

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:

Планкин И.И.

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

Гусятинер Л.Б.

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

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

Королев, 2020

Содержание

[**2.1. Установка интерпретатора Python 3** 3](#_Toc58934711)

[**2.2. Настройка окружения** 5](#_Toc58934712)

[**3.1. Техника работы в командной строке** 8](#_Toc58934713)

[**3.2. Техника работы в среде IDLE** 9](#_Toc58934714)

[**4.1 Техника работы с линейными программами.** 11](#_Toc58934715)

[**4.2. Техника работы с разветвляющимися программами** 13](#_Toc58934716)

[**5.2. Техника работы с циклическими программами \_ цикл while.** 15](#_Toc58934717)

[**6.1. Техника работы с числами.** 17](#_Toc58934718)

[**6.2. Техника работы с числами** 18](#_Toc58934719)

[Модуль math. 18](#_Toc58934720)

[Список функций модуля math. 19](#_Toc58934721)

[**Примеры применения модуля math и его функций:** 20](#_Toc58934722)

[Модуль cmath. 22](#_Toc58934723)

[Список функций модуля cmath. 22](#_Toc58934724)

[**Примеры применения модуля cmath и его функций:** 23](#_Toc58934725)

[**7.1. Техника работы со строками** 24](#_Toc58934726)

[**7.2. Техника работы со строками.** 27](#_Toc58934727)

[**8.1. Техника работы со списками** 28](#_Toc58934728)

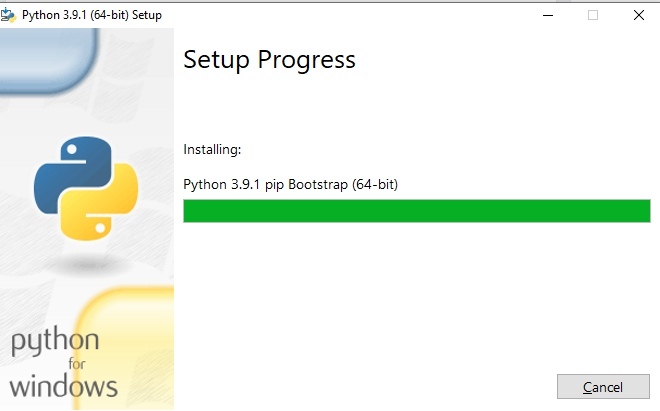
[**8.2. Техника работы со списками** 29](#_Toc58934729)

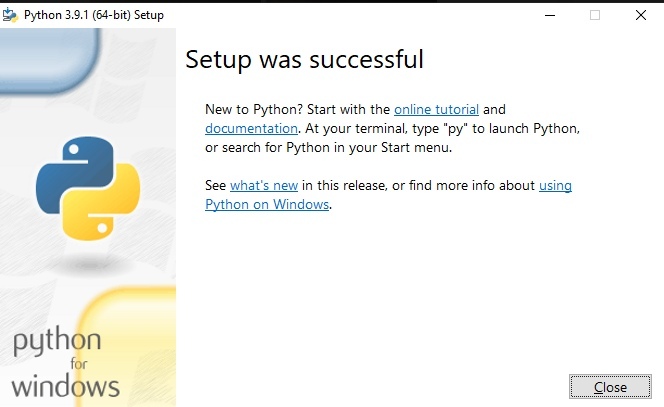
[**9.2. Техника работы с циклом for и генераторами списков.** 31](#_Toc58934730)

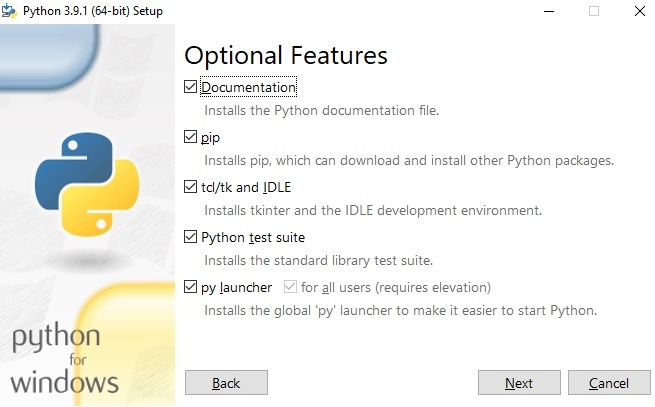
## **2.1. Установка интерпретатора Python 3**

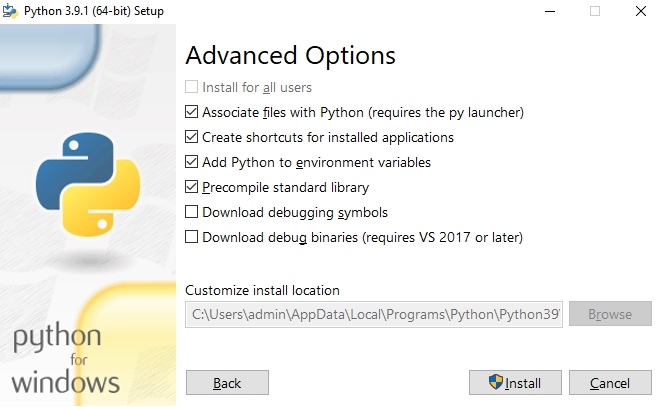
1. Установить актуальную версию Python на Windows, Linux и подготовить  
руководство (видео / презентацию / веб-страницы)

2. Быть готовым продемонстрировать полностью процесс установки с пояснениями.



