Code Structures in Python

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

Objectives

- To understand coding constructs like conditional statements, loops and iterations.
- To understand Comprehensions using Sequential Data Types.
- To Learn and implement Functions and Generators.
- To understand exceptions and ways to solve them.

If.. else Constructs

Python, like most other programming languages implements conditional statements using if..else constructs.
In general the if..else statement is used to check if a condition is True or False and based on the decision, decide whether or not to go ahead with the subsequent instruction. For example,

```
interesting = True
if interesting:
        print('Learn␣more␣about␣it..!')
else:
        print('Move␣on..')
```

Checking Multiple Conditions

More than one condition could be checked as follows:

```
interesting = True
easy = True
if interesting and easy:
        print('Learn more about it..!')
else:
        print('Move on..')
```

Nested Conditions

Conditional statements could be nested as follows:

```
interesting = True
easy = True
if interesting:
        if easy:
                print('Easy␣for␣you..!')
        else:
                print('Keep␣practicing..!')
else:
        print('Move␣on..')
```

Multiple Conditional Possibilities

In the event that there are more than 2 possibilities, we use a special construct called the elif which is just the shortened version of else..if.

```
time = 'morning'
if time == 'morning':
    print('Its time for Breakfast!!')
elif time == 'noon':
    print('Its time for lunch')
elif time == 'night':
    print('Its time for lunch')
else:
    print('I am hungry all the time anyways :(')
```

Coding Exercise

Try and answer why only the first if loop is getting executed and not the second one.

```
a = [1,2,3]
b = [1,2,3]

if a==b:
    print('They are equal')
if a is b:
    print('I said they are equal!!')
```

Coding Exercise

Now that you have mastered the art of variable comparison in Python, answer this.

```
a = 256
b = 256
a == b
True
a is b
True

a = 257
b = 257
a == b
True
a is b
False
```

Possible False Conditions

These are some of the possible conditions that could result in False

```
boolean False
null None
zero integer 0
zero float 0.0
empty string ''
empty list []
empty tuple ()
empty dict {}
empty set set()
```

Examples

```
# Example 1
a = 0
if a:
        print('Its␣␣True')
else:
        print('Its␣False')

# Example 2
def add(x,y):
    if type(x) is int and type(y) is int:
        return(x+y)

if sum:
    print('The␣sum␣is', sum)
else:
    print('I␣dont␣see␣no␣sum')
```

Repeat with While

While is the simplest looping construct that helps us to repeat a step any number of times as long as the condition evaluates to True.

```python
count = 1
while count < 5:
    print('The count is', count)
    count += 1
```

Using break can help us break the loop midway.

```python
count = 1
while count < 5:
    print('The count is', count)
    count += 1
    if count == 3:
        break
```

Use of Continue

Continue can be used to continue with the loop.

```
while True:
    value = input(”Enter␣an␣integer␣of␣choice.␣Pres
    if value == ’q’: # quit
        break
    else:
        number = int(value)
        if number%2 == 0:
            print(number, ”squared␣is”,\
             number*number)
            continue
        else:
            break
```

Summary

- We understood looping and conditional structures in Python.
- We learned the need for iteration and how it makes programming of repeated tasks easier.
- We understand looping structures such as if..else, while and for and other keywords such as break and continue.
- We also learned nesting and other complex code structures.