Введение в язык программирования Python

Желтым цветом выделены программы на питоне. Вы можете попробовать запустить их на своем компьютере.

0. Функция print().

- Позволяет выводить строки, числа, значения переменных и выражений, результат работы функций.
- Для этого нужно в скобках через запятую перечислить то, что мы хотим увидеть в качестве результата работы нашей программы.
- Аргументы (то, что вы указали в скобках через запятую) будут выведены через пробел. После того, как функция **print()** выведет все, что ей передали, она переводит курсор на новую строку.
- Количество аргументов не ограничено.

`print('hello', 'world', 1, 2, 3)`	`hello world 1 2 3`

`print('hello')`
`print('world', 1, 2)`
`print(3)`
	`hello`
	`world 1 2`
	`3`

`a = 5` `print(a + 3)`	`8`

1. Переменные.

- Переменные - это область памяти компьютера, которую мы называем каким-то именем.
- Мы можем записать в переменную значение, с которым в дальнейшем можно выполнять некоторые операции.
- Переменные имеют тип, который определяет допустимые операции (они отличаются для разных типов).
- В большинстве языков программирования тип нужно указывать в коде. Но в питоне достаточно присвоить в переменную значение, и тип будет вычислен автоматически.
- Узнать, к какому типу принадлежит ваша переменная, можно при помощи функции **type()**.

Тип int отвечает за целые числа. Тип str отвечает за строки. Как вы думайте, что выведут следующие программы? Проверьте свои догадки с помощью питона.	`print(type(2))`
	`a = 2` `print(type(a))`

	`a = 2`
	`b = 3`
	`print(type(a + b))`

	`print(type('hello'))`

	`s = 5`
	`s = 'hello'`
	`print(type(s))`

	`print(type(7 / 3))`

2. Строки.

- Питон считает, что набор символов в кавычках имеет тип **str** - строка.
- Можно использовать как одинарные, так и двойные кавычки, но с обеих сторон строки они должны быть одинаковыми.
- Для строк определены операции сложения - конкатенация (**+**), и умножения на целое число (*).

Подробнее о строках мы поговорим позднее.

`a = 'stroka'`
`b = "toje stroka"`
`print(a, b)`
	`stroka toje stroka`	`a = 'abc'`
		`b = 'def'`
		`print(a + b)`
			`abcdef`

`a = 'tak nelza"`
`print(a)`
	`ошибка`	`a = 'abc'`
		`b = 'def'`
		`print(a * b)`
			`ошибка`

`a = 'a "vot tak" mojno'` `print(a)`	`a "vot tak" mojno`	`a = 'abc'` `print(a * 4)`	`abcabcabcabc`

3. Целые числа.

- Целое число или выражение, результат которого целое число, питон распознает как тип данных **int**.
- Целые числа могут быть сколь угодно большими (или маленькими). Их размер ограничивается только имеющейся в распоряжении компьютера памятью.
- Для целых чисел определены следующие операции:
-

сложение	**+**	`a = 7`
		`b = 2`
		`print(a + b)`
			`9`

вычитание	**-**	`a = 2`
		`b = 3 - 1`
		`print(a - b)`
			`0`

умножение	*****	`print(4 - 3 * 2 + 12)`	`10`
возведение в степень	****	`a = 5` `print(a ** 3)`	`125`
		`print(4 ** (-1))`	`0.25`
целочисленное деление	**//**	`print(11 // 3)`	`3`
остаток от деления	**%**	`print(11 % 3)`	`2`

обычное деление	**/**	`a = 4`
		`b = 2`
		`print(a / b)`
			`2.0`

- Деление с помощью **/** возвращает число с точкой - это уже другой тип данных.
- Возведение в отрицательную или дробную степень тоже может привести к тому, что значение переменной перестанет быть целым числом.

4. Функция input().

- Позволяет считывать информацию из потока ввода, сохранять ее в переменные, а затем обрабатывать.
- Именно благодаря этой функции и переменным, написав пару строк кода, можно создать программу, которая будет выводить сумму для абсолютно любых двух чисел.

`a = int(input())`
`b = int(input())`
`c = a + b`
`print(c)`
	входные данные:	`5`
		`8`

	результат работы программы:	`13`

- Функция **input()** вернет все то, что было введено до перевода строки, причем тип результата - это строка.
- Для того, чтобы получить из строки число, нужно использовать функцию **int()**, которой в качестве аргумента мы должны передать строку, являющуюся правильным целым числом (иначе питон выдаст сообщение об ошибке).

`a = int('125')` `print(type(a))`	`<class 'int'>`	`a = int('qwerty')` `print(a)`	`ошибка`

- Два (или более) числа из одной строки таким образом прочитать не выйдет. Для считывания нескольких чисел, записанных в одной строке через пробел, нужно использовать следующую конструкцию.

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

- В левой части можно указать любое количество переменных через запятую. Для того, чтобы программа сработала без ошибки, входные данные должны содержать столько же целых чисел, разделенных пробелами.

- Примеры использования функции **input()**.

	входные данные	результат работы программы

`a = input()`
`print(a)`
`print(type(a))`
	`qwerty`	`qwerty`
		`<class 'str'>`

	`25`	`25` `<class 'str'>`

`a = int(input)`
`print(a)`
`print(type(a))`
	`-5`	`-5`
		`<class 'int'>`

	`qwerty`	`ошибка`

`a = int(input)`
`b = int(input)`
`print(a + b)`
	`5`
	`6`
		`11`

	`5 6`	`ошибка`
`a, b, c = map(int, input().split())` `print(a + b * c)`	`1 2 3`	`7`
	`1, 2, 3`	`ошибка`
	`1 2` `3`	`ошибка`

5. Условный оператор.

