Basics of Python

July 19, 2022

1 Basics of Python Programming Language

```
[3]: 10
```

```
[3]: 10
```

```
[4]: -100
```

```
[4]: -100
```

```
[5]: 12.2
```

```
[5]: 12.2
```

```
[6]: -146.55
```

```
[6]: -146.55
```

```
[7]: pavan
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [7], in <cell line: 1>()
----> 1 pavan

NameError: name 'pavan' is not defined
```

```
[8]: 'pavan'
```

```
[8]: 'pavan'
```

```
[9]: "pavan"
```

```
[9]: 'pavan'
```

1.1 Data types

- int

- float
- bool
- str
- range
- list
- tuple
- set
- dictionary

1.1.1 int

- postive, negative, zero

[2]:
```python
a = 10
print(a)
print(type(a))
```

```
10
<class 'int'>
```

[3]:
```python
b = -146
print(b)
print(type(b))
```

```
-146
<class 'int'>
```

1.1.2 float

- real number

[4]:
```python
a = 10.2
print(a)
print(type(a))
```

```
10.2
<class 'float'>
```

[5]:
```python
b = -12.67
print(b)
print(type(b))
```

```
-12.67
<class 'float'>
```

1.1.3 bool

- True or False

[6]:
```python
a = 10
b = 20
```

```
print(a > b)
```

```
False
```

[7]:
```
x = 10
y = 20
print(y > x)
```

```
True
```

1.1.4 str

- Everything put within two single quotes two double quotes or two triple quotes is a string in python

[8]:
```
name = 'pavan'
print(name)
print(type(name))
```

```
pavan
<class 'str'>
```

[9]:
```
a = '10'
print(a)
print(type(a))
```

```
10
<class 'str'>
```

[10]:
```
a = 10
b = 20
print(a + b)
```

```
30
```

[12]:
```
a = '10'
b = '20'
c = '30'
print(a + b + c)
```

```
102030
```

1.1.5 list

- Ordered collection of elements
- Homogeneous list, heterogeneous
- Homogeneous list: [10, 20, 30]
- Heterogeneous list: [10, 12.2, 'hello', True]

[14]:
```
lst = [10, 20, 30, 40, 50]
      # 0   1   2   3   4 --> indexes
```

```
print(lst)
print(type(lst))
```

```
[10, 20, 30, 40, 50]
<class 'list'>
```

[17]:
```
# Accessing list elements using indexes
lst = [10, 20, 30, 40, 50]
      # 0   1   2   3   4 --> indexes
print(lst[0]*2 + lst[3]*4 - lst[4]*2)
```

```
80
```

1.1.6 tuple

- (10, 20, 30)
- (10, 14.5, 'hello', False)

[18]:
```
t = (10, 20, 30, 40)
print(t)
print(type(t))
```

```
(10, 20, 30, 40)
<class 'tuple'>
```

[19]:
```
t = (10, 20, 30, 40)
print(t[2])
```

```
30
```

1.1.7 set

- Unordered collection of unique elements

[20]:
```
lst = [10, 20, 30, 40, 10]
print(lst)
```

```
[10, 20, 30, 40, 10]
```

[21]:
```
s = {10, 20, 30, 40, 10}
print(s)
```

```
{40, 10, 20, 30}
```

[22]:
```
fruits = {'apple', 'kiwi', 'kiwi', 'apple'}
print(fruits)
```

```
{'apple', 'kiwi'}
```

[23]:
```
a = {10, 20, 30, 40}
b = {10, 20, 50, 60}
print(a.intersection(b))
```

```
{10, 20}
```

1.1.8 Dictionaries

- word: definition
- hero: no.of films
- hero: industry hit
- name: rollnumber
- roll: list of marks

[24]:
```
d = {'pk':'gabbar singh', 'mb':'pokiri', 'nbk':'legend', 'chiru':'khaidi',
     'prabhas': 'mirchi', 'aa':'aarya', 'sampu':'kobbarimatta'}
print(d)
print(type(d))
```

```
{'pk': 'gabbar singh', 'mb': 'pokiri', 'nbk': 'legend', 'chiru': 'khaidi',
'prabhas': 'mirchi', 'aa': 'aarya', 'sampu': 'kobbarimatta'}
<class 'dict'>
```

[27]:
```
d = {'pk':'gabbar singh', 'mb':'pokiri', 'nbk':'legend', 'chiru':'khaidi',
     'prabhas': 'mirchi', 'aa':'aarya', 'sampu':'kobbarimatta'}
print(d['pk'])
print(d['sampu'])
print(d['prabhas'])
```

```
gabbar singh
kobbarimatta
mirchi
```

[ ]:
```
# Pangram
'a quick brown fox jumps over the lazy dog'
{'a': 2, 'b': 1, 'c'}
```