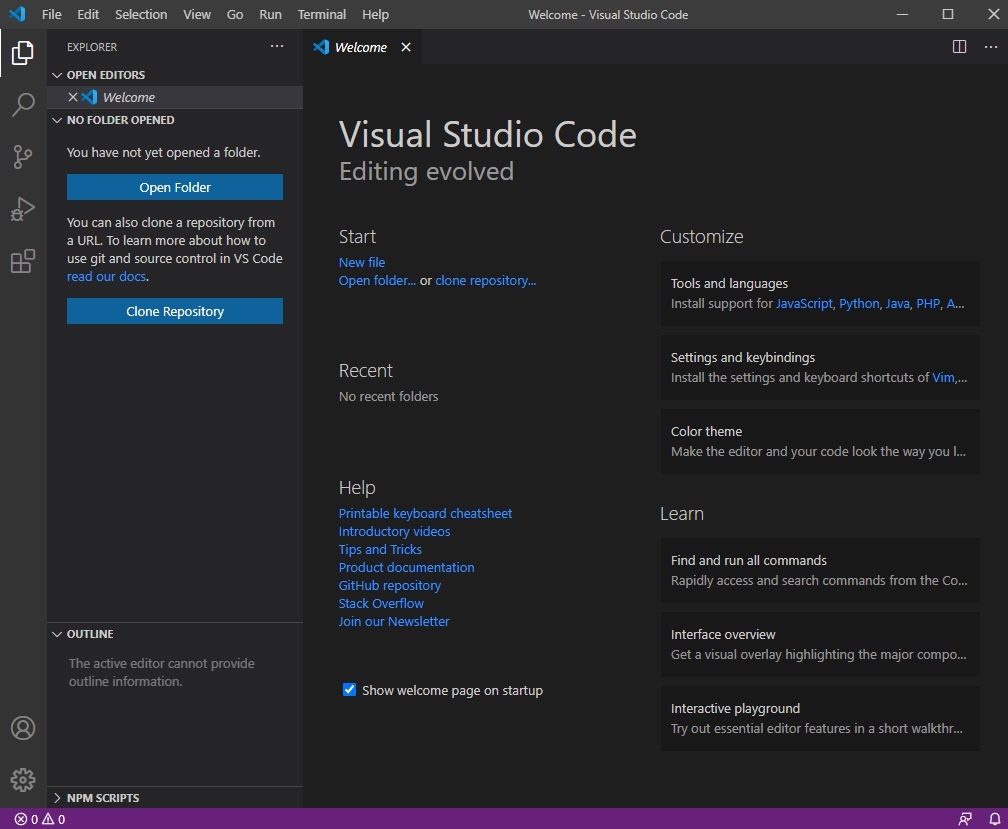


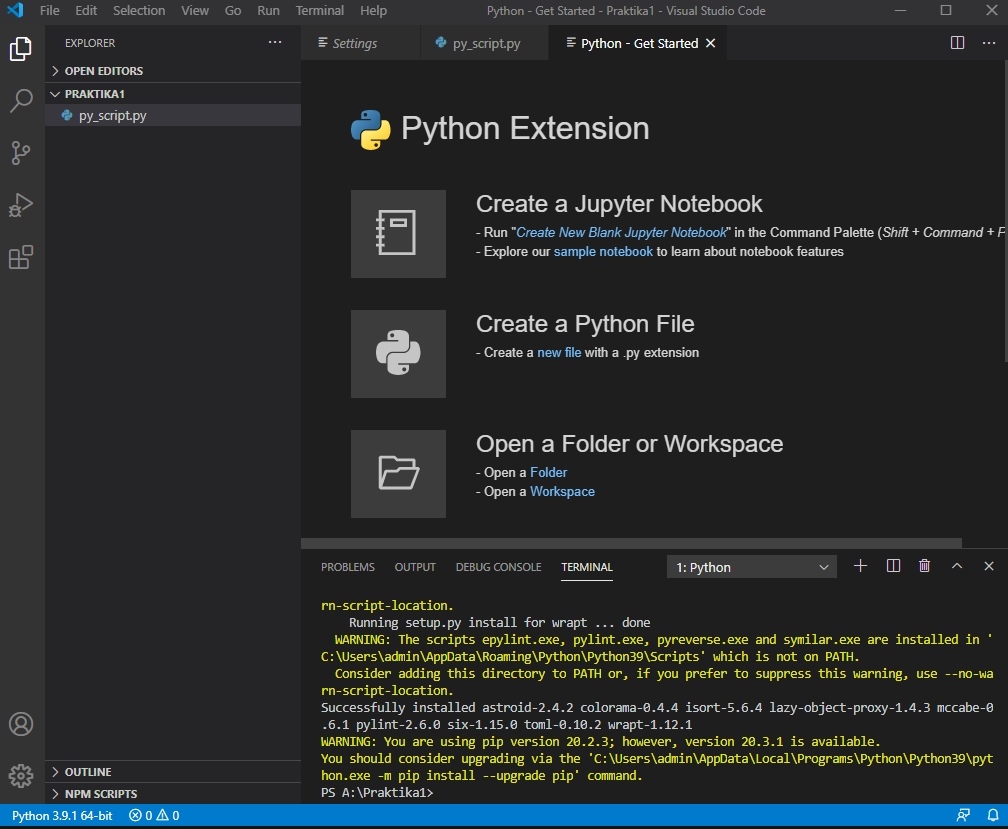


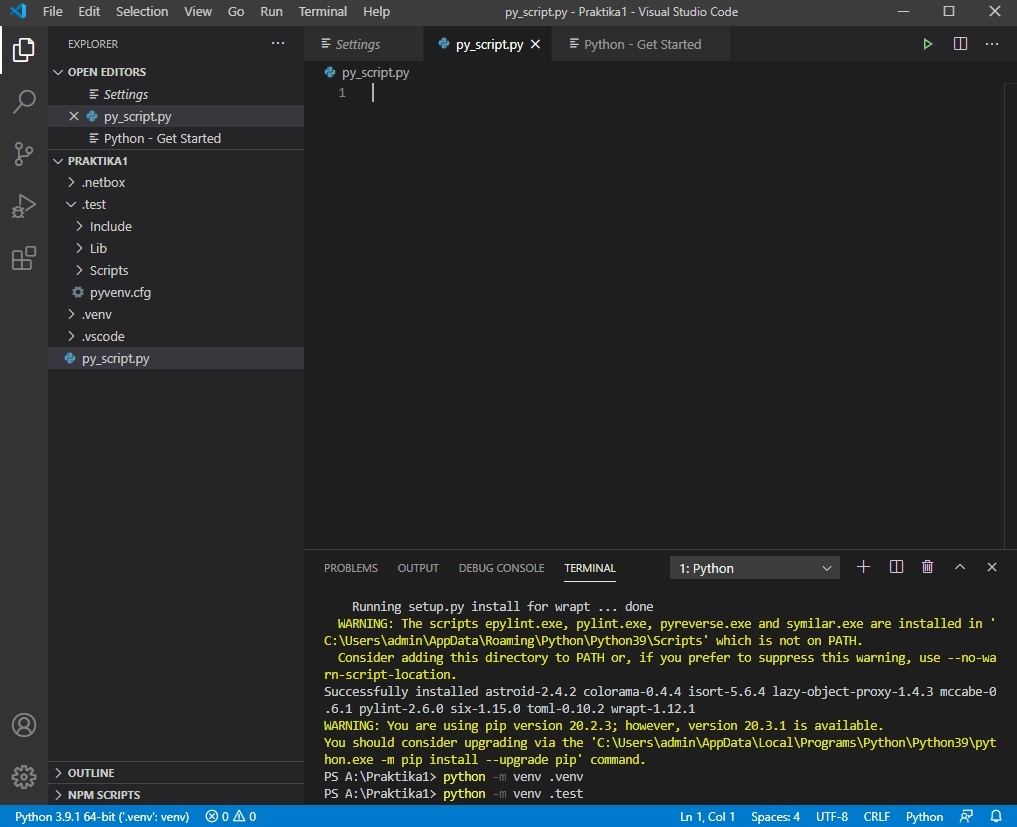


## **2.2. Настройка окружения**

УП.01 Учебная практика по модулю ПМ.01  
Консультация 2-2. Настройка окружения интерпретатора Python 3  
Составитель: Гусятинер Л.Б., 24.11.2020, МГОТУ ККМТ, П1-18, П2-18  
Задание.  
1. Продемонстрировать процесс настройки окружения

