sys.hexversion >= 0x030502F0: #3.5.2

# используйте дополнительные функции

...

else:

# используйте альтернативную реализацию

# или предупредите пользователя

...

**Задание 8.** Каталоги и пути интерпретатора Python.

**Листинг К15\_3/4.py:**

import sys

print(sys.prefix) # Специфичный для программы каталог Python

print(sys.executable) # Путь исполняемого файла интерпретатора

**Задание 9.** Объекты stdin, stdout, stderr модуля sys в Python.

**Листинг К15\_3/5.py:**

import sys

stdin = sys.stdin

stdout = sys.stdout

stderr = sys.stderr

sys.stdin = open("in.txt")

sys.stdout = open("out.txt")

sys.stderr = open("err.txt")

# теперь все данные будут извлекаться из файлов

# и записываться в файлы

# чтение из in.txt

line = input()

#запись в out.txt

print(line)

**Задание 10**. Функция exit() модуля sys в Python.

**Листинг К15\_3/6.py:**

import sys

# перехват SystemExit который вызывается при exit()

try:

print("Пора выходить")

sys.exit()

except SystemExit:

print("Нет, мы остаемся")

**Задание 11**. Работа с файлами в Python с помощью модуля OS.

**Листинг К15\_4/1.py:**

import os

# текущая директория

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

# создание папки

if not os.path.isdir("New Folder"):

os.mkdir("New Folder")

# изменение директории

os.chdir("New Folder")

# создание вложенных папок

os.chdir("..")

os.makedirs("Folders/Folder1/Folder2")

# создание файлов

os.chdir("New Folder")

newFile = open("New\_File.txt", "w")

newFile.write("hehe")

newFile.close()

# переименование файлов

os.rename("New\_File.txt", "Old File.txt")

# перемещение файлов

os.replace("Old File.txt", "../Folders/Folder1/Old File.txt")

# список файлов и директорий

print(os.listdir())

# удаление файлов

os.chdir("..")

os.remove("Folders/Folder1/Old File.txt")

# удаление директорий

os.rmdir("New Folder")

os.removedirs("Folders/Folder1/Folder2")

# получение информации о файлах

f = open("tmp.txt", "w")

f.write("hehe")

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

f.close()

os.remove("tmp.txt")

## Техника работы с классами.

**Задание 1.** Классы и объекты Python.

1. Создание класса

2. Создание объекта

3. Функция init

4. Методы объектов

5. Параметр self

6. Изменение свойств объекта

7. Удалить свойства объекта

8. Удаление объектов

**Листинг К16\_1/1.py:**

class Enemy:

def \_\_init\_\_(self, dmg, hp):

self.damage = dmg

self.health = hp

def doDamage(self):

print("Emeny attacks(damage = " + str(self.damage) + ")")

enemy = Enemy(10, 20)

enemy.doDamage() #Emeny attacks(damage = 10)

enemy.damage = 15

enemy.doDamage() #Emeny attacks(damage = 15)

del enemy.damage

#enemy.doDamage() #Error

del enemy

**Задание 2**. Работа с классами в Python.

1. Создание классов

2. Создание экземпляров класса

3. Доступ к атрибутам

4. Встроенные атрибуты класса

5. Уничтожение объектов (сбор мусора)

**Листинг К16\_2/1.py:**

class Enemy:

"""Enemy class"""

enemy\_cnt = 0

def \_\_init\_\_(self, dmg, hp):

self.damage = dmg

self.health = hp

Enemy.enemy\_cnt += 1

def \_\_del\_\_(self):

print("Enemy dead")

def doDamage(self):

print("Eneny attacks(damage = " + str(self.damage) + ")")

enemy1 = Enemy(10, 20)

enemy1.doDamage() #Eneny attacks(damage = 10)

enemy1.damage = 15

enemy1.doDamage() #Eneny attacks(damage = 15)

enemy2 = Enemy(15, 25)

enemy2.doDamage() #Eneny attacks(damage = 15)

print(Enemy.enemy\_cnt)

getattr(enemy2, "damage") # 15

hasattr(enemy2, "damage") # true

setattr(enemy2, "damage", 16)

