



Типы данных – назначение и роль в программе. Операнды и операторы – вычисление выражений

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur



Типы данных

Тип данных - описывает множество значений и операций над этими значениями

Тип данных	Примеры
int	1, 5, -9, 0
float	2.5, 2.3333333
bool	True, False
str	"Hello world"
NoneType	None
list	[4, 1, 0, 9]
tuple	(5, 3, 9, 2)
dict	{ name: 'ivan', kg: 63 }
set	{ 5, 9, 1, 0 }
complex	2 + 3j



Преобразование типов данных

Функция	Примеры
int()	int("1234")
float()	float("1234.43")
bool()	bool(2), bool("a")
str()	str(234)
list()	list("abcde")
tuple()	tuple("egda")
dict()	dict([('k', 'v')])
set()	set([2,2,2,1,1])
complex()	complex(123)



Основные элементы языка

- Основными элементами любого языка программирования являются его алфавит, синтаксис и семантика.
- **Алфавит** – это набор символов, которые допустимо использовать в программах на языке программирования. Обычно это буквы, цифры, знаки, разделители (пробелы и переводы строк). В Питоне можно использовать любые символы Юникод, хотя рекомендуется в именах применять только латинские буквы. Прописные и строчные буквы различаются: name и Name – различные имена.
- **Синтаксис** – совокупность правил образования языковых конструкций, или предложений языка программирования – выражений, команд (операторов), блоков, функций и пр. Необходимо строгое соблюдение правил правописания (синтаксиса) программы.
- **Семантика** – смысловое содержание конструкций, предложений языка, семантический анализ – это проверка смысловой правильности конструкции.



Структура программы

- **Программа** на Python – текст, содержащий последовательность команд (операторов).
- **Оператор** – предложение языка, описывающее определенное действие. Обычно каждый оператор записывается в отдельной строке программы.
- В программе могут быть также определения методов и классов, которые начинаются с ключевого слова `def` (будут изучаться позже).
- В операторах могут использоваться ключевые слова. **Ключевые слова** – английские слова, имеющие специальные значения. Эти слова зарезервированы и не могут использоваться в другом качестве, например в качестве имен. Среда разработки автоматически их выделяет в тексте программы.

```
>>> import keyword  
>>> keyword.kwlist  
[ 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',  
'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',  
'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield' ]
```



Константы

- Операторы языка работают с данными (числами, текстами, множествами и пр.), которые хранятся в памяти компьютера
- Константы** - это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения
- Примеры констант:
 - целые** числа: 12, -23, 0b1010001
 - действительные** числа: 1.0, -7.8
 - логические**: True (истина), False (ложь)
 - строковые**: "I'm study Python", 'информатика'



Переменные

- **Переменные** – это данные, которые могут изменять свои значения в ходе выполнения программы
- Переменная имеет
 - **Имя (идентификатор)** - как обращаться к переменной
 - **Значение** – что лежит в переменной
- **Оператор присваивания** устанавливает связь между именем и значением переменной

Имя = выражение

- **Имя** переменной может содержать буквы, цифры, знак _ и не может начинаться с цифры

```
>>> x = 2 + 2
>>> x
4
>>> name = "Вася"
>>> name
'Вася'
>>>
```



Выражения

- **Выражения** могут включать константы, переменные, вызовы функций, соединенные знаками операций. Вычисление значения выражение выполняется в соответствии с приоритетами операций

- Каждое выражение имеет **значение**, одному из **типов данных**

```
>>> (x + 2) * x - 10
      14
>>> name * 2
      'ВасяВася'
...
|
```

- **Тип данных** определяет

- Область допустимых значений
- Допустимые операции
- Объём и структуру памяти для хранения значения
- Python использует динамическую типизацию (тип переменной определяется ее значением) и строгий контроль типов