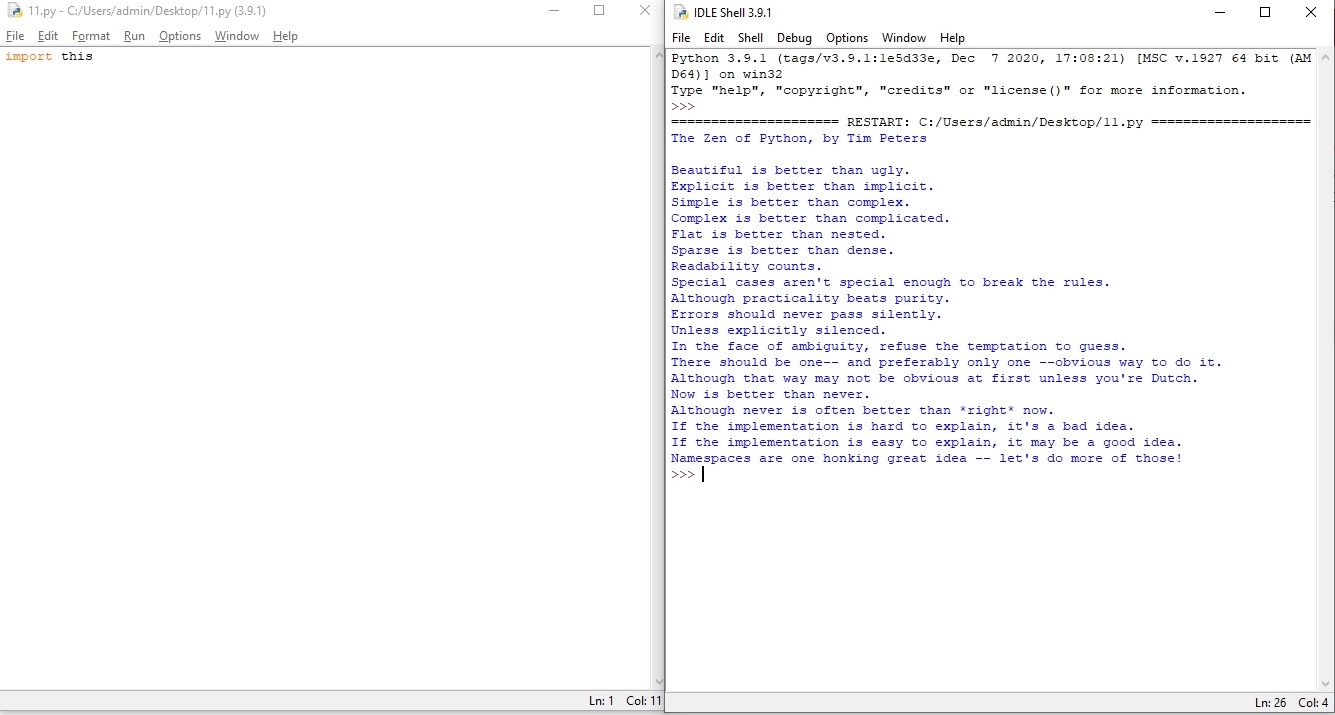


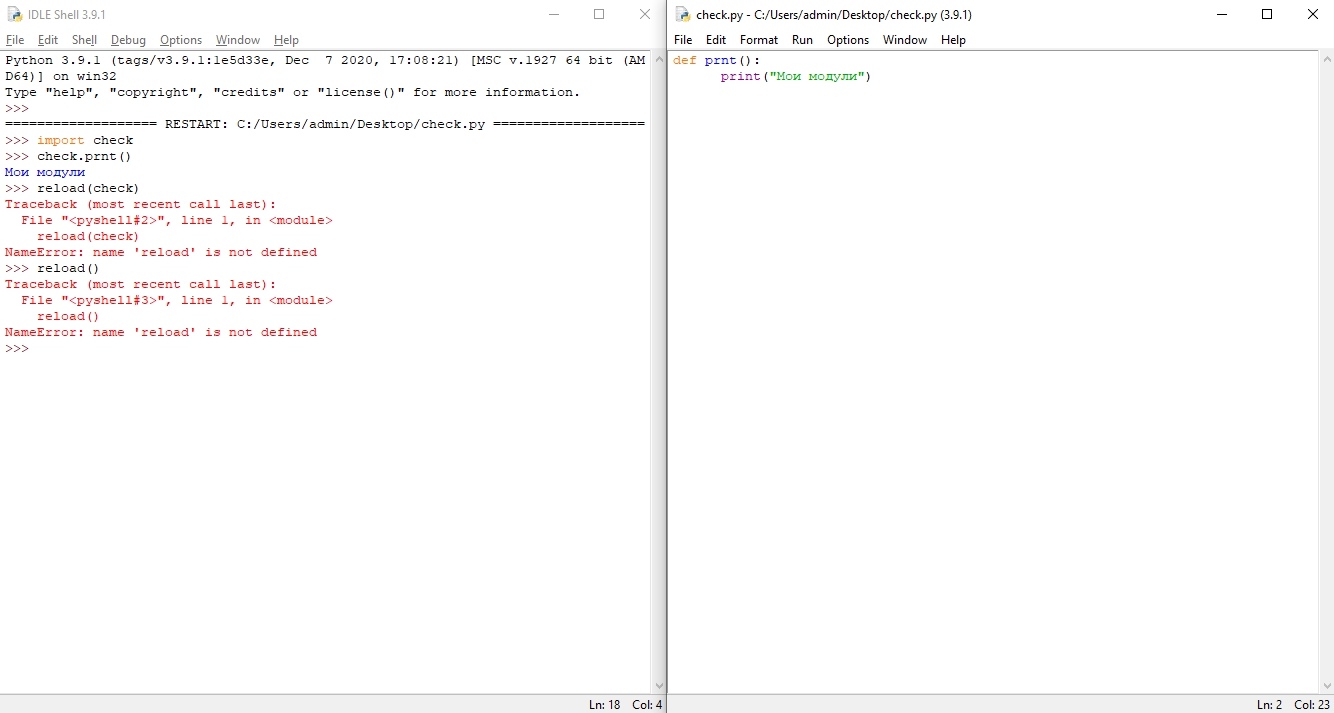




## **3.1. Техника работы в командной строке**

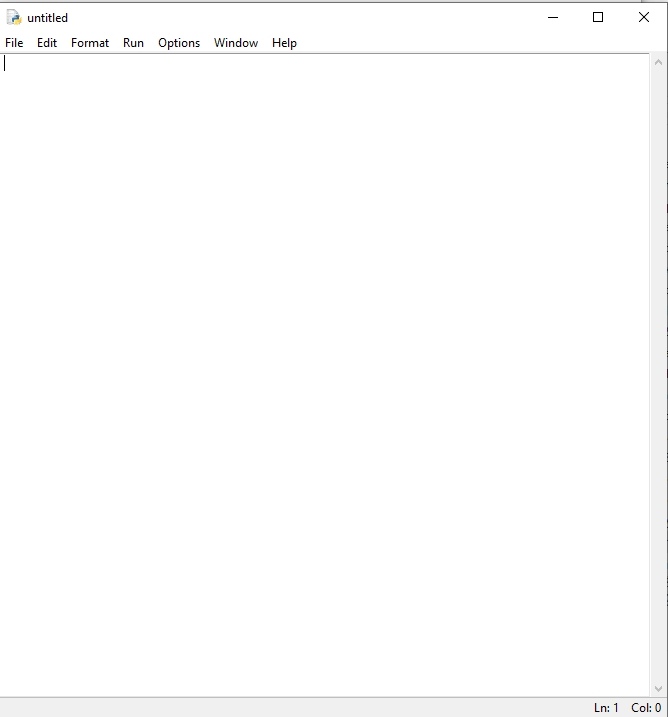
УП.01 Учебная практика по модулю ПМ.01  
Консультация 3. Техника работы в командной строке и среде IDLE  
3-1.  
Составитель: Гусятинер Л.Б., 24.11.2020, МГОТУ ККМТ, П1-18, П2-18  
Задание.  
1. Продемонстрировать работу в командной строке, включая  
- создание файла с кодом  
- запуск  
- import  
