

Python ¶

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990.
- Like Perl, Python source code is also available under the GNU General Public License (GPL).

Variable types in Python ¶

Below are different types of variables in Python.

```
hl-ipython3
```

- int
- float
- string
- list
- dictionary
- tuple

Integer Declaration (int) ¶

In [3]:

```
#Integer values can be assigned directly using '='  
  
age = 10
```

In [4]:

```
print(age) #Print function is used to print the value of a variable.
```

```
10
```

In [5]:

```
type(age) #We can find out the datatype of a variable using type() function
```

Out[5]:

```
int
```

In [6]:

```
age #We use Jupyter-notebook to write this tutorial, here just entering a variable will print it's value.
```

Out[6]:

```
10
```

Float Declaration (float) ¶

In [8]:

```
#Any value which contains a decimal point is considered as  
a float variable.
```

```
price = 10.64
```

In [9]:

```
price
```

Out[9]:

```
10.64
```

In [10]:

```
type(price)
```

Out[10]:

```
float
```

In [11]:

```
#Float cannot recognize any variables with more than one '
.'
```

```
ip = 10.0.0.1
```

<pre>

 File "<ipython-input-11-3bf74e90f524>"

, line 3

 ip = 10.0.0.1

^

SyntaxError: invalid syntax

</pre>

String Declaration (str) ¶

In [14]:

```
#A string should be declared inside a ' ' or ""
```

```
name = 'syam'
```

In [15]:

```
type(name)
```

Out[15]:

```
str
```

In [16]:

```
#Any numbers or alphabets can be saved as a string.
```

```
ip = '10.0.0.1'
```

In [17]:

```
type(ip)
```

Out[17]:

```
str
```

In [18]:

```
#Even numbers enclosed in ' ' or " " will be considered as a  
string.
```

```
age = '31'
```

In [19]:

```
type(age)
```

Out[19]:

```
str
```

Basic Mathematical Operations using Python 🐍

In [20]:

```
x = 10  
y = 3
```

In [21]:

```
#Addition  
  
x + y
```

Out[21]:

```
13
```

In [22]:

```
#Subtraction
```

```
x - y
```

Out[22]:

```
7
```

In [23]:

```
#Multiplication
```

```
x * y
```

Out[23]:

```
30
```

In [24]:

```
#Division
```

```
x / y
```

Out[24]:

```
3.3333333333333335
```

In [25]:

```
#Floor Division - Here the values of variables will be divided but returns only a whole number value.
```

```
x // y
```

Out[25]:

```
3
```

In [26]:

```
#Modulus - Divides and returns the remainder.
```

```
x % y
```

Out[26]:

```
1
```


All the above operations can only be done on an integer.

In [29]:

```
price = '10' #Here 10 is a string  
quantity = 5 #Here 5 is an integer
```

In [30]:

```
#When a string is multiplied by int, below happens.  
price * quantity
```

Out[30]:

```
'1010101010'
```

In [32]:

```
#Another example  
  
'a' * 20
```

Out[32]:

```
'aaaaaaaaaaaaaaaaaaaaa'
```

Converting a string into Integer ¶

In [33]:

```
x = '10' #A string is declared
```

In [34]:

```
y = int(x) #int() will convert string to integer. Here value of y will be integer 10 and value of x will remain str
```

In [35]:

```
y
```

Out[35]:

```
10
```

In [36]:

```
type(y)
```

Out[36]:

```
int
```

In [37]:

```
x * 5 #Since x is still str, it return below value
```

Out[37]:

```
'1010101010'
```

In [38]:

```
int(x) * 5 #x is converted to int and then multiplied with  
5
```

Out[38]:

```
50
```

Length function ¶

In [40]:

```
#Len() is used to find the length of a string.
```

```
lang = 'malayalam'
```

```
len(lang)
```

Out[40]:

```
9
```

In [41]:

```
#We cannot calculate the length of an integer
```

```
atm_pin = 1234
```

```
len(atm_pin)
```

```
<pre>
```

```
<span>-----</span>
```

```
<span>TypeError</span> Traceback (most recent call last)
```

```
<span><ipython-input-41-2edc3670c39b></span> in <span>
```

```
<module></span>
```

```
<span> 1</span> atm_pin <span>=</span> <span>1234</span>
```

```
<span>----> 2</span>len<span>(</span>atm_pin<span>)</span>
```

```
<span>TypeError</span>: object of type 'int' has no len()</pre>
```