Простые типы в Python

Целое число

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

Строка

```
>>> s="Hello"  
>>> s='Hello'  
>>> type(s)  
<class 'str'>
```

Действительное

```
>>> x=3.5  
число  
>>> type(x)  
<class 'float'>
```

Комплексное число

```
>>> z=3+5j  
>>> type(z)  
<class 'complex'>
```

Логический тип

```
>>> b=True  
>>> type(b)  
<class 'bool'>
```



Арифметические операции

+, -, *

/ - частное от деления

// - целая часть от деления

% - остаток от деления

```
>>> -10//3  
-4  
>>> -10%3  
2
```

```
>>> -10//(-3)  
3  
>>> -10%(-3)  
-1  
>>> 10//(-3)  
-4  
>>> 10%(-3)  
-2
```

```
>>> -10.5//3  
-4.0  
>>> -10.5%3  
1.5  
>>> -10.5//3.5  
-3.0  
>>> -10.5%3.5  
0.0
```



Арифметические операции

** возвведение в степень

```
>>> 3**4  
81
```

Все арифметические операции можно сокращать с присваиванием: +=, -=, ...

```
>>> a=5  
>>> a**=2  
>>> a  
25
```

```
>>> b=7  
>>> b*=3  
>>> b  
21
```



Математические функции

Модуль `math` нужно подключить командой
`import math`

`sqrt(x)` – квадратный корень из x

`fabs(x)` – модуль x

`sin(x), cos(x), tan(x)` – тригонометрические
функции

`floor(x)` – округление вниз

`ceil(x)` – округление вверх

```
>>> import math  
>>> math.sqrt(16)  
4.0
```



Перенос выражения

Перенос можно делать
внутри скобок или с
помощью знака \

```
>>> (3+5-
      7)
1
>>> 3+5\
      -7
1
>>> 3+5
8
```



Комментарии

- `#` комментирует весь текст до конца строки;
- `""` закомментированный текст `""`
- `"""` закомментированный текст `"""`

```
a=3+5#-4
""" комментарий
многострочный
"""
```



Вывод данных

- Используется функция `print(выр.1, выр.2...)`
- По умолчанию между выводимыми объектами ставится пробел, а в конце перевод строки

```
a = 4  
b = 6  
print(a, '+', b, '=')  
print(a+b)
```

```
-----  
4 + 6 =  
10  
>>> |
```

- Можно изменить разделитель в параметре `sep` и последний символ в параметре `end`

```
a = 4  
b = 6  
print(a, '+', b, '=', sep=' ', end=' ')  
print(a+b)
```

```
4+6= 10  
>>> |
```



Форматный вывод

```
a = 4  
b = 6  
print ('{}+{}={}'.format(a,b,a+b))
```

```
4+6=10  
>>> |
```

```
a = 4  
b = 6  
print(f"{a} + {b} = {a + b}")  
4 + 6 = 10  
|
```

```
import math  
print('{:10.3f}'.format(math.pi))
```

```
----- РЕЗУЛЬТАТ  
3.142  
>>> |
```



Ввод данных

- Вводится целая строка!

```
a = input()      36
print(type(a)) <class 'str'>
```

- При необходимости её надо преобразовывать в число:

```
a = int(input()) 35
print(a+1)        36
```



Ввод данных

- Можно выводить строку с вопросом перед вводом переменной:

```
a = int(input('Enter '))      Enter 45
print(a+1)                      46
```

$$A + B$$



```
a = int(input())
b = int(input())
c = a+b
print(c)
>>>
```

```
print(int(input())+int(input()))
```



Приоритет операторов

Самые приоритетные операции вверху, снизу — с низким приоритетом.

Вычисления выполняются слева направо, то есть, если в выражении встретятся операторы одинаковых приоритетов, первым будет выполнен тот, что слева.

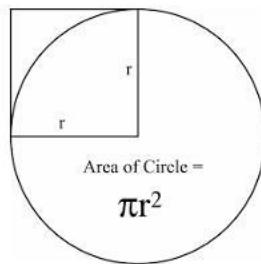
Оператор возвведения в степень исключение из этого правила. Из двух операторов `**` сначала выполнится правый, а потом левый.