- reload  
- отработку ошибок

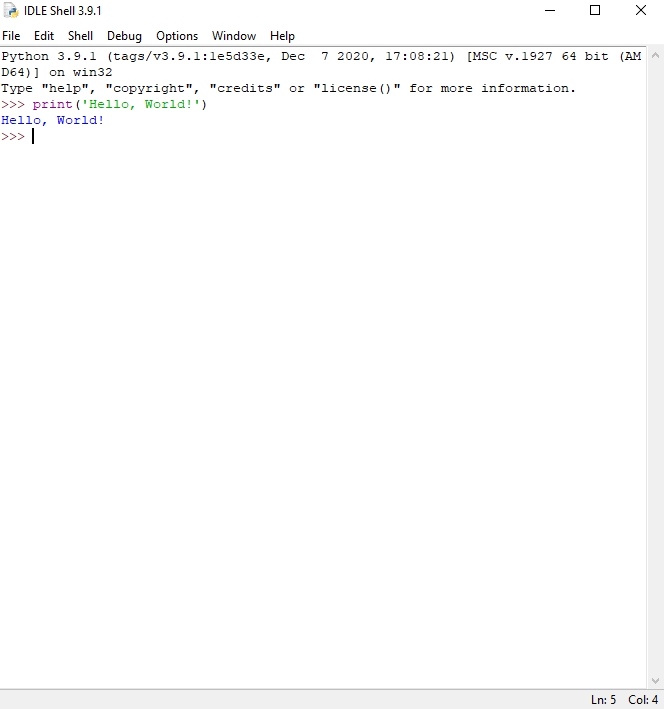


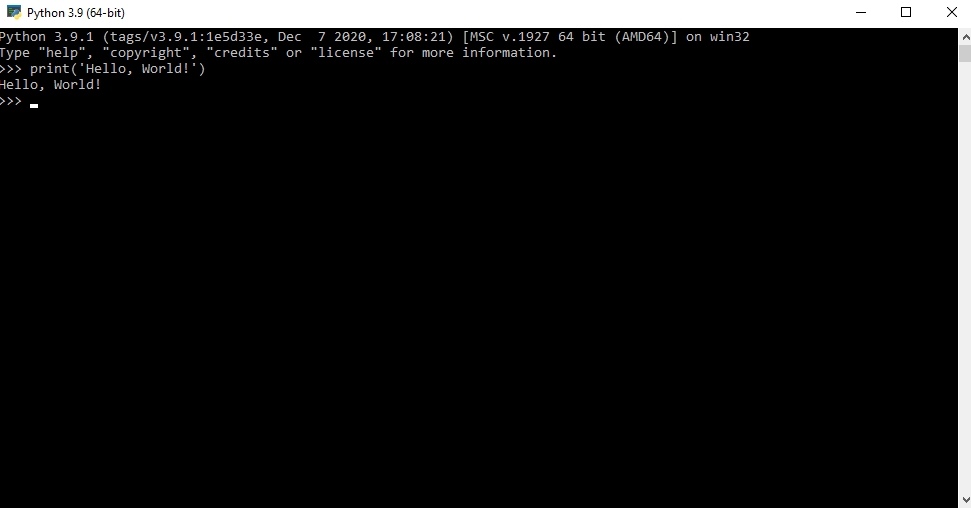


## **3.2. Техника работы в среде IDLE**

УП.01 Учебная практика по модулю ПМ.01  
Консультация 3. Техника работы в командной строке и среде IDLE  
3-1.  
Составитель: Гусятинер Л.Б., 24.11.2020, МГОТУ ККМТ, П1-18, П2-18  
Задание.  
1. Показать работу в оболочке IDLE как в самой среде, так и путём запуска  
файлов.  
2. Показать умение работать с меню.







