**Импорт модулей. Базовые модули (os.path, random, math). Pip и работа с пакетами.**

Функции языка Python позволяют упаковывать фрагменты программного кода, чтобы его можно было многократно использовать во всей программе. Но что делать если требуется использовать одни и те же функции в разных программах? Для решения такой проблемы в языке Python существуют модули.

**Импорт модулей**

Модуль – это средство объединения функций в коллекции, что бы их можно было использовать в разных программах. Любую программу, написанную на языке Python и расположенную в файле с расширением .py, можно считать отдельным модулем. Главное различие между модулем и программой состоит в том, что программа предназначен для того, чтобы её запускали, а модуль предназначен для того, чтобы его импортировали, т.е. подключали к основной программе, и использовали в программе.

Синтаксис подключения модулей к программе имеет несколько вариантов:

* import module\_name – импортируем в программу модуль с именем module\_name;
* import module\_name1, module\_name 2, … , module\_nameN – импортируем в программу N модулей;
* import module\_name as alias - импортируем в программу модуль с именем module\_name и с помощью ключевого слова as даём ему псевдоним alias;

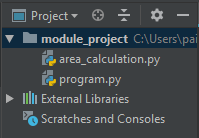
Ключевое слово import указывает на то, что после него следует перечисление имен модулей, которые будут использованы в проекте. Ключевое слово as используется для того, чтобы давать короткие названия для модулей внутри программы. Обычно, подключение модулей помещается в начало программы. Имена модулей содержат только буквы нижнего регистра и цифры, в качестве разделите между словами используется “\_”. Остальные символы не должны присутствовать в названии модуля.

Для обращения к функциям и переменным определенных в модули используется следующая конструкция: имя\_модуля(или псевдоним).название\_функции (переменной).

Язык программирования Python поставляется с большим количеством дополнительных модулей, которые объединены в стандартную библиотеку Python. Модули из этой библиотеки импортируются ранее описанным способом.

Пример:

Для примера, в проекте создадим 2 файла: area\_calculation и program. Структура проекта примет следующий вид:



В файле area\_calculation.py определим функции для расчета площади следующих фигур:

* прямоугольник;
* круг;
* треугольник;

Модуль area\_calculation:

# Импортируем модуль math из стандартной библиотеки языка Python  
import math  
  
  
# Фнкция возвращает площадь прямоугольника  
def rectangle\_area(a, b):  
 return a \* b  
  
  
# Функция возвращает площадь круга  
# Обратите внимание, число Pi берется из модуля math, что повышает "читаемость" кода.  
def circle\_area(r):  
 return math.pi \* r \*\* 2  
  
  
# Фнкция возвращает площадь треугольника  
# Для расчета квадратного корня числа, используется метод sqrt из модуля math  
def triangle\_area(a, b, c):  
 p = (a + b + c) / 2  
 return math.sqrt(p \* (p - a) \* (p - b) \* (p - c))

Программа program.py:

import area\_calculation as ac, math  
  
# Выведет на консоль площадь прямоугольника со сторонами 10 и 5  
print(ac.rectangle\_area(10, 5))  
  
# Выведет на консоль площадь круга с радиусом 5  
# Полученный результат округляется до ближайшего большего числа с помощью функции ceil.  
# Функция определена в модули math  
print(math.ceil(ac.circle\_area(5)))  
  
# Выведет на консоль площадь прямоугольника со сторонами 3,4,5  
print(ac.triangle\_area(3, 4, 5))

**Базовые модули (os.path, random, math)**

Как уже упоминалось ранее, в стандартную библиотеку языка Python входит множество модулей. В этом пособии рассмотрим следующие модули:

* os.path - является вложенным модулем в модуль os, и реализует полезные функции для работы с путями.
* random - предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.
* math - этот модуль предоставляет обширный функционал для работы с числами.

**Модуль os.path**

Модуль os.path обладает большой коллекцией функций предназначенных для работы с путями к файлам и папкам. Рассмотрим основные функции данного модуля:

* os.path.exists(path) – возвращает True или False, т.е. существует ли указанный путь в системе.   
  Пример:

import os.path  
  
# Проверяем существует ли файл program.py в папке с проектом. ./ - указывает на папку из которой запускается программа  
# На консоль выведет: True  
print(os.path.exists('./program.py'))

* os.path.join(path1,path2,…..,pathN) - соединяет пути с учётом особенностей операционной системы.  
  Пример:

import os.path  
  
path1 = 'd:/'  
path2 = 'python test'  
path3 = 'text.txt'  
# Выведет на консоль: d:\python test\text.txt  
print(os.path.join(path1, path2, path3))