(Скобки
<code>**</code>	Возведение в степень
<code>+x, -x, ~x</code>	Унарные плюс, минус и битовое отрицание
<code>*, /, //, %</code>	Умножение, деления, взятие остатка
<code>+, -</code>	Сложение и вычитание
<code><<, >></code>	Битовые сдвиги
<code>&</code>	Битовое И
<code>^</code>	Битовое исключающее ИЛИ (XOR)
<code> </code>	Битовое ИЛИ
<code>==, !=, >, >=, <, <=, is, is not, in, not in</code>	Сравнение, проверка идентичности, проверка вхождения
<code>not</code>	Логическое НЕ
<code>and</code>	Логическое И
<code>or</code>	Логическое ИЛИ



Задачи

1. Напишите программу Python, чтобы получить версию Python, которую вы используете.
2. Напишите программу, которая преобразует кг в граммы. На входе получает значение в кг, а на выходе значение в граммах
3. Напишите программу Python, которая принимает радиус круга от пользователя и вычисляет площадь



4. Напишите программу Python, которая принимает имя и фамилию пользователя, и выводит сперва фамилию, а потом имя с пробелом между ними.
5. Напишите программу Python, которая вычисляет периметр и площадь прямоугольника
6. Напишите программу Python, которая принимает два числа a и b , далее находит сумму, разность, произведение и деление a на b . В результате отобразите ответы для каждой из арифметической операции.



Операторы сравнения

Оператор	Описание	Пример	Результат
<code>==</code>	проверяет одинаково ли значение операндов, если одинаковы – то условие является истиной	<code>a == b</code>	<code>False</code>
<code>!=</code>	проверяет одинаково ли значение операндов, если НЕ одинаковы – то условие является истиной	<code>a != b</code>	<code>True</code>
<code>></code>	проверяет значение левого операнда, если он больше, чем правый – то условие является истиной	<code>a > b</code>	<code>True</code>
<code><</code>	проверяет значение левого операнда, если он меньше, чем правый – то условие является истиной	<code>a < b</code>	<code>False</code>
<code>>=</code>	проверяет значение левого операнда, если он больше или равен правому – то условие является истиной	<code>a >= b</code>	<code>True</code>
<code><=</code>	проверяет значение левого операнда, если он меньше либо равен правому – то условие является истиной	<code>a <= b</code>	<code>False</code>



Блок кода

Блок кода - (также говорят блок команд, блок инструкций) в программировании — это логически сгруппированный набор идущих подряд инструкций в исходном коде программы.

В Python блок кода задается при помощи отступов: пробелами или символами табуляции.

Инструкция - наименьшая автономная часть языка **программирования**; команда или действие.

```
if paddle.colliderect(ball):
```

```
    ball_xdir *= -1  
    ball_ydir *= -1
```

← Блок кода

Операторы ветвления

Условный оператор if - называемый также оператором выбора, позволяет в зависимости от истинности или ложности некоторого условия выполнять ту или иную последовательность команд в блоке кода.

```
age = int(input("Введите возраст: "))
```

```
if age >= 18:  
    print("Совершеннолетний")
```

```
print("Пока!")
```