## **4.1 Техника работы с линейными программами.**

- print

Функция print() в Python, печатает объект.

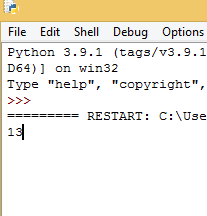
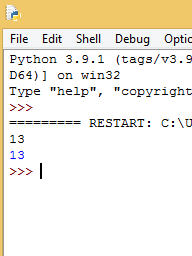
- input

Функция input() в Python, ввод данных с клавиатура.

a = input()

print(a)

Пример рабочей программы.



Русуноки 1 и 2. Демонстрация работы программы.

- форматная строка и метод формат

print('Hello, {0}, {1}, {2}, {3}!'.format(input(), input(),input(),input()))

Пример кода на формат.

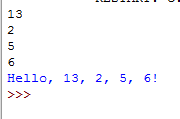


Рисунок 3. Демонстрация работы программы.

## **4.2. Техника работы с разветвляющимися программами**

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))

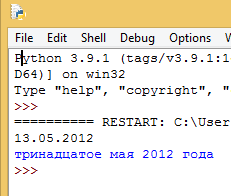


Рисунок 4. Демонстрация работы программы.

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

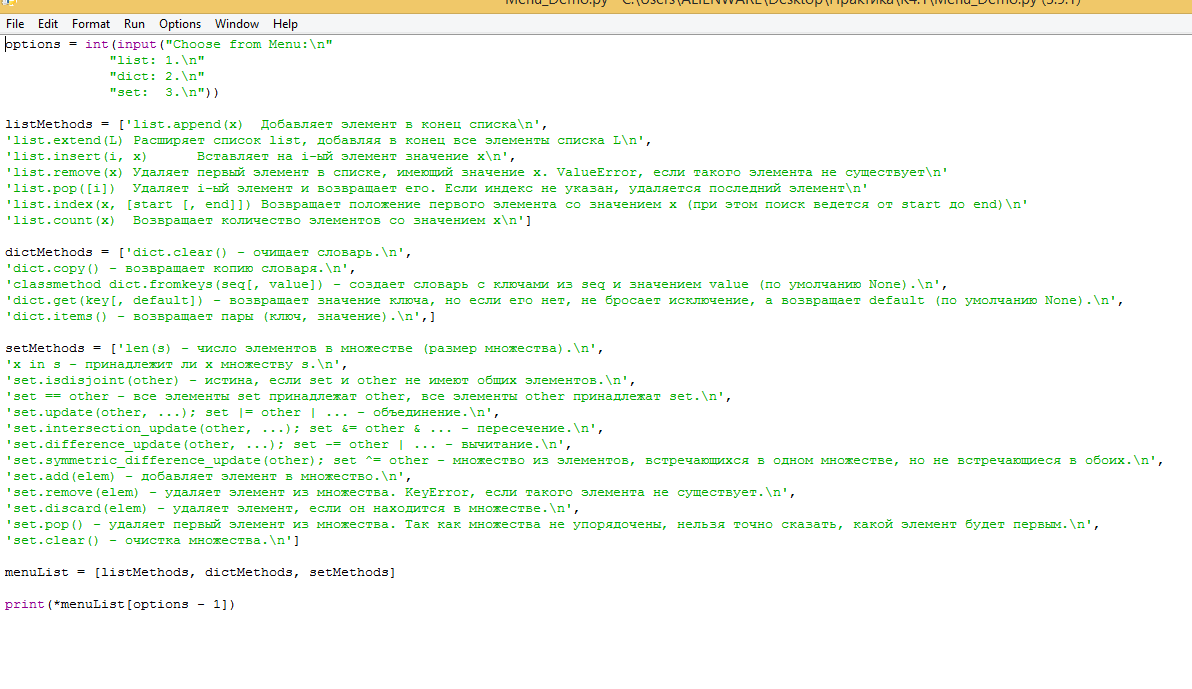


Рисунок 5. Рабочая программа.

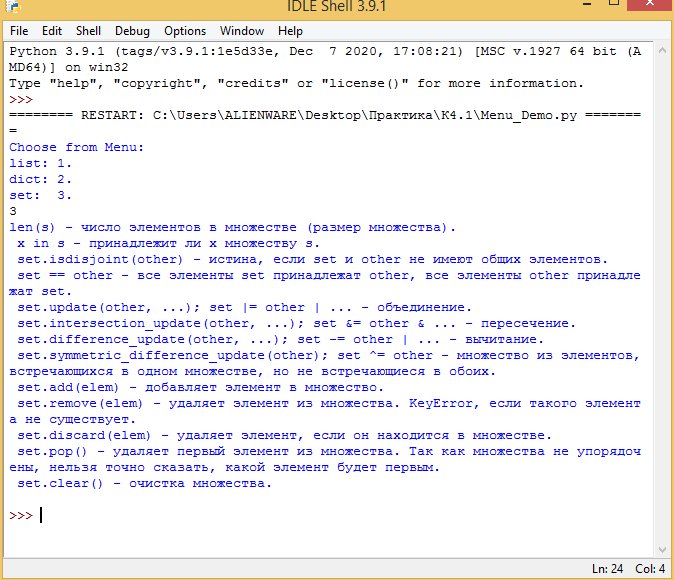


Рисунок 6. Демонстрация работы программы.