- Позволяет описывать в программе различное поведение (выполнять различные операции), в зависимости от значения переменных, выражений и результатов функций.
- Условная инструкция в Питоне имеет следующий синтаксис:

if условие:
  блок инструкций 1
elif условие 2:
  блок инструкций 2
elif условие 3:
  блок инструкций 3
else:
  блок инструкций 4
	• Условия проверяются поочередно, сверху вниз. Если какое-то из условий выполняется, то выполняется соответствующий блок инструкций.
	• После этого проверок других условий не происходит. То есть будет выполнено не более одного блока инструкций.
	• Если все условия ложны, то будет выполнен блок инструкций, относящийся к **else**. Если слово **else** и относящийся к нему блок инструкций отсутствуют, тогда никакой блок инструкций исполнен не будет, и программа продолжит выполнение со строки после условного оператора.

- Первая строка проверки условия и относящийся к ней блок инструкций являются обязательными.
- Условная конструкция может содержать любое количество строк, начинающихся с **elif** (в том числе не содержать их вовсе). Все они должны содержать условие и соответствующий блок инструкций.
- В условной инструкции может быть не более одного **else** и последующего блока инструкций. **Else** обязательно должен идти после всех остальных проверок условий. Также **else** можно не писать.

- Обратите внимание на отступы. Код, который принадлежит одному из блоков инструкций, должен быть написан с одинаковым отступом.
  Для отступа используйте 2 или 4 пробела, всегда одинаково в одной программе.

- Условия обычно имеют вид сравнения двух величин с помощью следующих операторов:

>	больше	<	меньше
>=	больше или равно	<=	меньше-равно
==	равно	!=	не равно

- В Питоне можно использовать двойные неравенства (когда значение переменной или выражения сравнивается сразу с двумя значениями).
- Примеры сравнений:

(a * 2 <= 10)	значение переменной a, умноженное на 2, меньше или равно 10	(5 <= x ** 2 < 20)	значение переменной x, возведенное в квадрат, больше или равно 5, но меньше 20

- Результат сравнения имеет тип данных **bool**. В этом типе данных всего два значения - **True** и **False** - соответственно правда и ложь. Если условие выполняется, то его значение принимается за **True**, а иначе - за **False**.

- Условия можно комбинировать между собой с помощью следующих операторов:

(условие 1) **and** (условие 2)	Логическое “И”. Оба условия должны выполняться.
(условие 1) **or** (условие 2)	Логическое “ИЛИ”. Достаточно, чтобы было выполнено хотя бы одно из двух условий.
not (условие)	Логическое “НЕ”. Меняет значение условия на противоположное. То есть чтобы значение было **True**, значение условия в скобках должно быть **False**.

- Блок инструкций может содержать внутри себя любые операции, так же, как и обычная программа на питоне. Можно написать **if** внутри другого **if**'а, но не забывайте про отступы. У блока инструкций отступ должен быть на один больше, чем у строки проверки условия.

- Примеры использования условного оператора.

`a = int(input())`
`if (0 <= a < 10):`
`  print(a)`
	`a = int(input())`
	`if (0 <= a) and (a < 10):`
	`  print(a)`
		`a = int(input())`
		`if (0 <= a):`
		`  if (a < 10):`
		`  print(a)`
			Выводит введенное число, если оно положительное и однозначное.

`x = int(input())`
`if (x < 0):`
`  x = -x`
`print(x)`
	Заменяет значение x на противоположное, в случае, если x отрицательное (вычисление модуля числа).

`a, b, c = map(int, input().split())`
`m = 0`
`if (a >= b) and (a >= c):`
`  m = a`
`if (b >= a) and (b >= c):`
`  m = b`
`if (c >= a) and (c >= b):`
`  m = c`
`print(c)`
	Поиск максимального среди трех чисел.

`a = int(input())`
`if (a < 1) or (a > 11):`
`  print('ошибка')`
`elif (a <= 4):`
`  print('начальная школа')`
`  print('корпус 1')`
`elif (a <= 9):`
`  print('средняя школа')`
`  print('корпус 2')`
`else:`
`  print('старшая школа')`
`  print('корпус 3')`
	`a = int(input())`
	`if (a < 1) or (a > 11):`
	`  print('ошибка')`
	`if (a <= 4):`
	`  print('начальная школа')`
	`  print('корпус 1')`
	`if (a <= 9):`
	`  print('средняя школа')`
	`  print('корпус 2')`
	`else:`
	`  print('старшая школа')`
	`  print('корпус 3')`
		По номеру класса, определяется в каком корпусе проходят уроки.
		
		Второй вариант не работает.
		Как вы думаете, почему?

6. Дополнительная информация.

- Питон - интерпретируемый язык программирования. Это означает, что программа не компилируется в один исполняемый файл с расширением .exe, а исполняется построчно.
- Поэтому, сообщение об ошибке исполнения программы появится, только если питон дойдет до строки с этой ошибкой.

Например, этот код будет работать корректно для значений a < 10 и упадет при a >= 10.	`a = int(input())`
	`if a < 10:`
	`  print(a)`
	`else:`
	`  print(a / 0)`