**Модуль random**

Рассмотри основные функции для генерации случайных чисел с помощью модуля random:

* random.randint(A,B) – возвращает случайное целое число N, которое удовлетворяет двойному неравенству: A<=N<=B.  
  Пример:

import random  
# Создаем пустой список  
rnd\_number = list()  
# Добавляем 3 случайных целых числа в список  
for i in range(0, 3):  
 rnd\_number.append(random.randint(1, 10))

* random.uniform(A,B) - возвращает случайное число N с плавающей точкой, которое удовлетворяет двойному неравенству: A<=N<=B.  
  Пример:

import random  
# Создаем пустой список  
rnd\_number = list()  
# Добавляем 3 случайных числа c плавающей точкой в список  
for i in range(0, 3):  
 rnd\_number.append(random.uniform(1, 10))

* random.choice(collection) – возвращает случайный элемент коллекции.  
  Пример:

import random  
# Создаем список из 3-х элементов: 'a','b' и 'c'  
s = list('abc')  
# Выбираем случайный элемент из списка s  
rnd\_char = random.choice(s)

* random.shuffle(collection) – перемешивает коллекцию collection. В качестве коллекции можно передать все типы структур, рассмотренные ранее, кроме кортежа.  
  Пример:

import random  
# Создаем список l из 3-х элементов  
l = [1, 2, 4]  
# Перемешиваем список l   
random.shuffle(l)  
# Выведет на консоль: [4, 1, 2]   
print(l)  
t = tuple(l)  
# Выведет на консоль сообщение об ошибки, т.к. кортеж нельзя перемешивать  
random.shuffle(t)

**Модуль math**

В дополнение к рассмотренным ранее функциям и переменным: ceil, sqrt, pi; модуль math содержит множество полезных математических функций. Рассмотрим наиболее важные функции:

* math.fabs(x) – возвращает модуль числа x  
  Пример:

import math  
# Выведет на консоль: 100.0  
print(math.fabs(-100.0))

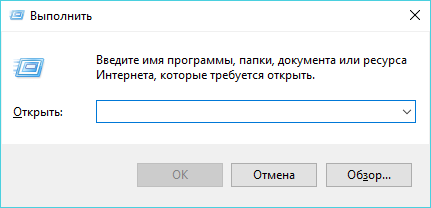
* math.cos(x), math.sin(x), math.tan(x) – возвращают соответственно cos, sin и tan для угла X. X – указывается в радианах.  
  Пример:

import math  
# Выведет на консоль: Cos(Pi) = -1.0 Sin(Pi) = 1.2246467991473532e-16 Tan(Pi) = -1.2246467991473532e-16  
# Поему sin(Pi) и Tan(Pi) не равны 0? это связано с тем, что функции sin() и cos() возващают числа типа float  
print('Cos(Pi) = ', math.cos(math.pi), 'Sin(Pi) = ', math.sin(math.pi), 'Tan(Pi) = ', math.tan(math.pi))

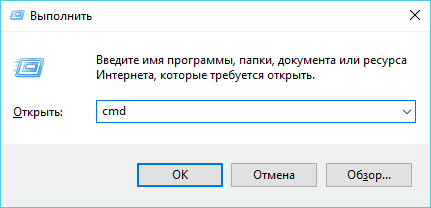
**Pip и работа со сторонними пакетами**

В языке Python предусмотрена работа не только с собственными модулями или со стандартной библиотекой Python, но и с модулями других программистов. Такие модули называются пакетами, и они устанавливаются на компьютер отдельно с помощью специального менеджера пакетов – pip.

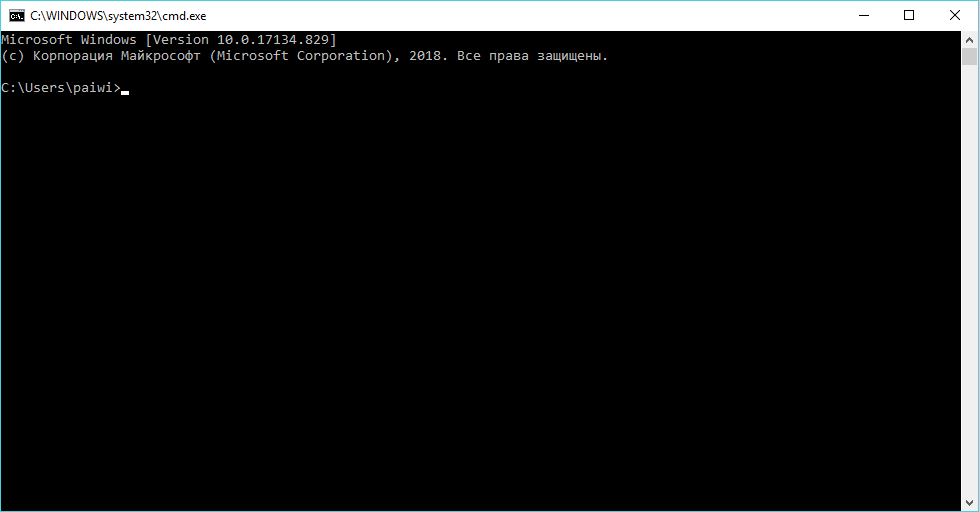
Установим с помощью pip пакете для математических операций под названием “numpy”. Для этого, нажмем комбинацию клавиш Win + R которая открывает следующее окно:



