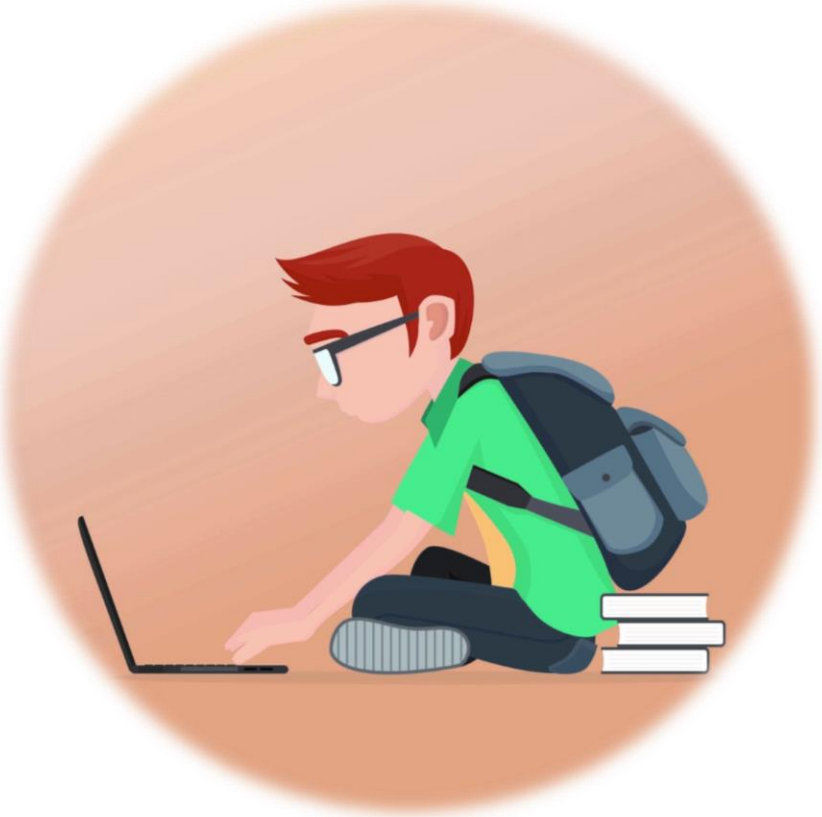
ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON

Основные особенности языка Python

Крашенинников Роман Сергеевич
Главный специалист отдела системного администрирования РХТУ им. Д.И. Менделеева, ведущий программист кафедры информационных компьютерных технологий

ТЕМЫ

- Ключевые особенности языка **Python**
- Области применения языка
- Популярные среды для разработки на языке Python
- Основные типы переменных
- Простейшие операции над переменными
- Логические операции
- Конструкции **if**, **else**, **elif**
- Списки
- Циклы **while** и **for**



ПОЛЕЗНЫЕ РЕСУРСЫ

- https://www.python.org/
- https://docs.python.org/
- https://pythonworld.ru/
- https://www.anaconda.com/
- https://jupyterlab.readthedocs.io/
- https://www.spyder-ide.org/
- https://www.codewars.com/

КЛЮЧЕВЫЕ ОСОБЕННОСТИ ЯЗЫКА PYTHON

Python – это высокоуровневый кроссплатформенный интерпретируемый язык программирования общего назначения, появившийся порядка 30 лет тому назад. Он ориентирован на повышение эффективности разработчика и читаемости кода. Синтаксис у данного языка минималистичен, и сосредоточен на сокращении программного кода при помощи коротких языковых конструкций, а стандартная библиотека включает большой объем поддерживаемых функций.

Python способен поддерживать множество парадигм программирования, включая объектно-ориентированное, императивное, структурное и функциональное программирование. Если говорить об основных чертах языка, то следует отметить динамическую типизацию, автоматическое управление памятью, механизм обработки исключений, а также полную интроспекцию. Помимо этого, Python поддерживает многопоточность и способность работать с высокоуровневыми структурами данных. Кроме того, в Python имеется возможность разбивать программы на отдельные модули, их же в свою очередь в дальнейшем можно объединять в пакеты.

КЛЮЧЕВЫЕ ОСОБЕННОСТИ ЯЗЫКА PYTHON

Преимущества:

- Качество программного обеспечения
- Высокая скорость разработки
- Динамическая типизация
- Переносимость программ
- Поддержка библиотек

Недостатки:

- Необходимость установки интерпретатора
- Относительно низкая скорость выполнения программ
- Сложности с динамической типизацией

КЛЮЧЕВЫЕ ОСОБЕННОСТИ ЯЗЫКА PYTHON

Python активно развивается и является одним из самых прогрессивных языков программирования. Новые версии выходят с частотой примерно раз в два с половиной года. (текущая версия 3.10.7) Для сравнения, Python является 4-м по популярности на портале StackOverflow, который создан для обсуждения вопросов, касающихся программирования. На данном ресурсе задано уже более 500 тысяч вопросов по данному языку. Кроме того, этот язык является 4-м по использованию на крупнейшем портале для хранения репозиториев GitHub. Такая популярность в первую очередь обусловлена широкими возможностями, которые предоставляет Python своим разработчикам.

