Основы языка программирования Python

Подготовил:
Кудряшов А.С
Группа: 9/1-РПО-24/1

Кострома, 2024

Содержание

- Что такое Python
- Преимущества Python
- Основные свойства и возможности Python
- Структуры данных языка Python
- Строки в языке Python
- Операторы в языке Python
- Функции языка Python
- Классы языка Python
- Исключение в языке Python
- Импорт в языке Python
- Заключение

Что такое Python

Python – это высокоуровневый язык программирования, который был разработан в конце 1980-х годов. Его разработчик, Гвидо ван Россум, вложил в основу языка простоту и читабельность кода, что позволяет использовать Python для быстрой и эффективной разработки. Много популярных веб-сайтов, компьютерных игр и программ, написанных на Python, вы используете ежедневно: Dropbox, Uber, Sims, Google, GIMP и другие.

Язык отличается понятным синтаксисом, поэтому Python подходит для начинающих программистов. Он широко используется во многих областях: веб-разработка, научные исследования, анализ данных, искусственный интеллект, машинное обучение, разработка игр.

Преимущества Python

1. Простой и читаемый код. Python предлагает понятный синтаксис, что делает его привлекательным для опытных разработчиков и доступным для новичков.
2. Большое число полезных библиотек и модулей для Python позволяет быстро и легко решать различные задачи, такие как обработка данных, машинное обучение, работа с базами данных.
3. Язык подходит для большинства операционных систем. Код, написанный при помощи Python, может быть запущен на популярных ОС: Windows, macOS, Linux.
4. Python позволяет легко интегрировать код на других языках, таких как C ++ и Java. Это позволяет использовать уже существующий код и библиотеки на этих языках, чтобы расширять функциональность Python.
5. Активное сообщество разработчиков помогающее и поддерживающее новичков. Это значит, что всегда можно получить ответы на возникающие вопросы или найти готовый код для решения своих задач.

Основные свойства и возможности Python

1. Python интерпретируемый язык программирования – код на нем выполняется построчно, в режиме реального времени. Это свойство позволяет быстро исправлять и проверять код без необходимости компиляции.
2. Python является языком с динамической типизацией, то есть тип переменной определяется автоматически, во время выполнения кода. Это упрощает процесс программирования и делает его гибким при работе с данными различного типа.
3. Python поддерживает объектно-ориентированное программирование (ООП), что позволяет разрабатывать код в виде объектов, которые взаимодействуют друг с другом. Это делает код более модульным и повторно используемым.
4. Python поддерживает также императивное, функциональное и аспектно-ориентированное программирование. Таким образом, разработчики имеют возможность выбирать нужный подход для решения конкретной задачи.

Синтаксис языка Python

Синтаксис Python отличается своей простотой и ясностью. Рассмотрим основные элементы:

1. Вместо фигурных скобок или ключевых слов для обозначения блоков кода используются отступы. Отступ должен быть одинаковым и состоять из пробелов или табуляции.

Например:

```
if x > 5:
    print("x больше 5")
else:
    print("x меньше или равно 5")
```

Синтаксис языка Python

2. Комментарии начинаются с символа # и необходимы для пояснения кода или временного отключения определенных участков.

Например:

Комментарий

print("Hello!") # Комментарий после кода

3. Переменные объявляются присваиванием значения. Тип переменной определяется автоматически во время выполнения программы.

Пример:

```
x = 7  # переменная целого числа

y = 3.11  # переменная числа с плавающей запятой

name = "Ivan"  # переменная строки
```

Синтаксис языка Python

4. Python поддерживает все основные математические действия (сложение, вычитание, умножение, деление, равенство, неравенство, больше, меньше и др.).

Приведем пример:

```
x = 9

y = 5

sum = x + y

difference = x - y

product = x * y

quotient = x / y

print(sum, difference, product, quotient)  # выводит: 14, 4, 45, 1.8
```

Структуры данных языка Python

Python поддерживает следующие структуры данных:

1. Списки – упорядоченные коллекции элементов, которые могут быть любого типа данных. Доступ к элементам списка осуществляется по индексу.
2. Кортежи – упорядоченные коллекции элементов, которые могут быть любого типа данных, но не могут быть изменены после создания.
3. Словари – неупорядоченные коллекции пар "ключ-значение". Доступ к значениям словаря осуществляется по ключу.
4. Множества – неупорядоченные коллекции уникальных элементов.
5. Стеки и очереди – для реализации стеков и очередей используются списки.
6. Генераторы – функции, которые используются для генерации последовательностей значений.
7. Итераторы – объекты, которые позволяют проходить по последовательности значений, например, списку или словарю.
8. Файлы – структура данных, которая используется для чтения и записи информации на диске.

Строки в языке Python

При оформление строк в Python могут быть использованы различные методы и синтаксические возможности, например:

1. Кавычки. Строки в Python могут быть оформлены с использованием одиночных кавычек (''), двойных кавычек ("") или тройных кавычек (''' ''', """ """).
2. Чтобы использовать специальные символы, например, одинарная или двойная кавычка, внутри строки, используется обратный слеш (\).

```
message = "She said, \"Hello!\""
```

3. Для объединения строк используется оператор "+". Это называется конкатенацией строк.

```
name = "Ivan"

greeting = "Hello, " + name + "!"
```

4. Python версии 3.6 и выше поддерживает синтаксис интерполяции строк, который позволяет вставлять значения переменных непосредственно в строку с помощью фигурных скобок {} и f-префикса.

