Lecture 4 – Python Loops

Mr. Yousif G. Arshak

University of Zakho

Computer Science Department

yousif.arshak@uoz.edu.krd

OUTLINES

- Python While Loops
  - `while` loops
  - `for` loops
- Python Functions

Python Loops

- Python has two primitive loop commands:
  - `while` loops
  - `for` loops
- The while Loop

  With the `while` loop we can execute a set of statements as long as a condition is true.

  Example: Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
Output
1
2
3
4
5
```

Note: remember to increment i, or else the loop will continue forever.

The break Statement

- With the `break` statement we can stop the loop even if the while condition is true:

  Example: Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
Output
1
2
3
```

The continue Statement

- With the `continue` statement we can stop the current iteration, and continue with the next:

Example: Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
Output
1
2
4
5
6
```

The else Statement

- With the `else` statement we can run a block of code once when the condition no longer is true:

  Example: Print a message once the condition is false:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
Output
1
2
3
4
5
i is no longer less than 6
```

Exercise:

- Print `i` as long as `i` is less than 6.
- Print a message once the condition is true.
- Continue to the next iteration if i is 5.
- Exit the loop when i is 8.

Python For Loops

- A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the `for` keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the `for` loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example: Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
Output:
apple
banana
cherry
```

The `for` loop does not require an indexing variable to set beforehand.

Looping Through a String

Example: Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

```
Output:
b
a
n
a
n
a
```

Exercise

- Print each day in a week list.
- Print each month in a year list.

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

Creating a Function

In Python a function is defined using the `def` keyword:

Example

```
def my_function():
  print(“I’m a function")
```

Calling a Function

- To call a function, use the function name followed by parenthesis:
- Example

```
def my_function():
  print("Hello from a function")

my_function() // calling function
```

```
Output: Hello from a function
```

Arguments

- Information can be passed into functions as arguments.
- Example

```
def my_function(fname):
  print("Your first name is: " + fname)
my_function("Yousif")
```

Output: Your first name is: Yousif

```
def my_function(fname, lname):
  print("Your full name is: " + fname + " " + lname)
my_function("Yousif", "Yousif")
```

Output: Your first name is: Yousif Yousif

Arbitrary Arguments, *args

- If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.
- This way the function will receive a *tuple* of arguments, and can access the items accordingly:

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

```
Output: The youngest child is Linus
```

Arbitrary Keyword Arguments, **kwargs

This way the function will receive a *dictionary* of arguments, and can access the items accordingly:

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

```
Output: His last name is Refsnes
```

Default Parameter Value

- If we call the function without argument, it uses the default value:

```
def my_function(country = “Belgium"):
  print("I am from " + country)


my_function() # I am from Belgium
my_function("Sweden") #I am from Sweden
```