## **5.2. Техника работы с циклическими программами \_ цикл while.**

**Задача №1**.

Вычислить значение sin(x) с точностью до epsilon при помощи разложения в ряд

Построить блок-схему.

def y(b):

def foo(x):

if x < 1:

return x\*\*2 - 2

elif x == 1:

return 1 + x

else:

return math.sqrt(b\*\*3 + x\*\*2)

return foo

#test

y1 = y(1.25)

for a in map(y1, [0.0, 0.5, 1.0]):

print(a)

**Вывод:**

-2.0

-1.75

2.0

**Задача №2.**

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

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

a = int(input())

b = 0

while a != 0:

b = b + a

a = int(input())

print(b)

**Входные данные:**

5

-3

8

4

0

**Вывод:**

14

**Задание 3.**

Разработать программу для нахождения наибольшего общего делителя

#Алгоритм Евклида для нахождения НОД

a = int(input())

b = int(input())

while a!=0 and b!=0:

if a > b:

a = a % b

else:

b = b % a

print (a+b)

**Входные данные:**

12

64

**Вывод:**

4

**Задача 4.**

С использованием результата задания 2 разработать программу для нахождения наименьшего

общего кратного

a = int(input())

b = int(input())

i = min(a, b)

while True:

if i%a==0 and i%b==0:

break

i += 1

print(i)

**Входные данные:**

24

124

**Вывод:**

744

**Задание 5.**

Напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...

(число повторяется столько раз, чему равно).

На вход программе передаётся неотрицательное целое число n — столько элементов

последовательности должна отобразить программа.

На выходе ожидается последовательность чисел, записанных через пробел в одну строку.

Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.

a=int(input())

b=''

for i in range(a+1):

b = b + (str(i) + ' ')\*i

print(b[0:a\*2])

**Входные данные:**

7

**Вывод:**

1 2 2 3 3 3 4

## **6.1. Техника работы с числами.**

**Задача №1.** Составить и выполнить по 3 примера использования модулей для работы

с дробными числами (fractions), для точных вычислений (decimal).

**Задача №2.** Подготовить инструкцию по использованию модулей fractions, decimal.

#Decimal обеспечивает поддержку правильного округления десятичной арифметики с плавающей точкой.

from decimal import Decimal

number1 = Decimal(input())

number2 = Decimal(input())

print(number1 + number2)

#Округление осуществляется с помощью метода quantize().

#В качестве первого аргумента – объект Decimal, указывающий на формат округления:

from decimal import Decimal

number = Decimal(input())

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

# округление до 3 чисел в дробной части

print(number.quantize(Decimal('1.00')))

# округление до 2 чисел в дробной части

print(number.quantize(Decimal('1.0')))

# округление до 1 числа в дробной части

from fractions import Fraction

a = Fraction(input())

b = Fraction(input())

print(a + b)

#Fraction.limit\_denominator(max\_denominator=1000000)

#- ближайшее рациональное число со знаменателем не больше данного.

x = Fraction(input())

print(x.limit\_denominator

# **6.2. Техника работы с числами**

Если вы пишете программу для выполнения определенных задач, таких как изучение периодического движения или моделирования электрических цепей, вам нужно будет работать с тригонометрическими функциями, а также с комплексными числами. Хотя вы не можете использовать эти функции напрямую, вы можете получить к ним доступ, включив сначала два математических модуля. Эти модули являются [math](https://docs.python.org/3/library/math.html) и [cmath](https://docs.python.org/3/library/cmath.html).

Первый дает вам доступ к гиперболическим, тригонометрическим и логарифмическим функциям для действительных чисел, а последний позволяет работать с комплексными числами. В этом уроке я рассмотрю все важные функции, предлагаемые этими модулями. Если явно не указано, все возвращаемые значения - это float.

## Модуль math.

Модуль **Math** в Python обеспечивает доступ к некоторым популярным математическим функциям и константам, которые можно использовать в коде для более сложных математических вычислений. Библиотека является встроенным модулем Python, поэтому никакой дополнительной [установки через pip](https://python-scripts.com/how-to-install-modules-python) делать не нужно. В данной статье будут даны примеры часто используемых функций и констант библиотеки Math в Python.

* Первой важной математической константой является число Пи (π). Оно обозначает отношение длины окружности к диаметру, его значение 3,141592653589793.
* Число Эйлера (е) является основанием натурального логарифма. Оно также является частью **модуль Math** в Python.
* Библиотека Math в Python поставляется с функцией exp(), которую можно использовать для вычисления значения е. К примеру, ex — экспонента от х. Значение е равно 2.718281828459045.
* Арифметические [функции](https://python-scripts.com/functions-python) используются для представления чисел в различных формах и осуществления над ними математических операций.
* Модуль **math** в Python поддерживает все тригонометрические функции
* Python может конвертировать начальный тип числа в другой указанный тип.

## Список функций модуля math.

**math.ceil**(X) – округление до ближайшего большего числа.

**math.copysign**(X, Y) - возвращает число, имеющее модуль такой же, как и у числа X, а знак - как у числа Y.

**math.fabs**(X) - модуль X.

**math.factorial**(X) - факториал числа X.

**math.floor**(X) - округление вниз.

**math.fmod**(X, Y) - остаток от деления X на Y.

**math.frexp**(X) - возвращает мантиссу и экспоненту числа.

**math.ldexp**(X, I) - X \* 2i. Функция, обратная функции math.frexp().

**math.fsum**(последовательность) - сумма всех членов последовательности. Эквивалент встроенной функции sum(), но math.fsum() более точна для чисел с плавающей точкой.

