Конспект по теме «Знакомство с Python»

Почему мы выбрали Python для этого интенсива?

- Python простой

Синтаксис Python легко выучить за очень короткое время и можно быстро приступить к созданию программы.

- Python популярный

Разработчики любят его и продолжают выбирать для новых проектов, так что без работы не останетесь.

- Python универсальный

Используется в самых разных сферах. Даже если вы не будете разработчиком, он может вам пригодиться.

Пример программы на языке Python:

```
sum = 0
array = [45, 7, -934, 0, 2839]
for i in array:
    sum += i
print(sum)
```

Как запускаются программы на Python?

Языки программирования по способу преобразования текста программы в машинный код (инструкции, которые может понимать и выполнять компьютер) делятся на компилируемые и интерпретируемые.

Текст программы на компилируемом языке программирования сначала полностью преобразовывается специальной программой (компилятором) в машинный код.
На выходе получается файл специального формата (исполняемый).
Именно его и выполняет компьютер.

Текст программы на интерпретируемом языке программирования сразу выполняется специальной программой (интерпретатором) следующим образом: каждая команда обрабатывается поочередно, после чего сразу выполняется эквивалентный ей набор команд на машинном языке.

Python является интерпретируемым языком. Для запуска программы на вашем компьютере достаточно простого текстового редактора, где вы будете писать программный код, и интерпретатора Python, который будет выполнять эту программу.
В рамках данного интенсива мы предлагаем пользоваться онлайн-ресурсами: https://repl.it/ (первая часть курса) и https://www.pythonanywhere.com (вторая часть курса)

Программирование на Python

Программу на любом языке программирования, в том числе и на Python, можно представить как алгоритм действий над некоторым набором данных. На этой лекции речь пойдет про данные и про работу с данными в Python.

Для вывода информации на экран в Python существует функция:

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

В скобках мы передаем функции то, что хотим распечатать на экране.
В этом примере распечатывается строка: `Hello, world!`

```
print(34)
```

Распечатается число `34` .

Для того, чтобы внести данные в программу извне, существует функция `input().`

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

При запуске данного примера на экране компьютера мы увидим подсказку “Введите Ваш возраст”, а затем - поле для ввода данных. Введенное в это поле значение будет сохранено в переменную `age` .

Что такое переменная?

Переменная - это область памяти компьютера, выделенная для хранения информации. Переменную можно сравнить с ящиком или коробкой: имя переменной - это название коробки, а значение переменной - это то, что хранится в этой коробке.

```
age = 30
```

Простые типы данных

Числа: типы int и float

Все объекты в Python принадлежат какому-то классу или, другими словами, у всех данных есть свой определенный тип. Чтобы узнать тип данных, можно воспользоваться следующим кодом:

```
1 print(type(5))
  print(type(5.8))
```

В первом случае будет выведен результат `<class 'int'>` , а во втором - `<class 'float'>` . Целые числа принадлежат классу `int` , дробные - классу `float` .

Также можно определить тип переменной, то есть того значения, которое хранится в этой переменной.

```
1 a = 5
2 print(type(a))
3 a = 8.9
  print(type(a))
```

В первом случае будет распечатан тип `int` , во втором - `float` .

Операции с числами

Над объектами из классов `int` и `float` можно производить стандартные арифметические операции:

```
1 print(5+7)
2 print(10-4)
```

В Python есть 3 типа делений:

```
print(17/2) #обычное деление - результат 8.5
print(17//2) #целочисленное деление - результат 8, целая часть при делении
print(17%2) #деление с остатком - результат 1, остаток при делении
```

Обратите внимание на текст после знака `#`. Так в Python обозначаются комментарии. Эта часть кода не обрабатывается интерпретатором, она нужна для вас и других разработчиков, которые будут работать с кодом.

Еще один вариант комментариев в коде:

```
'''Различные варианты деления в Python
обычное деление
целочисленное деление
деление с остатком'''
```

Возведение в степень:

```
print(3**6)
print(10**(-1))
print(25**(1/2))
```

Результат выполнения этой программы:

```
729
0.1
5.0
```

Строки

Мы уже рассматривали строку `"Hello, world!"`. Строки относятся к классу `str`.

```
print(type("Hello, world!"))
```

Строки тоже можно складывать. При сложении они склеиваются.

```
str1 = 'Hello, '
str2 = 'world!'
print(str1+str2)
```

Результат: `“Hello, world!”`

Строку можно умножить на число:

```
print(str1*5)
```

Результат - это повторение строки `str1` 5 раз.

Часто нам будет полезно уметь определять длину строки. Для этого существует функция `len( )`:

```
print(len('строка'))
```

Кстати, строку можно обрамлять как одинарными кавычками, так и двойными.

как с числами, то можно воспользоваться приведением типов:

```
a = int(input())
b = float(input())
```

Логический тип данных

В Python существует логический тип данных - класс `bool.` Переменные этого типа могут принимать только 2 значения: `True` и `False` . Подробнее с этим типом данных мы столкнемся, когда будем рассматривать логические выражения и условные конструкции.

```
print(True)
print(False)
print(type(True))
print(type(False))
```

Тип данных None

В Python существует зарезервированное слово `None` , которое обозначает отсутствие типа.

```
a = None
print(type(a))
```

Составные типы данных

Списки

Совокупность объектов в Python представляется в виде структур данных.

Список - это упорядоченная и изменяемая коллекция объектов произвольного типа.

```
array = [1, 2, 3, 4, 5, 6, 7] # список из целых чисел int
print(type(array)) # класс list
ar = [2, 4.6, 'str', [1, 2, 3]] # список, состоящий из целого числа,
# из числа с плавающей точкой, из строки и из списка
```

Обратиться к элементу списка мы можем по его индексу. Нумерация элементов начинается с 0.

```
print(array[0], array[1]) # печатаем 2 первых элемента в списке: 1 и 2
array[0] = 0 # в нулевой элемент списка кладем цифру 0
```

Мы можем расширить список, добавить в конец списка новый элемент:

```
array.append(8)
```

В список `array` добавили элемент 8 с помощью метода `append` .

Списки можно складывать (склеивать):

```
array2 = [9, 10]
new_array = array + array2
```

`new_array = [0, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

Можно определить длину списка `new_array` :

```
len(new_array)
```

```
sum(new_array)
```

Сейчас наш список `new_array` выглядит так `[0, 2, 3, 4, 5, 6, 7, 8, 9, 10]`. Предположим, что мы хотим добавить в список новый элемент 1. Мы можем добавить его в конец с помощью метода `append`. Но мы хотим, чтобы 1 оказалась между 0 и 2. Мы можем сортировать наш список по возрастанию с помощью метода `sort`.

```
1 new_array.append(1)
2 print(new_array)
3 new_array.sort()
```

В первом случае распечатается список `[0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1]`.
А во втором - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`.

Словари

Еще одна распространенная структура данных в Python - это словари.

Словарь - это неупорядоченные коллекции произвольных объектов с доступом по ключу.

Пример словаря:

```
dictionary = {'dog' : 'собака', 'table' : 'стол', 'computer': 'компьютер'}
```

В данном словаре ключами являются слова на английском языке, а значения - слова на русском языке.

Мы можем обращаться к значениям словаря по ключу.

```
1 print(dictionary['dog']) # печатаем строку 'собака'
2 dictionary['dog'] = 'пес' # изменяем значение 'собака' на 'пес'
3 dictionary['laptop'] = 'ноутбук' # добавляем новый элемент с ключом
  # 'laptop' и значением 'ноутбук' в словарь
```

Вам понравилось?