Python является одним из самых быстрорастущих языков программирования и поданным аналитики это может быть обусловлено многими факторами. Однако, основополагающий фактор – это его высокая применимость в науке, благодаря обширному числу библиотек по данной тематике, а также низкому порогу вхождения в данный язык.

ОБЛАСТИ ПРИМЕНЕНИЯ ЯЗЫКА

ПОПУЛЯРНЫЕ СРЕДЫ ДЛЯ РАЗРАБОТКИ НА ЯЗЫКЕ PYTHON

Поддерживают **Python**

Visual Studio Code

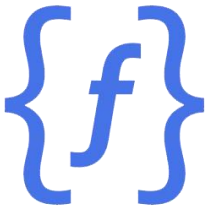
Eclipse

Созданы для **Python**

PyCharm

Anaconda

Онлайн работа с **Python**



Yandex Cloud Functions

Google Collab

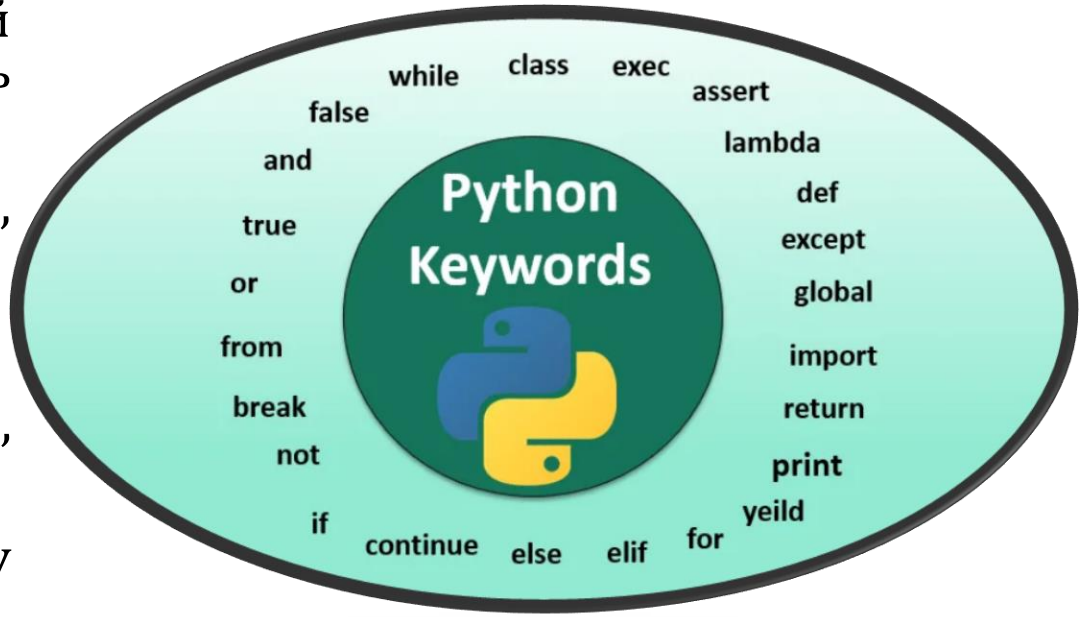
ОСНОВНЫЕ ТИПЫ ПЕРЕМЕННЫХ

Название	Тип	Пример и краткое описание
Integer	int	3, 79, 0, -35 - целые числа
Floating point	float	7.98, -0.905, 1.4 - упорядоченная последовательность символов
String	str	"hello", 'Ivan', "2345" - упорядоченная последовательность символов
List	list	["Hello", 55, 26.6, "Hello"] - упорядоченная последовательность объектов
Dictionary	dict	{“name”: “Ivan”, “age”: 20} - упорядоченная последовательность пар ключ-значение
Tuple	tup	("Hello", 55, 26.6, "Hello") - упорядоченная неизменяемая последовательность объектов
Boolean	bool	True, False - логический тип данных

ОСОБЕННОСТИ ИМЕНОВАНИЯ

Переменные в Python не требуют объявления типа переменной (так как Python – язык с динамической типизацией) и являются ссылками на область памяти. Правила именования переменных:

- Имя переменной может состоять только из букв, цифр и знака подчёркивания;
- Имя не может начинаться с цифры;
- Имя не может содержать специальных символов @, $, %;
- Имя переменной не должно равняться ключевому слову;
- Желательно, чтобы имя было «говорящим»



ПРОСТЕЙШИЕ ОПЕРАЦИИ НАД ПЕРЕМЕННЫМИ

Оператор	Описание	Пример	Результат
+	Сложение	7 + 3	10
-	Вычитание	7 - 3	4
*	Умножение	7 * 3	21
/	Деление (истинное)	7 / 3	2.3333333333333335
**	Возведение в степень	7**3	343
//	Целочисленное деление	7 // 3	2
%	Остаток от деления	7 % 3	1