enemy2.doDamage() #Eneny attacks(damage = 16)

delattr(enemy2, "damage")

print("Enemy.\_\_doc\_\_", Enemy.\_\_doc\_\_)

print("Enemy.\_\_name\_\_", Enemy.\_\_name\_\_)

print("Enemy.\_\_module\_\_", Enemy.\_\_module\_\_)

print("Enemy.\_\_bases\_\_", Enemy.\_\_bases\_\_)

print("Enemy.\_\_dict\_\_", Enemy.\_\_dict\_\_)

del enemy2 #Enemy dead

**Задание 3**. Работа с классами в Python.(продолжение)

1. Наследование класса

2. Переопределение методов

3. Популярные базовые методы

4. Приватные методы и атрибуты класса

**Листинг К16\_3/1.py:**

class Enemy:

"""Enemy class"""

enemy\_cnt = 0

\_\_damage = 0

\_\_health = 0

def getDmg(self):

return self.\_\_damage

def setDmg(self, dmg):

self.\_\_damage = dmg

def getHp(self):

return self.\_\_health

def setHp(self, hp):

self.\_\_health = hp

def \_\_init\_\_(self, dmg, hp):

self.setDmg(dmg)

self.setHp(hp)

Enemy.enemy\_cnt += 1

def \_\_del\_\_(self):

print("Enemy dead")

def doDamage(self):

print("Eneny attacks(damage = " + str(self.getDmg()) + ")")

def \_\_add\_\_(self, other):

return Enemy(self.getDmg() + other.getDmg(), self.getHp() + other.getHp())

class FireEnemy(Enemy):

def doDamage(self):

print("Enemy attacks with fire (dmg = " + str(self.getDmg()) + ")")

enemy1 = Enemy(10, 20)

enemy1.doDamage() #Eneny attacks(damage = 10)

enemy1.setDmg(15)

enemy1.doDamage() #Eneny attacks(damage = 15)

enemy2 = Enemy(15, 25)

enemy2.doDamage() #Eneny attacks(damage = 15)

enemy3 = FireEnemy(30, 40)

enemy3.doDamage()

enemy4 = enemy1 + enemy2

print(enemy4.getDmg(), enemy4.getHp()) #30 45

print(Enemy.enemy\_cnt)

**Задание 4.** Работа с классами в Python.(продолжение)

1. Придумать собственный класс

2. Неформально описать функционал класса

3. Реализовать класс в модуле

4. Разработать скрипт для демонстрации работы с классом (импортировать модуль,создать экземпляры, вызвать методы)

**Листинг К16\_4/MyClasses/Enemy.py:**

class Enemy:

"""Enemy class"""

enemy\_cnt = 0

\_\_damage = 0

\_\_health = 0

def getDmg(self):

return self.\_\_damage

def setDmg(self, dmg):

self.\_\_damage = dmg

def getHp(self):

return self.\_\_health

def setHp(self, hp):

self.\_\_health = hp

def \_\_init\_\_(self, dmg, hp):

self.setDmg(dmg)

self.setHp(hp)

Enemy.enemy\_cnt += 1

def \_\_del\_\_(self):

print("Enemy dead")

def doDamage(self):

print("Eneny attacks(damage = " + str(self.getDmg()) + ")")

def \_\_add\_\_(self, other):

return Enemy(self.getDmg() + other.getDmg(), self.getHp() + other.getHp())

class FireEnemy(Enemy):

def doDamage(self):

print("Enemy attacks with fire (dmg = " + str(self.getDmg()) + ")")

**Листинг К16\_4/script.py:**

import MyClasses.Enemy as myEnemy

if \_\_name\_\_ == "\_\_main\_\_":

e = myEnemy.Enemy(10, 20)

e.doDamage() #Eneny attacks(damage = 10)

e1 = myEnemy.FireEnemy(15, 20)

e1.doDamage() #Enemy attacks with fire (dmg = 15)

e1.setDmg(20)

e1.doDamage() #Enemy attacks with fire (dmg = 20)

del e #Enemy dead

del e1 #Enemy dead