Python Basics

Somenath Sinha

December 2017

Contents

Chapter 1

Basics

1.1 Syntax

The lines in python don't end with semicolon. Thus, the end of the lines matter, and the spaces matter. The

1.2 Printing

Printing is done with `print()`. Each print automatically prints a newline at end, unless the end character is specified. The escape sequences are respected as usual.

```
1        print("Hello")
2        print("Hello")
3        print("Hello", end="")
4        print("Hello")
5        print("Hello", end="! ")
6        print("Hello")
7        print("Hel\nlo")
8        print("Hello")
9        print("He\tllo")
10       print("Hello")
```

Output

```
Hello
Hello
HelloHello
Hello! Hello
Hel
lo
Hello
He      llo
Hello
```

1.2.1 Printing multiple words

The procedure to print multiple words using the same print is:

```
print("Hello"+"World")
```

Output

```
HelloWorld
```

1.2.2 Comments

```
#   Single line comment

'''
Block comment!
'''
```

1.2.3 Variables

Python doesn't need a specific datatype declaration. So, we can directly assign a value to a variable. In python, both single and double quotes represent a string.

```
x = 4.5
y = 'a word'
z = "a new string"
```

1.2.4 Printing Value of a variable

When printing a variable, python automatically prints a space every time a variable's value is printed.

```
x = 4
print("x =",x)
y = "Space"
print("Without", y, sep="")
```

Output

```
x = 4
WithoutSpace
```

1.3 Arithmetic Operations

1.3.1 Basic Arithmetic

In Python +, -, * and % (modulus) all act as in Java. Division however acts different.

```
a=5
b=4
x=a+b
y=a-b
z=a*b
w=a%b

print("a+b =",x)
print("a-b =",y)
print("a*b =",z)
print("a%b =",w)
```

Output

```
a+b = 9
a-b = 1
a*b = 20
a%b = 1
```

Division

In case of java, the division is called integer division where integer truncation occurs with the result. In python, a value with a decimal point will be returned. To bypass this, we use the // (floor division) operator.

```
a=5
b=4
x=a/b
y=a//b


print("a/b =",x)
print("a//b =",y)
```

Output

```
a/b = 1.25
a//b = 1
```

Exponents

The exponent operator is **.

```
x=2**3

print("2^3 =",x)
```

Output

```
2^3 = 8
```

1.3.2 Casting

```
x=int(3.5)

print("x =",x)
```

Output

```
x = 3
```

1.3.3 Library Math functions

```
x=max(3,4,5)
y=min(3,4,5)

print("Max =",x)
print("Min =",y)
```

Output

```
Max = 5
Min = 3
```

1.4 User Input

```
print("Enter a value for x: ")
x=input();
y=input("Test value for y: ")
print("x =",x)
print("y =",y)
```

Output

```
Enter a value for x:
5
Test value for y: 6
x = 5
y = 6
```

1.4.1 Casting user input

If the input needs to be casted, it should be done so immediately, after the input.

```
x=float(input("Enter a num: "))
y=int(x)
print("x =", x)
print("y =", y)
```

Output

```
Enter a num: 3.5
x = 3.5
y = 3
```

1.5 String functions

Just like in Java, strings are immutable in Python, and thus each string function returns a new string.

1.5.1 Printing String length

```
1 s="input"
2 print("Length of s =",len(s))
```

Output

```
Length of s = 5
```

1.5.2 Substring

```
1 s="input"
2 print("Last 3 characters: ",s[2:])        # Called slice notation
3 print("2rd and 4th characters: ",s[2:4])
4 print("Last 3 characters: ", s[-2:])        # Negative index indicates count from
  ↪ the last.
```

Output

```
Last 3 characters:  put
2rd and 4th characters:  pu
Last 3 characters:  ut
```

1.5.3 In operator

```
1 s="input"
2 print("Contains pu: ", "pu" in s)   # Returns true if the string is present in s.
```

Output

```
Contains pu:  True
```

Chapter 2

Datastructures

2.1 Lists

```
1         list = []   # Creates an empty list
2         list.append("House")
3         list.append("Mouse")
4         list.append("Blouse")
5         print(list)
6         print("Size :", len(list))
7         print("Index 1:", list[1])
8         list.insert(1,"Grouse")
9         print("Index 1:", list[1], "\nEntire list:", list)
10        del(list[1:2])    # Delete the item at index 1 & 2 of list
11        print(list)
```

Output

```
['House', 'Mouse', 'Blouse']
Size : 3
Index 1: Mouse
Index 1: Grouse
Entire list: ['House', 'Grouse', 'Mouse', 'Blouse']
['House', 'Blouse']
```

2.1.1 Extend

```
1         list1 = []   # Creates an empty list1
2         list1.append("House")
3         list1.append("Mouse")
4         list1.append("Blouse")
5
6         list2 = []   # Creates an empty list2
7         list2.append("House")
8         list2.append("Mouse")
9         list2.append("Blouse")
10
11        list3 = list1 + list2
12        list1.extend(list2)
13
14        print(list1)
```

```
        print(list3)

        list4 = []  # Will be a list of lists!
        list4.append(list1)
        list4.append(list2)
        print(list4)
        print(list4[0][4])
```

Output

```
        ['House', 'Mouse', 'Blouse', 'House', 'Mouse', 'Blouse']
        ['House', 'Mouse', 'Blouse', 'House', 'Mouse', 'Blouse']
        [['House', 'Mouse', 'Blouse', 'House', 'Mouse', 'Blouse'], ['House', 'Mouse',
↪  'Blouse']]
        Mouse
```

2.1.2 Immutable Tuples

A `tuple` is an immutable list. Once created, it cannot be changed, although new tuples can be created from it.

```
        x= 2,3,4,5  # x is a tuple.
        print(x)
```

Output

```
        (2, 3, 4, 5)
```

Tuple of tuples

```
        x= 2,3,4,5  # x is a tuple.
        print(x)
        y= x,6,7
        print(y)
```

Output

```
        (2, 3, 4, 5)
        ((2, 3, 4, 5), 6, 7)
```

Tuple of only 1 element

```
        x=5,
        print(x)
```

Output

```
        (5,)
```

2.1.3 Tuple functions

```
x=5,4,3,2,1
print(x)
print(len(x))
print(3 in x)
```

Output

```
(5, 4, 3, 2, 1)
5
True
```