Converting to string ¶

In [43]:

```
str(atm_pin) #str() can be used to convert integer to string.
```

Out[43]:

```
'1234'
```

In [44]:

```
len(str(atm_pin)) #So we can now calculate the length of an integer
```

Out[44]:

```
4
```

Comparison Operators ¶

In [46]:

```
10 > 5 #Here the output will be boolean (True or False)
```

Out[46]:

```
True
```

In [47]:

```
5 < 5
```

Out[47]:

```
False
```

In [48]:

```
5 < 5 or 5 == 5 #if any one of o/p is True, or returns True.  
== is used to compare if values are equal.
```

Out[48]:

```
True
```

In [49]:

```
1 <= 1
```

Out[49]:

```
True
```

String Membership Operation ¶

In [52]:

```
fruit = 'pineapple'
```

In [53]:

```
'apple' in fruit #This will return True because while traversing through pineapple and check for the string apple in it.
```

Out[53]:

```
True
```

In [54]:

```
'app' not in fruit #This statement is False because string app is present in pineapple
```

Out[54]:

False

In [55]:

```
'syam' not in fruit #This statement is True because string  
syam is not present in pineapple
```

Out[55]:

True

Input() Function ¶

In [56]:

```
#This function is used to read values from keyboard input.
```

```
input()
```

Syam

Out[56]:

'Syam'

In [57]:

```
name = input("Please enter your name: ") #This will print  
the message and wait for your keyboard input.
```

```
Please enter your name: Syam
```

In [58]:

```
pin = input("Please neter your ATM PIN: ")
```

```
Please neter your ATM PIN: 1234
```

In [59]:

```
type(pin) #All input read via input() will be considered a  
s a string.
```

Out[59]:

```
str
```

If condition ¶

In [60]:

```
#If condition is used to check if a statement is true or false and execute certain commands respectively.
```

```
original = 4556
```

```
pin = input("Please Enter your ATM PIN : ")
```

```
if int(pin) == original:                                #Checking if entered pin is equal to value set to variable 'original'
```

```
    print("Success")                                    #If True it will execute this
```

```
    print("Dummy Message")                              #And this and any other line coming with same indentation
```

```
else:                                                    #If entered pin is not equal, it will execute the commands below this line
```

```
    print("Failed")
```

```
Please Enter your ATM PIN : 1234
```

```
Failed
```

String Methods ¶

In [61]:

```
lang = 'Malayalam'
```

In [62]:

```
id(lang) #Every string will be assigned a unique id which  
will be modified when the value of variable changes,
```

Out[62]:

```
139638595136688
```

In [63]:

```
lang = 'English'  
id(lang)
```

Out[63]:

```
139638595162720
```

dir()

In [64]:

```
#dir() any value will print whatever default modules can b  
e used to it.
```

```
dir(lang)
```

Out[64]:

```
['__add__',  
 '__class__',  
 '__contains__',  
 '__delattr__',  
 '__dir__',  
 '__doc__',  
 '__eq__',  
 '__format__',  
 '__ge__',  
 '__getattr__',  
 '__getitem__',  
 '__getnewargs__',  
 '__gt__',  
 '__hash__',  
 '__init__',  
 '__init_subclass__',  
 '__iter__',  
 '__le__',  
 '__len__',  
 '__lt__',  
 '__mod__',  
 '__mul__',  
 '__ne__',  
 '__new__',
```

```
'__reduce__',  
'__reduce_ex__',  
'__repr__',  
'__rmod__',  
'__rmul__',  
'__setattr__',  
'__sizeof__',  
'__str__',  
'__subclasshook__',  
'capitalize',  
'casefold',  
'center',  
'count',  
'encode',  
'endswith',  
'expandtabs',  
'find',  
'format',  
'format_map',  
'index',  
'isalnum',  
'isalpha',  
'isdecimal',  
'isdigit',  
'isidentifier',  
'islower',  
'isnumeric',  
'isprintable',
```

```
'isspace',  
'istitle',  
'isupper',  
'join',  
'ljust',  
'lower',  
'lstrip',  
'maketrans',  
'partition',  
'replace',  
'rfind',  
'rindex',  
'rjust',  
'rpartition',  
'rsplit',  
'rstrip',  
'split',  
'splitlines',  
'startswith',  
'strip',  
'swapcase',  
'title',  
'translate',  
'upper',  
'zfill']
```

We can use any of the above modules to modify the output of value

lang.