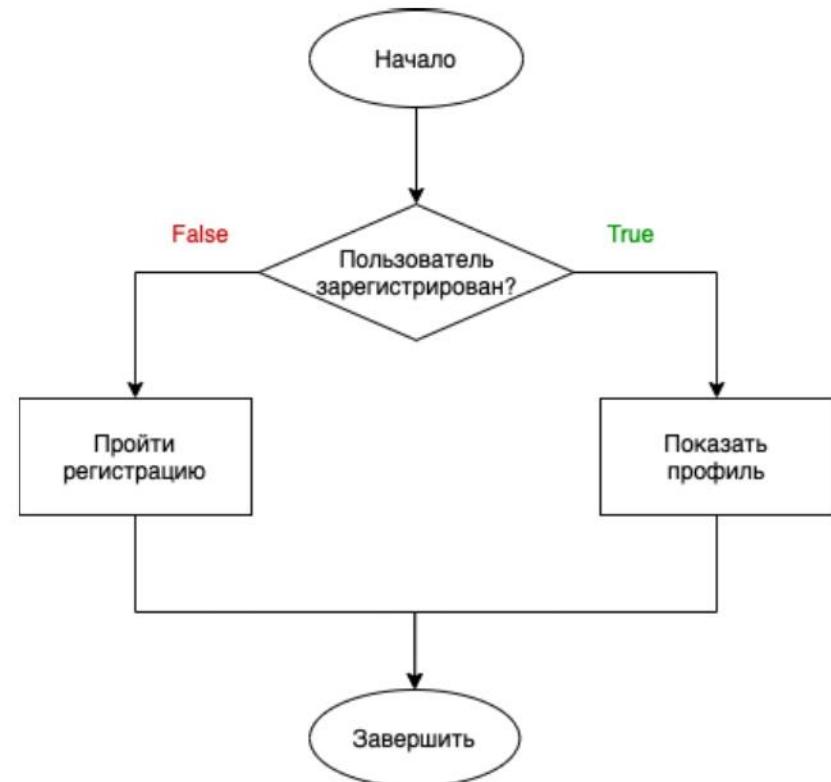


Оператор if-else

Оператор **if** выполняет блок кода, если указанное условие выполняется (истинно). Если условие не выполняется (ложно), то в противном случае выполняется блок кода внутри **else**.

```
registered = True
```

```
if registered:  
    print("Показать профиль")  
else:  
    print("Пройти регистрацию")  
  
print("До встречи!")
```





Задачи

Определить равенство трех чисел

1. На входе получить три числа и сохранить в переменные x , y , z
2. Если x равно y равно z , то сообщить о равенстве

Иначе сообщить о неравенстве

Определить число четное или нечетное?

1. Пользователь вводит целое число
2. Если число делится на 2 без остатка, то сказать четное

Иначе нечетное

Проверить делимость одного целого числа на другое

1. Пользователь вводит числа a и b
2. Если a делится на b без остатка, то число a делится нацело на b

Иначе, в остальных случаях, есть остаток и он не равен нулю.

Следовательно, a не делится на b

Программа для расчета площади и длины окружности круга

1. Пользователь вводит радиус
2. Если радиус больше или равен 0, то

Вывести чему равна длина окружности и площадь круга

Иначе

Попросить ввести положительное число

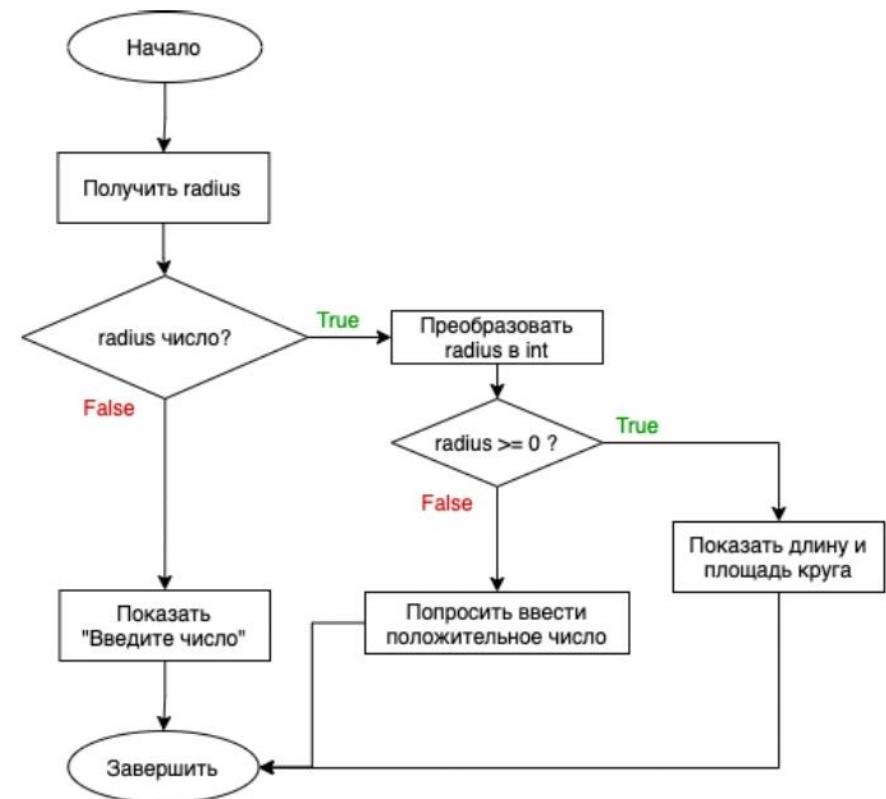
Оператор if внутри другого if

Если требуется больше проверок, то можно один оператор if поместить внутри другого оператора if

```
input_radius = input("Введите радиус: ")

if input_radius.isdigit():
    radius = int(input_radius)

    if radius >= 0:
        print("Длина окружности = ", 2 * 3.14 * radius)
        print("Площадь = ", 3.14 * radius ** 2)
    else:
        print("Введите положительное число")
else:
    print("Введите число")
```





Задачи

Задача: определить оценки студента на основе введенных баллов

Пользователь вводит оценку

Если оценка ≥ 90

Вывод: "Ваша оценка А (отл)"

Иначе

Если оценка ≥ 80

Вывод: "Ваша оценка В (хор)"

Иначе

Если оценка ≥ 70

Вывод: "Ваша оценка С (средне)"

Иначе

Если оценка ≥ 80

Вывод: "Ваша оценка D (плохо)"

Иначе

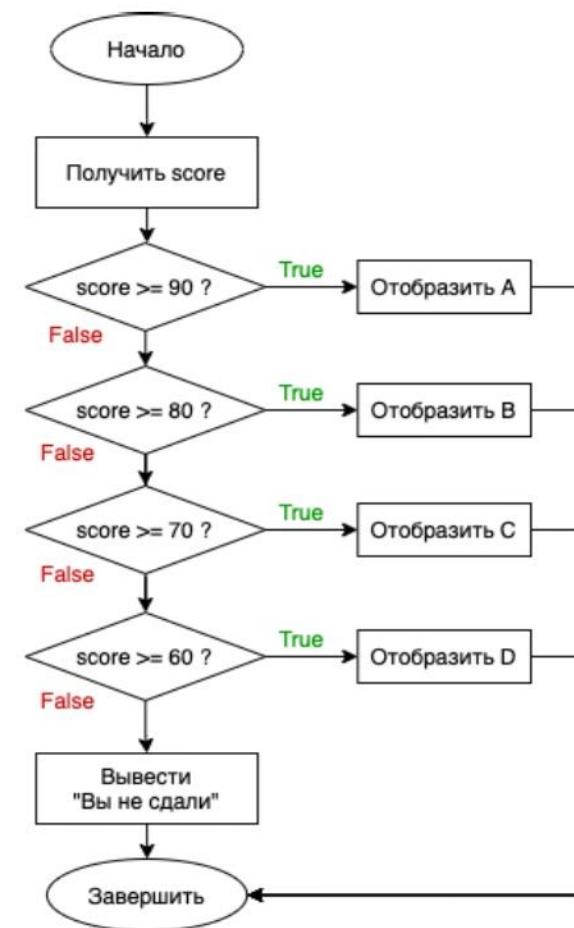
Вывод: "Вы не сдали"

Оператор if-elif-else

Оператор if-elif-else — это альтернативное представление оператора if-else, которое позволяет проверять несколько условий, вместо того чтобы писать вложенные if-else.

```
score = int(input("Введите вашу оценку: "))

if score >= 90:
    print("Ваша оценка А (отл)")
elif score >= 80:
    print("Ваша оценка В (хор)")
elif score >= 70:
    print("Ваша оценка С")
elif score >= 60:
    print("Ваша оценка D")
else:
    print("Вы не сдали экзамен")
```





Задачи

Задача: Расчет массы, плотности или объема

Пользователь вводит один символ: m - расчет массы, d - плотности или v - объема

Если 'm'

Запросить плотность и объем

Вычислить по формуле $m = dv$

Сохранить результат в переменной result

Иначе если 'd'

Запросить массу и объем

Вычислить по формуле $d = m/v$

Сохранить результат в переменной result

Иначе

Запросить массу и плотность

Вычислить по формуле $v = m/d$

Сохранить результат в переменной result

Отформатировать вывод оставив два знака после запятой

Вывести отформатированное значение переменной result



Логические операторы

Логические операторы принимают на входе логические значения и возвращают True или False. Позволяют объединить несколько "простых условий" в более "сложное".

