

```
#identifiers are case-sensitive, i.e : 'cat' & 'CAT' are different.
a = 20
A = 40
print(a)
print(A)
```

20 40

Variables: - Variables are the name given for memory allocations that store some values.

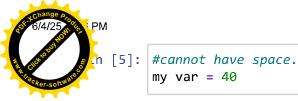
As python is Dynamically-typed language therefore we don't have to de clare a variable name with datatype.

Rules for variable naming convention :

- 1. keywords cannot be variables.
- 2. cannot start with a digit.
- 3. cannot have space.
- 4. cannot include any special characters except underscore '_'.
- 5. Are case-sensitive, i.e : 'a' & 'A' are different.

```
In [2]: #keywords cannot be variables.
        True = 10
          Cell In[2], line 2
            True = 10
        SyntaxError: cannot assign to True
In [3]: #cannot start with a digit.
        1ab = 30
          Cell In[3], line 2
            1ab = 30
        SyntaxError: invalid decimal literal
```

In [4]: ab1 = 23



```
The Change Processing the Control of the Change Processing the Control of the Change Processing the Change Pro
```

```
my var = 40

Cell In[5], line 2
  my var = 40

SyntaxError: invalid syntax
```

```
In [6]: myvar = 40
In [7]: #cannot include any special characters except underscore '_'.
@a = 34

Cell In[7], line 2
@a = 34

SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Datatypes :- . It represents the kind of value a variable have.

Name	Туре	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey":"value", "name":"Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

Out[10]: int

In [8]:

a = 456



```
#float
b = 4.6
type(b)
```

```
Out[11]: float
```

Out[12]: complex

```
In [13]: #sequential
    #string
    a = 'python'
    type(a)
```

Out[13]: str

Out[14]: list

```
In [15]: #tuple
    t = (4,5,6,7,9)
    type(t)
```

Out[15]: tuple

Out[16]: dict

```
In [17]: #set
s = {4,5,6,7,8,3,34,5}
type(s)
```

Out[17]: set

```
In [18]: #boolean
b = True
type(b)
```

Out[18]: bool



```
PM PythonBasic - Jupyter Notebook

In [ ]:

In [ ]:

In [ ]:
```

While representing a multiline string/sentence we should always use "" "" or """" """.

User input in python:

```
User input from stdin can be recieved via input() built-in function. Syntax:
```

input(message) >> message is a string

input() returns str datatype by default, to take numerical datatype a
s input(), convert it into required datatype explicitly.

```
In [20]: | a = input('Enter a number: ')
         print(a)
         type(a)
         Enter a number: 4
Out[20]: str
In [21]: | a = int(input('Enter a number: '))
         print(a)
         type(a)
         Enter a number: 5
Out[21]: int
In [22]: a = 67
         b = 78
         print(a+b)
         145
In [23]: 57+78
Out[23]: 135
```



```
a = input('Enter a number: ')
         b = input('Enter a number: ')
         c=a+b
         print(c)
         type(c)
         Enter a number: 5
         Enter a number: 6
         56
Out[26]: str
In [27]: | a = int(input('Enter a number: '))
         b = int(input('Enter a number: '))
         c=a+b
         print(c)
         type(c)
         Enter a number: 7
         Enter a number: 6
         13
Out[27]: int
 In [ ]:
 In [ ]:
 In [ ]:
```

Typecasting: It is a process of changing one's default datatype to another datatype explicitly.

typecasting is otherwise known as type conversion or datatype conversion.

```
In [28]: a = 'python'
In [30]: #string to List
v = list(a)
print(v)
type(v)

['p', 'y', 't', 'h', 'o', 'n']
Out[30]: list
```





```
[32]: #string to tuple
         t = tuple(a)
         print(t)
         type(t)
         ('p', 'y', 't', 'h', 'o', 'n')
Out[32]: tuple
In [33]: #string to set
         s = set(a)
         print(s)
         type(s)
         {'p', 'o', 'h', 'y', 't', 'n'}
Out[33]: set
In [34]: #string to dictionary
         d = dict(a)
         print(d)
         ValueError
                                                    Traceback (most recent call last)
         Cell In[34], line 2
               1 #string to dictionary
         ----> 2 d = dict(a)
               3 print(d)
         ValueError: dictionary update sequence element #0 has length 1; 2 is required
 In [ ]:
```

Output in python :-

User can see the output via print() built-in function.

```
In [35]: print('hello this is print function')
```

hello this is print function

String Datatype :- It is an immutable sequence of characters, where space is also a character.

Each character has a specific position number called index, and the i ndex is used to address the character.

The index value starts from 0.

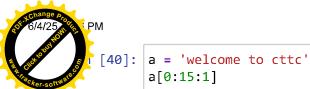
A string can be assigned using ' ', " ", ''' ''', or """ """.