Строки в языке Python

5. Для форматирования строк используется метод format() и оператор %.

Метод format():

```
name = "Ivan"
age = 25
message = "My name is {} and I'm {} years old.".format(name, age)
```

Оператор %:

```
name = "Ivan"
age = 25
message = "My name is %s and I'm %d years old." % (name, age)
```

Операторы в языке Python

Операторы в языке программирования используются для выполнения различных операций, например, арифметические вычисления, сравнение значений, присваивание, логические операции и т. д.

- Оператор присваивания (=) используется для присваивания значения переменной.
- Оператор арифметических вычислений (+, -, *, /) используются для выполнения арифметических операций.
- Оператор сравнения (==, !=, >, >=, <, <=) используется для сравнения двух значений.
- Оператор логического И (and), ИЛИ (or), НЕ (not) используются для комбинирования условий в логические выражения.
- Оператор ветвления if-else используется для выполнения определенного блока кода в зависимости от условия.

Пример: if x > 6:

```
print("x is greater than 6")

 else:

     print("x is less than or equal to 6")
```

Операторы в языке Python

- Оператор цикла for используется для выполнения определенного блока кода для каждого элемента в указанной последовательности.

Пример:

```
for i in range(5):
    print(i)
```

- Оператор цикла while используется для выполнения определенного блока кода, пока указанное условие истинно.

Пример: while x < 11:

```
    print(x)
        x += 2
```

- Оператор ветвления try-except используется для обработки исключений в коде.

Пример: try:

```
    x = 11 / y
     except ZeroDivisionError:
        print("Error: can't divide by zero")
```

Функции языка Python

Встроенные функции Python используются для решения различных задач. Примеры некоторых из них:

– ` print()` – для вывода текста или переменных на консоль.

– `input()` – запрашивает ввод данных от пользователя.

– `len()` – возвращает длину объекта (например, строки, списка или кортежа).

– `type()` – возвращает тип объекта.

– `range()` – создает последовательность чисел.

– `int()`, `float()`, `str()`, `bool()` – используются для преобразования объектов в целые числа, числа с плавающей запятой, строки и булевые значения соответственно.

– `sum()` – суммирует элементы последовательности чисел.

– `sorted()` – сортирует элементы последовательности.

– `max()` и `min()` – возвращают максимальный и минимальный элементы последовательности.

– `abs()` – возвращает абсолютное значение числа.

– round()` – Округляет число до указанного количества знаков после запятой.

– `str.upper()`, `str.lower()` – преобразуют все символы в строке в верхний или нижний регистр соответственно.

Классы языка Python

Классы в Python позволяют определять объекты с их собственными свойствами и методами. Они могут иметь атрибуты (переменные) и методы (функции), которые могут быть использованы в экземплярах этого класса. Классы также поддерживают наследование, позволяющее создавать новые классы на основе существующих.

Пример создания класса в Python:

```
class Person:

  def __init__(self, name, age):

    self.name = name

    self.age = age

  def greet(self):
```

Исключение в языке Python

Исключения в Python – тип данных, с помощью которых разработчик узнает об ошибках и необычных ситуациях. Примеры исключений в Python:

1. TypeError означает, что операция выполняется над объектом несовместимого типа данных.
2. ValueError возникает, когда функция получает аргумент правильного типа, но с неправильным значением.
3. IndexError появляется, когда происходит попытка получить доступ к элементу списка или строки с использованием недопустимого индекса.
4. KeyError – при попытке получить доступ к элементу словаря по ключу, которого нет в словаре.
5. FileNotFoundError – возникает, при открывании файла, которого не существует.
6. ZeroDivisionError – деление на ноль происходит в программе.
7. IOError – появляется, когда возникают проблемы с вводом-выводом, например, при чтении или записи файлов.

Вы можете создавать собственные исключения в Python, отталкиваясь от встроенных классов исключений.

Импорт в языке Python

Импорты используются для подключения модулей или библиотек.

- Импорт всего модуля: import module_name
- Импорт модуля с псевдонимом: import module_name as alias_name
- Импорт конкретной функции или класса из модуля:

from module_name import function_name

from module_name import class_name

- Импорт нескольких функций или классов из модуля:

from module_name import function_name1, function_name2

from module_name import class_name1, class_name2

- Импорт всего модуля со всеми его функциями и классами: from module_name import *

Обратите внимание, на корректность указания имени модуля и функции и их доступность для импорта.

Заключение

Python – это мощный и, в то же время, легкий в использовании язык программирования, который позволяет создавать разнообразные проекты и находить решения для большого спектра задач.

Его отличает простота и универсальность, что позволяет применять Python в различных областях, в том числе и начинающими программистами. Большое сообщество разработчиков обеспечивает помощь и поддержку при работе с языком.

Возможность запускать Python на различных операционных системах позволяет создавать приложения и скрипты, которые могут быть запущены в разных окружениях без необходимости переписывать код. Дополнительные инструменты и библиотеки позволяют использовать Python для анализа данных, машинного обучения, веб-разработки и во многих других областях.