ELABORATION.ASIA
INTERNATIONAL EDUCATION CENTRE

Урок 5:Поток управления

Уровень: начинающий
Курс: введение Python

Содержание

ELABORATION.ASIA
INTERNATIONAL EDUCATION CENTRE

- Добро пожаловать на этот урок по Control Flow! Поток управления - это последовательность выполнения кода. Здесь мы узнаем о нескольких инструментах в Python, которые мы можем использовать для воздействия на поток управления нашего кода:
- Условные операторы;
- Логические выражения;
- Циклы For и While;
- Операторы Break и Continue;
- Функция Zip и Enumerate;
- Генератора списка.

Создание списков

ELABORATION.ASIA
INTERNATIONAL EDUCATION CENTRE

- Помимо извлечения информации из списков, как мы делали в первом примере выше, вы также можете создавать и изменять списки с помощью циклов **for**.
- Вы можете **создать** список путем присоединения к новому списку в каждой итерации цикла **for** следующим образом:

```
# Creating a new list
cities = ['new york city', 'mountain view', 'chicago', 'los angeles']
capitalized_cities = []

for city in cities:
    capitalized_cities.append(city.title())
```

Изменение списков

- **Изменение** списка немного сложнее и требует использования функции **range ().**
  - Мы можем использовать функцию **range ()** для генерирования индексов для каждого значения в списке **городов**.
  - Это позволяет нам получить доступ к элементам списка с городами [индекс], чтобы мы изменили значения в списке **городов** на месте.

```
cities = ['new york city', 'mountain view', 'chicago', 'los angeles']

for index in range(len(cities)):
    cities[index] = cities[index].title()
```

Quiz: Create Usernames

- Где задание

Building Dictionaries

- Где задание

Итерация по словарям с циклами for

Через цикл for мы можем красиво вывести наш словарь

```
ab = { 'Elaboration' : ' elaboration.kz@gmail.com ',
        'Larry' : 'larry@wall.org',
        'Matsumoto' : 'matz@ruby-lang.org',
        'Spammer' : 'spammer@hotmail.com' }
for name, address in ab.items():
        print('Контакт {0} с адресом {1}'.format(name, address))
```

Итерация по словарям с циклами for

Мы обращаемся ко всем парам ключ-значение нашего словаря, используя метод items, который возвращает список кортежей, каждый из которых содержит пару элементов: ключ и значение. Мы получаем эту пару и присваиваем её значение переменным name и address соответственно в цикле for...in..., а затем выводим эти значения на экран в блоке for.

Циклы While

- Циклы **for** являются примером «определенной итерации», означающей, что тело цикла выполняется заданное количество раз.
- Это отличается от «неопределенной итерации», когда цикл повторяется неизвестное число раз и заканчивается, когда выполняется некоторое условие, что и происходит в цикле **while**. Вот пример цикла **while**.

```
card_deck = [4, 11, 8, 5, 13, 2, 8, 10]
hand = []

# adds the last element of the card_deck list to the hand list
# until the values in hand add up to 17 or more
while sum(hand)  < 17:
    hand.append(card_deck.pop())
```

- В этом примере представлены две новые функции. **sum** возвращает сумму элементов в списке, а **pop** - это метод списка, который удаляет последний элемент из списка и возвращает его.

Компоненты цикла While

1. Первая строка начинается с ключевого слова **while**, обозначая, что это цикл **while.**
2. Затем следует условие, которое нужно проверить. В этом примере это **sum(hand) <= 17.**
3. Заголовок цикла **while** всегда заканчивается двоеточием **:.**
4. После этого заголовка ставится тело цикла **while** с отступом. Если условие цикла **while** истинное, строки кода в теле цикла будут выполнены.
5. Затем мы возвращаемся к строке заголовка **while,** условие снова оценивается. Этот процесс проверки условия и последующего выполнения цикла повторяется до тех пор, пока условие не станет ложным.
6. Когда условие становится ложным, мы переходим к строке, следующей за телом цикла, которая будет с отступом.

- Тело цикла с отступом должно изменить хотя бы одну переменную в условии проверки. Если значение условия проверки никогда не изменяется, результатом является бесконечный цикл!