Upper() ¶

In [65]:

```
lang.upper() #This will traverse through the string and change it's values to UPPER CASE.
```

Out[65]:

```
'ENGLISH'
```

lower() ¶

In [66]:

```
lang.lower() #This will traverse through the string and change it's value to LOWER CASE.
```

Out[66]:

```
'english'
```

count() ¶

In [67]:

```
lang = 'MaḷAyaḷAM'
```

In [68]:

```
lang.count('a') #This will count only 'a' in the above variable 'A' will not be considered.
```

Out[68]:

```
2
```

In [69]:

```
#So to check the correct number of 'a' we need to convert to upper or lower.  
lang.lower().count('a')
```

Out[69]:

```
4
```

replace() ¶

In [71]:

```
lang.replace('a','x') #This will replace all 'a' with 'x'  
(Note : 'A' will not be considered here also)
```

Out[71]:

```
'MxlAyxlAM'
```

In [76]:

```
str.replace? #Adding a ? after any inbuilt function will p  
rovide info about it.
```

endswith() ¶

In [77]:

```
#To check if a string ends with a specific value. It will  
return a boolean value.
```

```
conf = '/etc/httpd/conf/httpd.conf'  
conf.endswith('.conf')
```

Out[77]:

True

startswith() ¶

In [78]:

```
#To check if a string starts with a specific value. It will  
also return a boolean value.
```

```
conf.startswith('/etc')
```

Out[78]:

True

isdigit() ¶

In [79]:

```
#To check if a string is a digit or not.
```

```
number = '12345'
```

```
number.isdigit()
```

Out[79]:

True

In [80]:

```
#If it contains any alphabets, it will return False
```

```
number='123gcv'
```

```
number.isdigit()
```

Out[80]:

False

isalpha() ¶

In [81]:

```
#To check if a string contains only alphabets
```

```
name = 'syam'
```

```
name.isalpha()
```

Out[81]:

True

In [82]:

```
conf = '/etc/httpd/conf/httpd.conf'  
conf.isalpha()
```

Out[82]:

```
False
```

rstrip()

In [86]:

```
#To remove any values from the right side of a string.  
#If no arguments are given, white spaces will be removed.
```

```
name = '    syam    '  
name.rstrip()
```

Out[86]:

```
'    syam'
```

In [88]:

```
name.rstrip('am    ')
```

Out[88]:

```
'    sy'
```

lstrip() ¶

In [89]:

```
#To remove any values from left side of a string.  
#If no arguments are given, white spaces will be removed.  
  
name.lstrip()
```

Out[89]:

```
'syam    '
```

strip() ¶

In [90]:

```
#This will search for occurrences of a string and remove it  
from left and right end.  
#If no arguments are given, white spaces will be removed.  
name.strip()
```

Out[90]:

```
'syam'
```

In [92]:

```
name.strip(' s')
```

Out[92]:

```
'yam'
```

In [93]:

```
name = '-----syam--'  
name.strip('-')
```

Out[93]:

```
'syam'
```

In [94]:

```
name = '<color><h1>'  
name.strip('<').strip('>')
```

Out[94]:

```
'color<h1'
```

String Indexing ¶

In [95]:

```
#The characters in a string is indexed and can be accessed  
using an index number.
```

```
#By default index number starts with 0.
```

```
name = 'syam'
```

```
name[0]
```

Out[95]:

```
's'
```

In [96]:

```
#We can read the string from backwards as well using '-'
```

```
name[-1]
```

Out[96]:

```
'm'
```

In [97]:

```
name[-2]
```

Out[97]:

```
'a'
```

String Slicing ¶

In [98]:

```
timestamp = '12/Dec/2015:18:25:11'
```

In [100]:

```
#We can slice a string using indexes.
```

```
timestamp[0:11]          #Here it will print from index 0  
- 10.
```

Out[100]:


```
'12/Dec/2015'
```

In [101]:

```
#The same can be done like this also.
```

```
timestamp[:11]
```

Out[101]:

```
'12/Dec/2015'
```

In [103]:

```
#This can also be reversed.
```

```
timestamp[12:]          #This will print from the 12th index  
to the end of string.
```

Out[103]:

```
'18:25:11'
```

In []: