Знакомство с языком программирования Python. Ввод. Вывод. Оператор присваивания. Математические операции

Самые популярные языки программирования:

- Java
- Python
- JavaScript
- C#
- C
- C++
- PHP
- SQL
- Visual Basic .NET
- Ruby

Python – это интерпретируемый язык программирования с динамической типизацией данных, поддержкой объектно-ориентированного программирования для создания программ самого разнообразного назначения.

Python используется в таких проектах как:

Google YouTube Instagram

Яндекс facebook и др.

- Впервые язык Python был анонсирован в 1991 году голландским разработчиком Гвидо Ван Россумом.
- Текущей актуальной версией на момент написания данного материала является **3.11.0**.
- Более подробную информацию о всех релизах, версиях и изменения языка, а также собственно интерпретаторы и необходимые утилиты для работы и прочую полезную информацию можно найти на официальном сайте https://www.python.org/.

Python – это интерпретируемый язык программирования. Т.е. программный код на языке Питон преобразуется в машинный код построчно специальной программой — интерпретатором.

Python имеет достаточно простой синтаксис. Читать код на этом языке программирования легко, т.к. в нем используется минимум вспомогательных элементов, а правила языка заставляют программистов делать отступы.

С другой стороны, Python – это полноценный, можно сказать универсальный, язык программирования. Это язык так называемого сверх высокого уровня: он поддерживает объектно-ориентированное программирование (на самом деле он и разрабатывался как объектноориентированный язык).

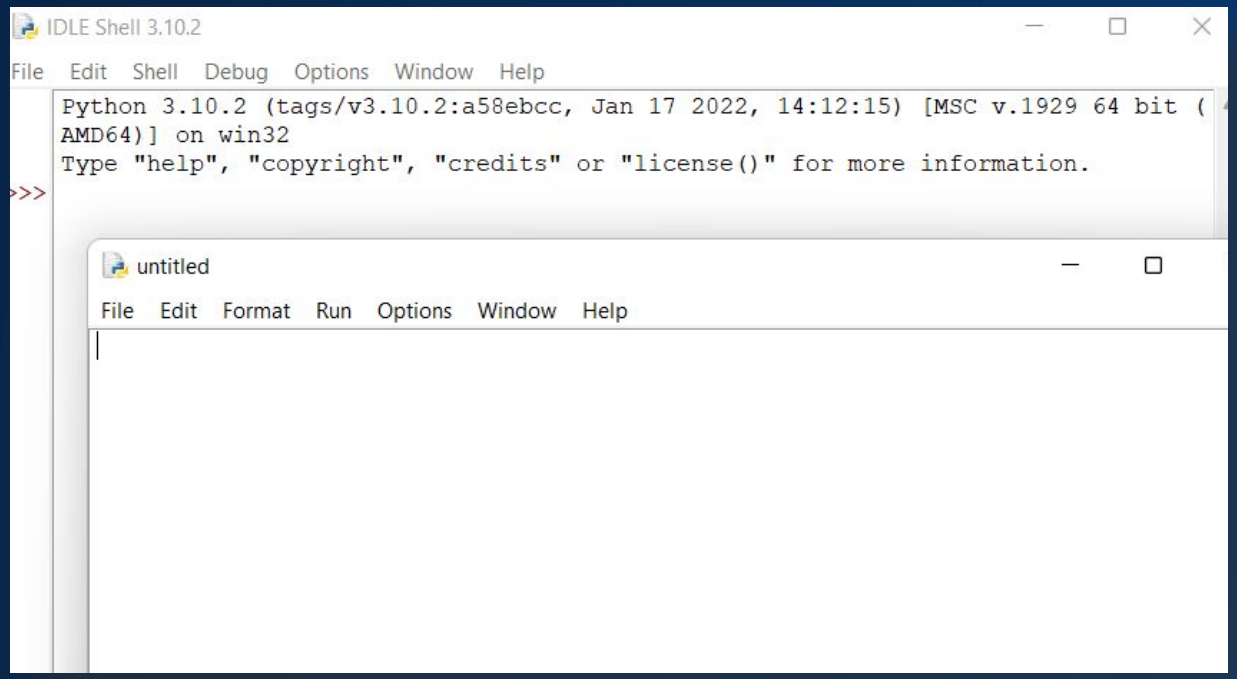
- ✔Веб разработка
- ✔Работа с данными
- ✔Машинное обучение и нейронные сети
- ✔Написание скриптов под разные системы