Quiz: Count By

задание

Циклы For против циклов While

- Теперь, когда вы знакомы с циклами **for** и **while**, давайте рассмотрим, когда нужно использовать каждый из них.
- циклы **for** наиболее подходят, когда **число итераций известно или ограничено**.
- Примеры:
  - Когда у вас есть итерируемая коллекция (список, строка, набор, кортеж, словарь)
    - **Для имени в именах:**
  - Если вы хотите выполнить обход цикла определенное количество раз, используя range()
    - **для i в range(5):**

Циклы For против циклов While

- Циклы **While** наиболее подходят, когда **итерации должны продолжаться, пока не будет выполнено условие**.
- Примеры:
  - Когда вы хотите использовать операторы сравнения
    - **While count <= 100:**
  - Когда вы хотите использовать оператора цикла на основе получения конкретного пользовательского ввода.
    - **While user_input == 'y':**

Функции Break, Continue

- Иногда нам нужно внимательнее контролировать момент, когда цикл должен закончиться, или пропустить итерацию.
- В этих случаях мы используем ключевые слова **break** и **continue**, которые можно использовать в циклах **for** и **while**.
  - Break завершает цикл;
  - Continue пропускает одну итерацию цикла.

Quiz: Break, Continue

1. Напишите программу, которая спрашивает у пользователя, в каких городах он бывал. Как только пользователь вводит ‘quit’ программа должна вывести весь список
2. Напишите цикл который считает от 1 до 10, но выводит только нечетные числа (используйте метод continue)

Функции Zip и Enumerate

ELABORATION.ASIA
INTERNATIONAL EDUCATION CENTRE

- **zip** и **enumerate - п**олезные встроенные функции, которые могут пригодиться при работе с циклами
- **Zip** возвращает итератор, который объединяет несколько итерируемых в одну последовательность кортежей. Каждый кортеж содержит элементы в этой позиции из всех итераций. Например, печать
- **список (zip([‘a’, ‘b’, ‘c’], [1, 2, 3]))** дает - **[(‘a’, 1),(‘b’, 2),(‘c’, 3)]**
- Как и в случае с **range(),** нам нужно преобразовать его в список или выполнить обход по нему с циклом, чтобы увидеть элементы.

Функции Zip и Enumerate

- Вы можете распаковать каждый кортеж в цикле **for** следующим образом:

```
letters = ['a', 'b', 'c']
nums = [1, 2, 3]

for letter, num in zip(letters, nums):
    print("{}: {}".format(letter, num))
```

- Помимо объединения двух списков вместе, вы также можете распаковать список в кортежи, используя звездочку.

```
some_list = [('a', 1), ('b', 2), ('c', 3)]
letters, nums = zip(*some_list)
```

- Это создаст те же **буквы** и **числа** кортежей, которые мы видели ранее.

Enumerate

- **enumerate** - это встроенная функция, которая возвращает итератор кортежей, содержащих индексы и значения списка.
- Вы будете часто использовать это, когда вам нужен индекс вместе с каждым элементом итерируемого в цикле.

```
letters = ['a', 'b', 'c', 'd', 'e']
for i, letter in enumerate(letters):
    print(i, letter)
```

- Этот код дает результат:

```
0 a
1 b
2 c
3 d
4 e
```

Генераторы списка

- В Python вы можете создавать списки очень быстро и кратко, используя генераторы списки. Этот пример ранее:

```
capitalized_cities = []
for city in cities:
    capitalized_cities.append(city.title())
```

- можно свести к:

```
capitalized_cities = [city.title() for city in cities]
```

- Генераторы списка позволяет нам создать список, используя цикл **for** за один шаг.
  - Вы создаете генератор списка в скобках [], включая выражение для оценки каждого элемента в итерируемом объекте. Этот генератор списка выше вызывает **city.title ()** для каждого элемента **город** в **городах**, чтобы создать каждый элемент в новом списке **capitalized_cities.**

Условия в генераторах списка

- Вы также можете добавить условные выражения в генераторы списков (listcomps). После итерируемого вы можете использовать ключевое слово **if** для проверки условия в каждой итерации.

```
squares = [x**2 for x in range(9) if x % 2 == 0]
```

- Код выше устанавливает **квадраты числа,** равные списку [0, 4, 16, 36, 64], поскольку x к степени числа 2 оценивается, только если x является четным.
- Если вы хотите добавить **else**, вы получите синтаксическую ошибку.

```
squares = [x**2 for x in range(9) if x % 2 == 0 else x + 3]
```

Условия в генераторах списка

- Если вы хотите добавить **else**, вы должны переместить условные выражения в начало генератора списка, сразу после выражения.

```
squares = [x**2 if x % 2 == 0 else x + 3 for x in range(9)]
```

- Генераторы списков не встречаются в других языках, но очень распространены в Python.

Quiz

1. Сгенерируйте список, заполненный квадратами целых чисел
2. Сгенерируйте список, заполненный квадратами четных чисел задом наперед
3. Выведите весь предыдущий список в формате ‘индекс-значение’