**math.isfinite**(X) - является ли X числом.

**math.isinf**(X) - является ли X бесконечностью.

**math.isnan**(X) - является ли X NaN (Not a Number - не число).

**math.modf**(X) - возвращает дробную и целую часть числа X. Оба числа имеют тот же знак, что и X.

**math.trunc**(X) - усекает значение X до целого.

**math.exp**(X) - eX.

**math.expm1**(X) - eX - 1. При X → 0 точнее, чем math.exp(X)-1.

**math.log**(X, [base]) - логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.

**math.log1p**(X) - натуральный логарифм (1 + X). При X → 0 точнее, чем math.log(1+X).

**math.log10**(X) - логарифм X по основанию 10.

**math.log2**(X) - логарифм X по основанию 2.

**math.pow**(X, Y) - XY.

**math.sqrt**(X) - квадратный корень из X.

**math.acos**(X) - арккосинус X. В радианах.

**math.asin**(X) - арксинус X. В радианах.

**math.atan**(X) - арктангенс X. В радианах.

**math.atan2**(Y, X) - арктангенс Y/X. В радианах. С учетом четверти, в которой находится точка (X, Y).

**math.cos**(X) - косинус X (X указывается в радианах).

**math.sin**(X) - синус X (X указывается в радианах).

**math.tan**(X) - тангенс X (X указывается в радианах).

**math.hypot**(X, Y) - вычисляет гипотенузу треугольника с катетами X и Y (math.sqrt(x \* x + y \* y)).

**math.degrees**(X) - конвертирует радианы в градусы.

**math.radians**(X) - конвертирует градусы в радианы.

**math.cosh**(X) - вычисляет гиперболический косинус.

**math.sinh**(X) - вычисляет гиперболический синус.

**math.tanh**(X) - вычисляет гиперболический тангенс.

**math.acosh**(X) - вычисляет обратный гиперболический косинус.

**math.asinh**(X) - вычисляет обратный гиперболический синус.

**math.atanh**(X) - вычисляет обратный гиперболический тангенс.

**math.erf**(X) - функция ошибок.

**math.erfc**(X) - дополнительная функция ошибок (1 - math.erf(X)).

**math.gamma**(X) - гамма-функция X.

**math.lgamma**(X) - натуральный логарифм гамма-функции X.

**math.pi** - pi = 3,1415926...

**math.e** - e = 2,718281...

### **Примеры применения модуля math и его функций:**

# Импорт модуля math

import math

# Дробный номер

number=8.10

# выводим целую часть числа с округлением к большему

print("Верхний предел 8.10 это:",math.ceil(number))

# выводим целую часть числа с округлением к меньшему

print("Нижний предел 8.10 это:",math.floor(number))

**Вывод:**

Верхний предел 8.10 это: 9

Нижний предел 8.10 это: 8

# Импорт модуля math

import math

n = -8.10

# Вывод абсолютного значения числа

print(math.fabs(n))

**Вывод:**

8.1

**Пример:**

# Импорт модуля math

import math

number = 5

# вывод факториала числа

print("факториала числа", math.factorial(number))

**Вывод:**

факториала числа 120

## Модуль cmath.

Модуль cmath – предоставляет функции для работы с комплексными числами.

Сложные числа хранятся внутри с использованием прямоугольных или декартовых 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), а 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).

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).

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.

## Список функций модуля cmath.

**cmath.phase**(x) - возвращает фазу комплексного числа (её ещё называют аргументом). Эквивалентно math.atan2(x.imag, x.real). Результат лежит в промежутке [-π, π].

Получить модуль комплексного числа можно с помощью встроенной функции abs().

**cmath.polar**(x) - преобразование к полярным координатам. Возвращает пару (r, phi).

**cmath.rect**(r, phi) - преобразование из полярных координат.

**cmath.exp**(x) - ex.

**cmath.log**(x[, base]) - логарифм x по основанию base. Если base не указан, возвращается натуральный логарифм.

**cmath.log10**(x) - десятичный логарифм.

**cmath.sqrt**(x) - квадратный корень из x.

**cmath.acos**(x) - арккосинус x.

**cmath.asin**(x) - арксинус x.

**cmath.atan**(x) - арктангенс x.

**cmath.cos**(x) - косинус x.

**cmath.sin**(x) - синус x.

**cmath.tan**(x) - тангенс x.

**cmath.acosh**(x) - гиперболический арккосинус x.

**cmath.asinh**(x) - гиперболический арксинус x.

**cmath.atanh**(x) - гиперболический арктангенс x.

**cmath.cosh**(x) - гиперболический косинус x.

**cmath.sinh**(x) - гиперболический синус x.

**cmath.tanh**(x) - гиперболический тангенс x.

**cmath.isfinite**(x) - True, если действительная и мнимая части конечны.

**cmath.isinf**(x) - True, если либо действительная, либо мнимая часть бесконечна.

**cmath.isnan**(x) - True, если либо действительная, либо мнимая часть NaN.

**cmath.pi** - π.

**cmath.e** - e.

### **Примеры применения модуля cmath и его функций:**

import cmath

print(cmath.polar(complex(1.0, 1.0)))

print(cmath.phase(complex(1.0, 1.0)))

print(abs(complex(1.0, 1.0)))

**Вывод:**

(1.4142135623730951, 0.7853981633974483)

0.7853981633974483

1.4142135623730951

import cmath

print(cmath.sqrt(complex(25.0, 25.0)))

print(cmath.cos(complex(25.0, 25.0)))

**Вывод:**

(5.49342056733905+2.2754493028111367j)

(35685729345.58163+4764987221.458499j)

## **7.1. Техника работы со строками**

**Задача №1.** С клавиатуры вводятся строки, последовательность заканчивается точкой.  
Выведите буквы введенных слов в верхнем регистре, разделяя их пробелами.

b = []  
a = input()  
while a != '.':  
b.append(a.upper())  
a = input()  
for i in range(len(b)):  
print(' '.join(b[i]))

b делает пустой список, а ждет ввода данных, вайл а делает так, что мы будем добавлять слова в конец списка и их буквы будут увеличиваться пока не будет точки, фор делает так, что он проходит столько, сколько нам нужно слов( это 3 цикла) и делает пробелы между буквами в словах.

**Задача №2.**  
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.

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

s приимает на ввод данные, реплайс заменяет 1 заданное значение на другое, принт выводит данные

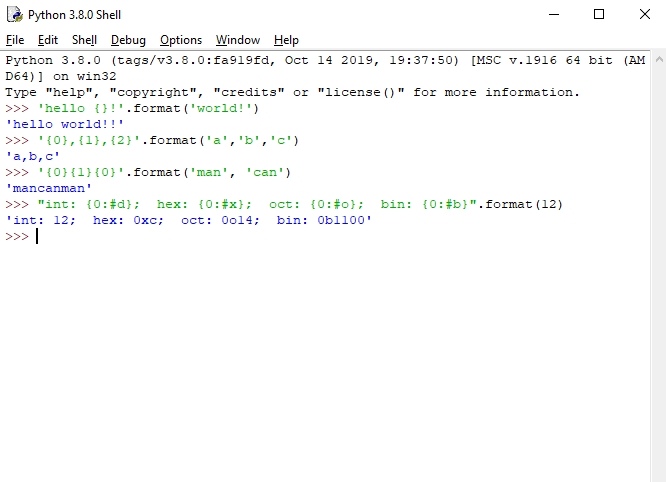
**Задача 3.**   
Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел,  
а затем сумму получившихся чисел.  
  
s = input()  
s = s.replace('.', ' ')  
print(s)  
s = s.split()  
summ = 0  
for x in s:  
x = int(x)  
summ += x  
print(summ)

**Входные данные:**192.168.0.1  
**Вывод:**  
192 168 0 1  
361  
s принимает на ввод данные, реплайс заменяет 1 заданное значение на другое, принт выводит данные s, после применяем split, это еще способ разбить строчки на части, фор делает так, чтобы посчитал все целочисленные значение потом выводит еще одни данные, который подсчитывает сумму этих чисел.