В основном, при знакомстве с языком программирования Python нас буду интересовать три типа данных:

– целые числа (integer) – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0);
– числа с плавающей точкой (float point) – дробные числа (например, 1.45, -3.789654, 0.00453). Примечание: разделителем целой и дробной части служит точка, а не запятая;
– строки (string) — набор символов, заключенных в кавычки (например, “ball”, “What is your name?”, 'dkfjUUv', '6589'). Примечание: кавычки могут быть одинарными или двойными.

Открыть среду программирования Python:

Пуск => Python 3.10 => IDLE (Python GUI) => File => New File



Основы Python

Программа на языке Python состоит из набора инструкций. Каждая инструкция помещается на новую строку. Например:

- print(2 + 3)
- print("Hello")
- Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой.

Python - регистрозависимый язык, поэтому выражения print и Print или PRINT представляют разные выражения. И если вместо метода print для вывода на консоль мы попробуем использовать метод Print:

Print("Hello World")

у нас ничего не получится.

Ввод и вывод

Основной функцией для вывода информации на консоль является функция print(). В качестве аргумента в эту функцию передается строка, которую мы хотим вывести:

```
print("Hello Python")
```

Если же нам необходимо вывести несколько значений на консоль, то мы можем передать их в функцию print через запятую:

```
print("Full name:", "Tom", "Smith")
```

В итоге все переданные значения склеятся через пробелы в одну строку.

Первая программа:

```
print("Hello, World!!")
```

Запуск программы:
Клавиша F5
Или в меню:
Run => Run Module

Переменная и оператор присваивания

```
message = 'Hello, World!'
print(message)
```

`message` – переменная

`=` – оператор присваивания

Переменная – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.

Имена переменных

Имена переменных могут состоять из:

- Латинские буквы (строчные и заглавные буквы различаются!)
- Русские буквы (не рекомендуется)
- Цифры (имя не может начинаться с цифры и состоять только из цифр)
- Знак подчеркивания _

Нельзя использовать в именах переменных:

- Пробелы
- Знаки +,-,>,<,=,(), ! и др.
- Ключевые слова языка Python

Нельзя использовать как имена переменных ключевые слова языка Python:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	print
break	except	in	raise	

Математические операции

```
a = 78001457
b = 2546880
c = a + b
print(c)
```

```
a = 78
b = 25
c = (a-b)*(a+b)/27
print(c)
```

Другие математические операции:

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)
x ** y	Возведение в степень
ceil(x)	Округление до ближайшего большего числа.

Модуль math

```
import math

a = int(input())
c = math.sqrt(a*a)
print(c)
```

Функция ввода

```
name = input("Введите своё имя: ")
print("Привет,", name)
```

Изменение типа данных

Бывают случаи, когда программа получает данные в виде чисел, а в результате должна выдавать строки (или наоборот). В таком случае можно использовать, так называемые, встроенные функции, позволяющие преобразовать один тип данных в другой. Так функция int() преобразует переданную ей строку (или число с плавающей точкой) в целое, функция str() преобразует переданный ей аргумент в строку, float() - в дробное число.

По умолчанию все **введённые** данные интерпретатор Питона понимает, как строки. Поэтому, если мы хотим получить число, то строку придётся преобразовать в число.