1.2 Varibles

- Variable is container of data

1.2.1 Rules to create variable names

1. Variable name can be alphanumeric, but it should not start with a digit
2. Variable name should not contain any white spaces within.
3. No other special character is allowed other than underscore (_)
4. Underscore itself can be used as a variable.
5. You cannot use keywords as variable names.
6. Variable name is case sensitive (NUM, Num, num)

[ ]:
```
```

[ ]:
```
```

[ ]:
```

```

Variable name can be alphanumeric, but it should not start with a digit

[28]:
```
# valid
person1 = 'Dave'
print(person1)
```

```
Dave
```

[29]:
```
# invalid
1stperson = 'Dave'
print(1stperson)
```

```
  Input In [29]
    1stperson = 'Dave'
    ^
SyntaxError: invalid decimal literal
```

Variable name should not contain any white spaces within.

[30]:
```
# invalid
first name = 'captain'
last name = 'america'
```

```
  Input In [30]
    first name = 'captain'
          ^
SyntaxError: invalid syntax
```

[33]:
```
# valid
first_name = 'captain'
last_name = 'america'
print(first_name + " " + last_name)
```

```
captain america
```

No other special character is allowed other than underscore (_)

[34]:
```
person_1 = 'thor'
print(person_1)
```

```
thor
```

[35]:
```
person#1 = 'thor'
print(person#1)
```

```
  Input In [35]
    print(person#1)
                   ^
SyntaxError: incomplete input
```

Underscore itself can be used as a variable.

```
[37]: _ = 10
      print(_ * _)
```

```
100
```

```
[38]: in = 10
      print(in)
```

```
  Input In [38]
    in = 10
    ^
SyntaxError: invalid syntax
```

```
[39]: in1 = 10
      print(in1)
```

```
10
```

Variable name is case sensitive (NUM, Num, num)

```
[40]: NUM = 10
      num = 20
      Num = 30
      print(num)
```

```
20
```

Naming convensions

```
[ ]: num1 = 10
     num2 = 20
     num3 = num1 + num2
```

```
[41]: # use these
      # emp_name
      # emp_id
      # emp_sal
      # emp_loc
      # instead of these
      # a
      # b
      # c
      # d
```

1.3 print() function

- Used to print either single value of multple values

[42]:
```
a = 10
print(a)
```

```
10
```

[43]:
```
name = 'wanda'
print(name)
```

```
wanda
```

[45]:
```
a = 10
b = 20
c = 30
print(a,b,c)
```

```
10 20 30
```

[46]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(…)
    print(value, …, sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

[47]:
```
print(10, 20, 30, sep=',')
```

```
10,20,30
```

[48]:
```
print(10, 20, 30, sep='sdfj')
```

```
10sdfj20sdfj30
```

[49]:
```
print(10)
print(20)
```

```
10
20
```

[53]:
```
# \n --> newline character
print('he\nllo\nworld')
```

```
he
llo
world
```

[55]:
```
print('p\na\nv\na\nn')
```

```
p
a
v
a
n
```

[56]:
```
print(10, end=' ')
print(20)
```

```
10 20
```

[57]:
```
print(10)
print(20, end=' ')
print(30)
print(40, end='#')
print(50)
print(60)
```

```
10
20 30
40#50
60
```

[59]:
```
for i in range(5):
    print(i, end = ' ')
```

```
0 1 2 3 4
```

[63]:
```
print(10, 20, 30, 40, sep='\n')
```

```
10
20
30
40
```

1.4 I/O statements

1.4.1 input() function

- Used to take inputs from user (keyboard)
- But it will take those inputs in the form of a string

type conversion
int(input()) –> To read integers
float(input()) –> To read floating point values

[70]:
```
a = input()
```

```
10
```

[71]:
```
print(type(a))
```

```
<class 'str'>
```

[75]:
```
a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
print(a + b)
```

```
Enter a value: 10
Enter b value: 20
30
```

[73]:
```
help(input)
```

```
Help on method raw_input in module ipykernel.kernelbase:

raw_input(prompt='') method of ipykernel.ipkernel.IPythonKernel instance
    Forward raw_input to frontends

    Raises
    ------
    StdinNotImplementedError if active frontend doesn't support stdin.
```

[80]:
```
a = int(input("Enter a value: "))
b = int(input("Enter b value: "))
print(a + b)
```

```
Enter a value: 10
Enter b value: 20
30
```

[77]:
```
x = float(input("Enter x value: "))
y = float(input("Enter y value: "))
print(x + y)
```

```
Enter x value: 10.5
Enter y value: 0.5
11.0
```

1.5 What to look for when solving any problem

Inputs?

Process

Output?

1.6 Program to print area and perimeter of square

[79]:
```
side = int(input("Enter side length: "))
area = side * side
peri = 4 * side
print(area)
print(peri)
```

```
Enter side length: 5
25
20
```