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True



Задачи

Задача: Определить трудоспособный возраст

1. Пользователь вводит возраст
2. Если возраст меньше 18 или больше 60

Вывести: Вам нельзя работать

Иначе

Вывести: Вы можете работать

Задача: В какой четверти находится точка?

Пользователь вводит координаты точки x, y

Если $x > 0$ и $y > 0$:

```
print("Точка в I четверти")
```

Иначе если $x < 0$ и $y > 0$:

```
print("Точка во II четверти")
```

Иначе если $x < 0$ и $y < 0$:

```
print("Точка во III четверти")
```

Иначе если $x > 0$ и $y < 0$:

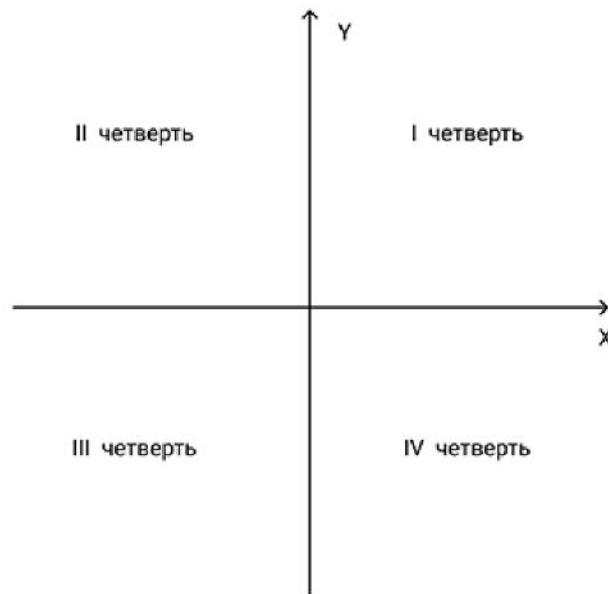
```
print("Точка во IV четверти")
```

Иначе если $x == 0$:

```
print("Точка на оси X")
```

Иначе если $y == 0$:

```
print("Точка на оси Y")
```





Введение в строки

Тип данных - множество значений и операций над этими значениями

Строка - объект имеющий тип данных str. Значением строки является последовательность одного или более символов, помещенных в кавычки. Строки являются неизменяемым типом данных.

```
s1 = "Строка в двойных кавычках"  
s2 = 'Строка в одинарных кавычках'  
multi_s = """
```

Это многострочная
строка

```
"""
```



Доступ к символам

Индекс - число, представляющее позицию элемента в строке.

Доступ к каждому символу в строке производится при помощи индекса.

```
s = "Hello World!"
```

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		W	o	r	l	d	!

```
print(s[0]) # -> "H"  
print(s[1]) # -> "e"  
print(s[3]) # -> "l"  
print(s[6]) # -> "W"
```



Строковые операторы

Оператор	Операция	Пример	Результат
+	Конкатенация	"Hello" + "World"	"HelloWorld"
*	Умножение	"Hello" * 3	"HelloHelloHello"
in	Проверка на вхождение	"cat" in "my cat"	True



Управляющие символы

Кодировка - это набор числовых значений, которые ставятся в соответствие группе алфавитно-цифровых символов, знаков пунктуации и специальных символов. В Python 3 по умолчанию используется кодировка utf-8

Управляющие символы - символы в кодировке, которым не приписано графическое представление, но которые используются для управления устройствами, организации передачи данных и других целей.