Ввод целого числа:

```
n = int(input("Введите число: "))
```

Функция преобразования к целочисленному типу:

```
n = int(s)
```

Функция преобразования к строковому типу:

```
s = str(n)
```

Задание. Напишите программу, которая получает на вход два числа и выводит их сумму:

```
a = input("Введите число a: ")
b = input("Введите число b: ")
sum = a+b
print("a+b=", sum)
```

Почему программа работает неправильно?
Что исправить в программе, чтобы она работала правильно?

Задача. В каждой строке определить тип и значение переменной:

```
a = 5
n = input()    #человек вводит цифру 8
c = int(n)
d = a*c
d = d-a
s = "Питон"
d = n+a
m = n+s
```

```
# Комментарии к программе, компьютер
# их не читает
```

Генератор случайных чисел

Функция генерации случайного целого числа из отрезка [x,y]:

```
import random
a = random.randint(x,y)
```

Задания

1) Вывести на экран три введенных с клавиатуры числа в порядке, обратном их вводу.
2) Ввести с клавиатуры два числа и вывести целую часть от деления первого на второе.
3) Ввести с клавиатуры основание и высоту треугольника и вывести площадь треугольника.
4) Ввести с клавиатуры два катета и вывести гипотенузу. (Квадратный корень – это возведение в степень (1/2))
5) Сгенерировать случайное двузначное число, вывести на экран это число, а также сумму и произведение его цифр.

   Для получения цифр используйте целочисленное деление на 10 и взятие остатка от деления на 10. Пример для числа 47:

   47//10=4 47%10=7

Логические выражения и логический тип данных

У логического типа всего два возможных значения: True (правда) — 1 и False (ложь) — 0. Эти значения и являются результатом логических выражений.

Логические операторы

В языках программирования обычно используются специальные знаки, подобные тем, которые используются в математических выражениях: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно). Новыми для восприятия могут оказаться обозначение равенства: == (два знака "равно"); а также неравенства !=. Часто начинающие программисты ошибаются и вместо двух знаков "=" пишут один. Однако, вы должны помнить – один знак используется в операциях присвоения.

Примеры работы с логическими выражениями на языке программирования Python:

1. x = 12 – 5 *# это не логическая операция, а операция присвоения переменной x результата выражения 12 — 5*
2. x == 4 *# x равен 4*
3. x == 7 *# x равен 7*
4. x != 7 *# x не равен 7*
5. x != 4 *# x не равен 4*
6. x > 5 *# x больше 5*
7. x < 5 *# x меньше 5*
8. x <= 6 *# x больше или равен 6*
9. x >= 6 *# x меньше или равен 6*

Сложные логические выражения

На практике не редко используются более сложные логические выражения. Может понадобиться получить логический тип ответа ("Да" или "Нет") в зависимости от результата выполнения двух простых выражений. Например, "на улице идет снег или дождь", "переменная new больше 12 и меньше 20" и т.п.

В таких случаях требуются специальные операторы, объединяющие два и более простых логических выражения. Мы рассмотрим два способа объединения: через, так называемые, логические И (and) и ИЛИ (or).

Чтобы получить истину (True) при использовании оператора and, необходимо, чтобы результат обоих простых выражений, которые связывает данный оператор, были истинными. Если хотя бы в одном случае результатом будет False (ложь), то и все сложное выражение будет ложным.

Чтобы получить истину (True) при использовании оператора or, необходимо, чтобы результат хотя бы одного простого выражения, входящего в состав сложного, был истинным. В случае оператора or сложное выражение становится ложным лишь тогда, когда ложны все составляющие его простые выражения.

1. x = 8
2. y = 13
3. x == 8 and y < 15 *# x равен 8 и y меньше 15*
4. x > 8 and y < 15 *# x больше 8 и y меньше 15*
5. x != 0 or y >15 *# x не равен 0 или y меньше 15*
6. x < 0 or y >15 *# x меньше 0 или y меньше 15*