**Задача №4.**    
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  
a = []  
b = input()  
while b != '.':  
a.append(b)  
b = input()  
for i in a:  
if i[:2] == '!!':  
print('предупреждение')  
if i[:2] == '@@':  
print('ошибка')  
if i[:2] == '//':  
print('информация')  
if i[:2] == '\*\*':  
print('подробное сообщение')  
a создает пустой список, б вводит данные, вайл вводит слова в конец строчки до точки, фор печатает столько, сколько у нас всего строчек(без точки), ифы смотрят начало строк, если там есть заданные символы, то печатают те значение, которые мы вписали

## **7.2. Техника работы со строками.**

П.01 Учебная практика по модулю ПМ.01  
Консультация 7. Техника работы со строками  
7-2.  
Составитель: Гусятинер Л.Б., 27.11.2020, МГОТУ ККМТ, П1-18, П2-18  
  
Задание 1. Подготовить сравнительную инструкцию по использованию  
форматирования строк



## **8.1. Техника работы со списками**

**Здача №1** «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.

l = input().split(' ')

c = 0

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

if l[i-1] <= l[i] >= l[i+1]:

c += 1

print(c)

**Входные данные:**

1 2 1 3 4

**Вывод:**

1

**Задача №2** «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.

l = input().split(' ')

c = 0

for i in range(0, len(l)):

for j in range(0, len(l)):

if (l[i] == l[j]) and (i != j):

c += 1

print(c // 2)

**Входные данные:**

1 1 1 1 1

**Вывод:**

10

**Задача №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

## **8.2. Техника работы со списками**

**Здача №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.

import random

с = random.randrange(2,20)

b = [random.randrange(0,10) for i in range(с)]

print(b)

for i in range(1,с) :

for j in range(0,с-i) :

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

b[j], b[j+1] = b[j+1], b[j]

print(b)

**Вывод:**

[0, 3, 3, 8, 9, 5]

[0, 3, 3, 5, 8, 9]

**Задача №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.

import random

c = random.randrange(2,10)

b = [random.randrange(0,10) for i in range(c)]

print(b)

for i in range(0,c) :

max\_val = max(b[:c-i])

max\_idx = b.index(max\_val)

b[max\_idx], b[c-i-1] = b[c-i-1], b[max\_idx]

print(b)

**Вывод:**

[2, 4, 8]

[2, 4, 8]

[2, 4, 8]

[2, 4, 8]

**Задача №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).

import random

c = random.randrange(2,10)

b = [random.randrange(0,10) for i in range(c)]

print(b)

for i in range(1,c) :

a = False

x = b[i]

q = i - 1

while q >= 0 :

if x >= b[q] :

break

else :

b[q+1] = b[q]

a = True

q -= 1

if a and q >= -1:

b[q+1] = x

print(b)

**Вывод:**

[3, 9, 9, 6]

[3, 9, 9, 6]

[3, 9, 9, 6]

[3, 6, 9, 9]

## **9.2. Техника работы с циклом for и генераторами списков.**

**Задача №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

c = random.randrange(2,15)

b = [random.randrange(1,11) for i in range(c)]

a = b[1::2]

print(b)

print(len(a))

print(a)

**Вывод:**

[9, 1, 1, 6, 7, 4, 6, 4, 5, 5, 7, 9, 9, 3, 10, 9, 9]

17

[9, 10, 11, 17, 24, 28, 34, 38, 43, 48, 55, 64, 73, 76, 86, 95, 104]

**Задача №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

c = random.randrange(2,21)

b = [random.randrange(1,11) for i in range(c)]

a = b[0::2] + b[1::2]

print(b)

print(len(a))

print(a)

**Входные данные:**

5

6

**Вывод:**

9 8 7 9 3 3

9 2 8 3 5 2

6 3 9 9 4 3

9 8 8 7 8 3

7 5 5 4 7 4

**Задача №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

c = random.randrange(2,21)

b = [random.randrange(1,11) for i in range(c)]

a = []

a.append(b[0])

print(b)

for i in range(1,c) :

a.append(b[i] + a[i-1])

print(len(a))

print(a)

**Вывод:**

[9, 1, 1, 6, 7, 4, 6, 4, 5, 5, 7, 9, 9, 3, 10, 9, 9]

17

[9, 10, 11, 17, 24, 28, 34, 38, 43, 48, 55, 64, 73, 76, 86, 95, 104]

**Задача №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

R = int(input())

C = int(input())

matrix = []

for i in range(R):

с =[]

for j in range(C):

с.append(random.randrange(2,10))

matrix.append(с)

for i in range(R):

for j in range(C):

print(matrix[i][j], end = " ")

print()

**Входные данные:**

5

6

**Вывод:**

9 8 7 9 3 3

9 2 8 3 5 2

6 3 9 9 4 3

9 8 8 7 8 3

7 5 5 4 7 4

**Задача 5.** Matrix56.

Дана матрица размера M x N (N — четное число). Поменять местами

левую и правую половины матрицы.

from random import randint as rnd

n,m = int(input('n:\n')),int(input('m:\n'))

upr = []

botl = []

a = [[rnd(1,10) for \_ in range(n)] for \_ in range(m)]

print(\*a, '\n', sep='\n')

for i in range(int(m/2)):

upr.append(a[i][int(n/2):])

for i in range(int(m/2),m):

botl.append(a[i][:int(n/2)])

for i in range(int(m/2)):

a[i][int(n/2):] = botl[i]

for i in range(int(m/2),m):

a[i][:int(n/2)] = upr[i]

print(\*a, sep='\n')

**Входные данные:**

n:

2

m:

3

**Вывод:**

[1, 4]

[10, 8]

[5, 1]

**Содержание:**

1. <https://pythonworld.ru/moduli/modul-math.html>
2. <https://pythonru.com/osnovy/modul-math-python>
3. <https://pythonworld.ru/moduli/modul-cmath.html>
4. <https://python-scripts.com/math>