Brush up Python

December 20, 2018

0.1 1. Basics

```
In [1]: # printing any value
        print("Hello")
```

```
Hello
```

0.1.1 Variable

A variable is a reserved memory location to store values. A varaiable name must begin alphabet/underscore followed by alphabets/numbers/underscores

```
In [2]: # python data types
        number = 3
        float_number = 99.99
        character = 'a'
        string = "Hello, World!"

        print(number, type(number))
        print(float_number, type(float_number))
        print(character, type(character))
        print(string, type(string))
```

```
3 <class 'int'>
99.99 <class 'float'>
a <class 'str'>
Hello, World! <class 'str'>
```

```
In [3]: # writing basic operations
        a = 6
        b = 2
        print(a+b)
        print(a-b)
        print(a*b)
        print(a/b)
        print(a%b)
```

```
8
4
12
3.0
0
```

```
In [4]: # getting input
        number = input("Enter a number: ")

        print(number, type(number))
```

```
Enter a number: 3
3 <class 'str'>
```

```
In [5]: # casting string to integer
        number = int(input("Enter a number: "))

        print(number, type(number))
        print("Square of the number:", number**2)
```

```
Enter a number: 3
3 <class 'int'>
Square of the number: 9
```

0.2 3. Python Built-in Data Structures

0.2.1 3.1. List

- List is an ordered collection of data.
- It is mutable and allow duplicates.
- It list is created by enclosing the data items with in square brackets delimited by ","

```
In [6]: # Creating a list
        marks = [35, 68, 96, 80, 90]

        print(marks)
```

```
[35, 68, 96, 80, 90]
```

```
In [7]: # Indexing list elements
        print("marks[0] : %d" % marks[0])
        print("marks[len(marks)-1] : %d" % marks[len(marks)-1])
        print("marks[-1] : %d" % marks[-1])
```

```
marks[0] : 35
marks[len(marks)-1] : 90
marks[-1] : 90
```

```
In [8]: # Modifying list elemets
        marks[0] = 38

        print(marks)
```

```
[38, 68, 96, 80, 90]
```

```
In [9]: # Slicing the list
        # Syntax: list[start:end(exclusive)]
        print(marks[0:len(marks)])
        print(marks[:len(marks)])
        print(marks[0:])
        print(marks[:])

        print(marks[2:4])
```

```
[38, 68, 96, 80, 90]
[38, 68, 96, 80, 90]
[38, 68, 96, 80, 90]
[38, 68, 96, 80, 90]
[96, 80]
```

```
In [10]: # Adding and removing list elements
         marks.append(89)
         print(marks)

         marks.insert(1, 70)
         print(marks)

         marks.remove(70)
         print(marks)
```

```
[38, 68, 96, 80, 90, 89]
[38, 70, 68, 96, 80, 90, 89]
[38, 68, 96, 80, 90, 89]
```

```
In [11]: # sorting list elements
         print(sorted(marks))
         print(sorted(marks, reverse=True))
```

```
[38, 68, 80, 89, 90, 96]
[96, 90, 89, 80, 68, 38]
```

```
In [12]: # List Comprehesion
         pass_mark = 50
         marks_new = [mark for mark in marks if mark > pass_mark]
         print(marks_new)
```

```
[68, 96, 80, 90, 89]
```

0.2.2 3.2. Tuple

- Tuple is an ordered collection of data.
- It is immutable and allow duplicates.
- It list is created by enclosing the data items with in round brackets delimited by ","

```
In [13]: # Creating a tuple
         marks = (35, 68, 96, 80, 90)

         print(marks)
```

```
(35, 68, 96, 80, 90)
```

```
In [14]: # Indexing tuple elements
         print("marks[0] : %d" % marks[0])
         print("marks[len(marks)-1] : %d" % marks[len(marks)-1])
         print("marks[-1] : %d" % marks[-1])
```

```
marks[0] : 35
marks[len(marks)-1] : 90
marks[-1] : 90
```

```
In [15]: #  Note: We cannot modify the elemnts in a tuple

         #marks[0] = 38

         print(marks)
```

```
(35, 68, 96, 80, 90)
```

```
In [16]: # Slicing the tuple
         # Syntax: tuple[start:end(exclusive)]
         print(marks[0:len(marks)])
         print(marks[:len(marks)])
         print(marks[0:])
         print(marks[:])

         print(marks[2:4])
```

```
(35, 68, 96, 80, 90)
(35, 68, 96, 80, 90)
(35, 68, 96, 80, 90)
(35, 68, 96, 80, 90)
(96, 80)
```

```
In [17]: # tuple Comprehesion
         pass_mark = 50
         marks_new = tuple((mark for mark in marks if mark > pass_mark))
         print(marks_new)
```

```
(68, 96, 80, 90)
```