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Python	Математика	Значение
<	<	Меньше чем
<=	≤	Меньше или равно чем
==	=	Равно
>=	≥	Больше или равно чем
>	>	Больше чем
!=	≠	Не равно

Оператор	Описание
and	Логический оператор "И". Условие будет истинным, если оба операнда истина
or	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным
not	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное

КОНСТРУКЦИИ IF, ELSE, ELIF

Простейший случай с **if**

```
if <условие>:
    <тело оператора>
```

Вариант с **else**

```
if <условие>:
    <тело оператора>
else:
    <тело оператора>
```

Вариант с **elif**

```
if <условие>:
    <тело оператора>
elif <условие>:
    <тело оператора>
```

Реальный пример

```
data = 0
if data > 0:
    print(data, "is a positive number.")
elif data == 0:
    print("The data is:", data)
else:
    print(data, "is a negative number.")
```

СПИСКИ

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Генерировать списки можно различными способами. Можно использовать встроенную функцию **list**, литерал [] или же генератора списков.

Для обращения к элементу списка, номер элемента передается в []. Важно помнить, что в **Python** нумерация начинается с 0!

Создание при помощи **list**

```
l = list('список')
# ['с', 'п', 'и', 'с', 'о', 'к']
```

Создание при помощи []

```
l = []
s = [1, 'h' , False]
```

Создание при помощи генератора

```
l = [c for c in 'список']
# ['с', 'п', 'и', 'с', 'о', 'к']
```

Обращение к элементу

```
s = [1, 'h' , False]
print(s[1])
```

Цикл while

Цикл while (“пока”) позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно.

Условие записывается до тела цикла и проверяется до выполнения тела цикла.

Как правило, цикл while используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла **while**

```
1 while <условие>:
2     <тело цикла>
```

Пример цикла **while**

```
a = 0
while a < 5:
    a = a + 1
    print(a)
```

Цикл for

Цикл for, также называемый циклом с параметром, в языке Питон богат возможностями.

В цикле for указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано списком, кортежем, строкой или диапазоном. В связке с **for** часто используют функцию **range**, создающую последовательность с определённым шагом.

Синтаксис цикла **for**

```
for <итератор> in <коллекция>:
    <тело цикла>
```

Пример цикла **for**

```
1 a = 0
2 for i in range(1, 5, 2):
3     a = a + i
4 print(a)  # Равно 4
```

Цикл while и for

Для программного выхода из цикла в языке **Python** используется слово **break.** Для того, чтобы пропустить итерацию цикла, необходимо использовать оператор **continue.**

Пропуск итерации

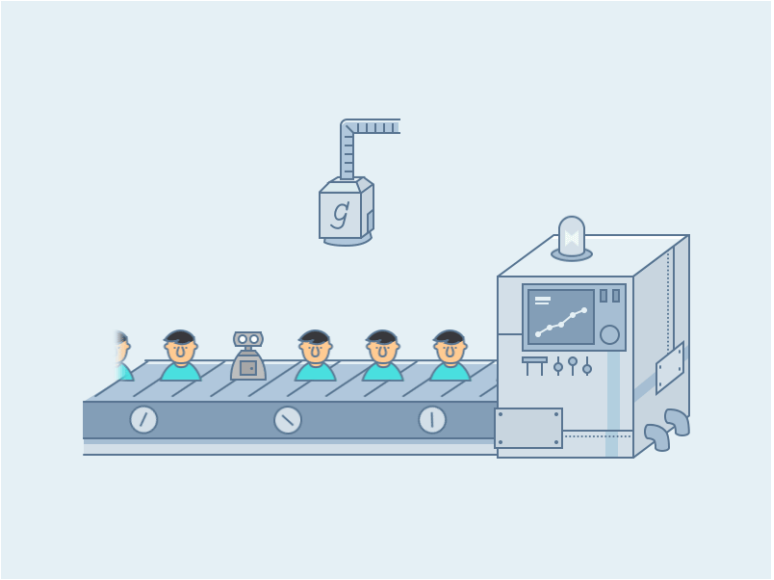
```
for i in range(1, 5):
    if i % 2:
        continue
    print(i)
```

Выход из цикла

```
i = 0
while True:
    if i == 5:
        break
    print(i)
    i += 1
```

break

continue



ЗАДАНИЕ ДЛЯ САМОКОНТРОЛЯ

- Создать программу на языке Python для решения квадратных уравнений, при этом при помощи управляющих конструкций if elif else, определять количество действительных корней уравнения.
- Создать программу, которая при помощи циклов, найдёт все простые числа в диапазоне от 1 до 200.
- Создать программу, которая будет подсчитывать процент гласных букв в строке, записанной на латинице.

СПАСИБО ЗА ВНИМАНИЕ!