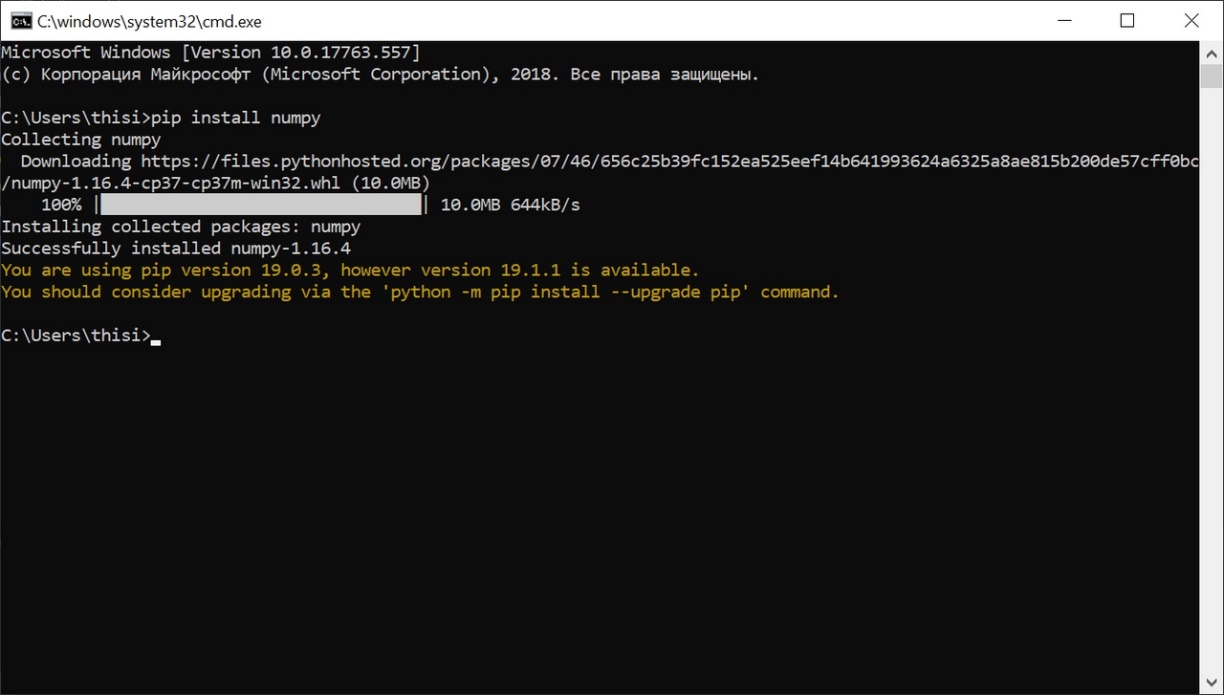
В открывшемся впишем следующую строку:



Нажмем кнопку “Ок” и откроется командная строка Windows:



В командной строке Windows пишем следующую строку: pip install numpy , где pip – обращение к пакетному менеджеру pip, install – команда для установки пакета, а numpy – название пакета. После выполнения команды pip install numpy, т.е. после введенной строки была нажата клавиша Enter, на экран будет выведена следующая запись:



Способ подключения сторонних пакетов к собственным проектам не отличается от подключения собственных модулей или модулей из стандартной библиотеки языка Python. После успешной установки пакета numpy, перейдите в IDE и подключите этот пакет к своей программе.

Пример:

# Подключаем пакет numpy к собственному проекту  
import numpy as np  
# Вызываем функцию рандом из вложенного модуля numpy.random  
# Выведет на консоль случайное число от 0 до 1   
print(np.random.random())