0.2.3 3.3. Set

- Set is an unordered collection of data.
- It is mutable and does not allow duplicates.
- It is created by enclosing the data items with in curly brackets delimited by ","

```
In [18]: cities = {"Madras", "Delhi", "Bombay", "Calcutta", "Madras"}

         print(cities)
```

```
{'Bombay', 'Delhi', 'Madras', 'Calcutta'}
```

```
In [19]: # Set Operations

         a = {1, 2, 3, 4}
         b = {4, 5, 6}

         # Set Union
         print(a | b)

         # Set Intersection
         print(a & b)

         # Set Difference
         print(a - b)

         # Symmetric Difference
         print(a ^ b)
```

```
{1, 2, 3, 4, 5, 6}
{4}
{1, 2, 3}
{1, 2, 3, 5, 6}
```

```
In [20]: # Adding and removing set elements
         a.add(7)
         print(a)

         a.remove(7)
         print(a)
```

```
{1, 2, 3, 4, 7}
{1, 2, 3, 4}
```

0.2.4 3.4. Dictionary

- Dictionary is an unordered collection of key-values pairs.
- It is mutable.
- It is created by enclosing the key-value parirs (key:value) with in curly brackets delimited by ","

```
In [21]: student = {"name":"Gokul", "reg_no":"15IT026", "course":"BTech"}

         print(student)
```

```
{'name': 'Gokul', 'reg_no': '15IT026', 'course': 'BTech'}
```

```
In [22]: # Accessing elements
         print(student["course"])

         for key, value in student.items():
             print("Key: %s ; Value: %s" % (key, value))
```

```
BTech
Key: name ; Value: Gokul
Key: reg_no ; Value: 15IT026
Key: course ; Value: BTech
```

0.2.5 3.5. String

- A string is a sequence of charactes
- It is immutable.
- It is created by enclosing text with single/double/triple quotes

```
In [23]: word = "Malayalam"
         print(word)
```

```
Malayalam
```

```
In [24]: # String operations

         print(word.lower())
         print(word.upper())
         print(len(word))
         print(word.startswith("A"))
         print(word.endswith("m"))
```

```
malayalam
MALAYALAM
9
False
True
```

```
In [25]: # Slicing the string
         print(word[0:5])
```

```
Malay
```

```
In [26]: # Coverting string to a list of characters
         print(list(word))
```

```
['M', 'a', 'l', 'a', 'y', 'a', 'l', 'a', 'm']
```

```
In [27]: # splitting the text
         sentence = "I love deep learning"
         words = sentence.split(" ")
         print(words)
```

```
['I', 'love', 'deep', 'learning']
```

```
In [28]: # finding length of a list
         print(len(words))
```

```
4
```

0.3 4. Control Structures

'if...else' syntax:

```
if CONDITION:
    code block 1
elif CONDITION:
    code block 2
...
else:
    code block n
```

```
In [29]: a = 3
         b = 5

         if a > b:
             print("a is greater than b")
```

```
In [30]: if a > b:
             print("a is greater than b")
         else:
             print("b is greater than a")
```

```
b is greater than a
```

```
In [31]: score = 80
         grade = None

         if score > 90:
             grade = "A"
         elif score > 80:
             grade = "B"
         elif score > 70:
             grade = "C"
         elif score > 60:
             grade = "D"
         elif score > 50:
             grade = "E"
         else:
             grade = "F"

         print("Grade: " + grade)
```

```
Grade: C
```

'for' syntax:

```
for ELEMENT in SEQUENCE:
    code block
```

```
In [32]: ofl_team = ["mitesh", "pratyush", "ananya", "rohith", "prem", "gokul"]
         for person in ofl_team:
             mail = person + "@onefourthlabs.com"
             print(mail)
```

```
mitesh@onefourthlabs.com
pratyush@onefourthlabs.com
ananya@onefourthlabs.com
rohith@onefourthlabs.com
prem@onefourthlabs.com
gokul@onefourthlabs.com
```

```
In [33]: for i in range(len(ofl_team)):
             mail = ofl_team[i] + "@onefourthlabs.com"
             print(mail)
```

```
mitesh@onefourthlabs.com
pratyush@onefourthlabs.com
ananya@onefourthlabs.com
rohith@onefourthlabs.com
prem@onefourthlabs.com
gokul@onefourthlabs.com
```

'while' syntax:

```
while CONDITION:
    code block
```

```
In [34]: n = int(input("Enter a number: "))

         n_factorial = 1
         x = 2
         while x <= n:
             n_factorial = n_factorial * x
             x = x+1

         print("%d! = %d" % (n, n_factorial))
```

```
Enter a number: 5
5! = 120
```

0.4 5. Functions

- A function is a block of code that can be reused.
- It is defined using the 'def' keyword in python.

Function syntax:

```
def function_name(parameters):
    code...
    return value
```

```
In [35]: # Defining the function
         def area_of_square(length, breadth):
             area = length*breadth
             return area

In [36]: # Calling the function
         area_of_square(2, 3)

Out[36]: 6
```

In [37]:
```
# Function with default arguments
def greet(name=""):
    print("Hello " + name)

greet("Harish")
greet()
```

```
Hello Harish
Hello
```