Символ	Результат
\n	Перевод каретки на новую строку
\b	Возврат каретки на один символ назад
\t	Горизонтальная табуляция
\f	Перевод каретки на новую страницу
\r	Возврат каретки на начало строки
\v	Вертикальная табуляция
\u, \U	16-битовый и 32-битовый символ Unicode



Экранирование управляемых символов

Экранирование - замена в тексте управляемых символов на соответствующие текстовые подстановки.

Пример	Результат
'I\'m super'	I'm super
"Hello \"Mi\"!"	Hello "Mi"!
"C:\\user"	C:\user
"Hello\\nWorld"	Hello\nWorld



Задачи

1. Напишите программу Python, которая отображала бы ваши данные, такие как имя, возраст, адрес, в трех разных строках. Например, чтобы она вывела:

Name: Alex

Age: 25

Address: Moscow, Tverskaya 1



Понятие объект, метод и свойство

Python — объектно-ориентированный язык программирования.

Почти все в Python — это объект с его атрибутами и методами.

Объект - это способ организации данных и кода в процессе работы программы, а также способ разделения сложных задач на более простые, что облегчает их решение. Более подробно понятие объекта будет рассмотрено в отдельном модуле.

Атрибут - предназначен для хранения данных внутри объекта. Проще говоря - это переменная, которая существует только вместе с объектом.

Метод - это функция, которая принадлежит объекту определенного типа данных. Вызов метода начинается с имени объекта, потом ставится точка и указывается имя метода с последующим вызовом при помощи круглых скобок.

Метод	Результат
"Ivan petrov".title()	"Ivan Petrov"
"Hello".upper()	"HELLO"
"PC".lower()	"pc"
"hello world".capitalize()	"Hello world"



Основные строковые методы

```
s = "hello world!"  
city = " moscow "
```

Функции и методы	Описание	Результат
s.upper()	В верхний регистр (сделать прописными)	"HELLO WORLD!"
s.lower()	В нижний регистр (сделать строчными)	"hello world!"
s.capitalize()	Сделать первую букву прописной	"Hello world!"
city.strip()	Удаление пробельных символов в начале и конце строки	"moscow"
s.title()	Сделать каждое слово с заглавной буквы	"Hello World!"
len(s)	Найти длину строки	12
s.isdigit(s)	Является ли строка числом	False
s..isalpha(s)	Не является ли числом	True



Задачи

1. Напишите программу на Python для вычисления суммы трех заданных чисел, если значения равны, верните трехкратную их сумму, то сумму увеличенную в 3-и раза
2. Напишите программу Python для суммы трех заданных целых чисел x, y, z. Однако, если два значения равны, сумма будет равна нулю, например, если x == y или x == z, то сумма равна нулю.
3. Напишите программу на Python для преобразования расстояния в метрах в следующие величины: в дюймы, ярды, мили и футы. На входе программы предложите пользователю выбрать величину, в которую будет выполнено преобразование.

1м = 39,3701 дюйма = 1,093614 ярда = 0,0006214 мили = 3,281 фута

4. Напишите программу на Python, чтобы проверить, является ли число положительным, отрицательным или нулевым.
5. Напишите программу Python для нахождения суммы двух заданных целых чисел. Однако, если сумма попадает в диапазон от 15 до 20, она должна вернуть 20.
6. Напишите программу, которая проверяет, является ли введеное занчение строкой с символами алфавита или числом. Если строка содержит число, определите какой тип данных имеет число и отобразите пользователю.



Входит в ГК Аплана



АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning
и очное
обучение



Ежегодные награды
Microsoft,
Huawei, Cisco и
другие

Направления обучения:

Информационные технологии

Информационная безопасность

ИТ-менеджмент и управление проектами

Разработка и тестирование ПО

Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск,
Хабаровск, Красноярск, Тюмень, Нижний
Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис
в Москве

Ресурсы более 400
высококлассных
экспертов и
преподавателей

Разработка
программного
обеспечения и
информационных
систем

Программы по
импортозамещению

Сеть региональных учебных центров
по всей России

Крупные заказчики



100+
сотрудников



АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:

Москва, Варшавское шоссе 47, корп. 4, 7 этаж

Тел: +7 (495) 150-96-00

academy@it.ru

academyit.ru