```
PythonBasic - Jupyter Notebook
   [87]: b = '''kjghb
          kjhliu
         lkhkj'''
          print(b)
          kjghb
          kjhliu
          lkhkj
In [88]: b = 'jytf
          jhgf
          kjhg'
          print(b)
            Cell In[88], line 1
              b = 'jytf
          SyntaxError: unterminated string literal (detected at line 1)
In [36]: | a = 'welcome to cttc'
          type(a)
Out[36]: str
In [37]: #indexing
          #var[index number]
          a[2]
Out[37]: '1'
In [38]: a[8]
Out[38]: 't'
In [39]: a[9]
Out[39]: 'o'
In [ ]:
```

String Slicing:

Syntax : var[start_index : stop_index+1 : step]



```
ENT Change Propile Land Control of the Control of t
```

Q. Extract the following sub-string from string

s1 = "welcome to my blog"

```
i) 'loet'
ii) 'om lg'
iii) 'wloet
iv) 'emoclew'
```

s2 = "aeroplane"

```
i) 'pore'ii) 'lane'iii) 'apra'
```



```
E CHANGE FOR THE STATE OF THE S
```

```
[101]: s1 = "welcome to my blog"
          #i) 'loet'
          s1[2:10:2]
Out[101]: 'loet'
 In [47]: #ii) 'om lg'
          s1[9:18:2]
 Out[47]: 'om lg'
 In [48]: #iii) 'wloet
          s1[0:9:2]
 Out[48]: 'wloet'
 In [49]: s1[::-1]
 Out[49]: 'golb ym ot emoclew'
In [52]: #iv) 'emoclew'
          s1[6::-1]
Out[52]: 'emoclew'
In [102]: s1[-9::-1]
Out[102]: 'ot emoclew'
```





String Attributes:-

capitalize() Converts the first character to upper case			
casefold() Converts string into lower case			
count() Returns the number of times a specified value occurs			
in a string			
endswith() Returns true if the string ends with the specified va			
lue			
find() Searches the string for a specified value and returns			
first the position of where it was found			
isalnum() Returns True if all characters in the string are alph			
anumeric			
isalpha() Returns True if all characters in the string are in t			
he alphabet			
isdigit() Returns True if all characters in the string are digi			
t			
islower() Returns True if all characters in the string are lowe			
r case			
isspace() Returns True if all characters in the string are whit			
espaces			
istitle() Returns True if the string follows the rules of a tit			
le			
isupper() Returns True if all characters in the string are uppe			
r case			
isnumeric() Returns True if the string contains only numbers			
lower() Converts a string into lower case			
split() Splits the string at the specified separator, and ret			
urns a list			
swapcase() Swaps cases, lower case becomes upper case and vice v			
ersa			
title() Converts the first character of each word to upper ca			
se			
upper() Converts a string into upper case			
strip() Used to remove starting and ending whitespace.			
center(length, character) Returns a centered string			
string.join(iterable) takes all items in an iterable and join			
s them into one string.			
replace(oldvalue, newvalue) replaces a specified phrase with anothe			
r specified phrase.			

All functions applied to the strings are involved for the same instance. They do not affect the original string permanently.





```
In [99]: #capitalize():
          #var.attributename()
          s1 = "welcome to my blog"
          s1.capitalize()
Out[99]: 'Welcome to my blog'
In [100]: s1
Out[100]: 'welcome to my blog'
 In [58]: |#upper()
          s1 = "welcome to my blog"
          s1.upper()
Out[58]: 'WELCOME TO MY BLOG'
 In [59]: #Lower()
          s2 = 'WELCOME TO MY BLOG'
          s2.lower()
Out[59]: 'welcome to my blog'
 In [60]: #casefold()
          s2 = 'WELCOME TO MY BLOG'
          s2.casefold()
 Out[60]: 'welcome to my blog'
 In [61]: #title()
          b = 'welcome to my blog'
          b.title()
Out[61]: 'Welcome To My Blog'
 In [62]: |#count()
          b = 'welcome to my blog'
          b.count('o')
Out[62]: 3
 In [63]: #find()
          b = 'welcome to my blog'
          b.find('o')
Out[63]: 4
```



```
PythonBasic - Jupyter Notebook
```

```
In [66]: #replace(oldvalue, newvalue)
         b = 'welcome to blog'
         b.replace('blog','cttc')
Out[66]: 'welcome to cttc'
In [67]: #center()
         c = 'welcome'
         c.center(17, ' (**)
Out[67]: ' ♥ ♥ ♥ ♥ ♥ welcome ♥ ♥ ♥ ♥ ♥ '
In [68]: #join()
         c = 'welcome'
         ' % '.join(c)
Out[68]: 'w % e % 1 % c % o % m % e'
In [69]: #swapcase()
         c = 'Welcome'
         c.swapcase()
Out[69]: 'wELCOME'
In [70]: |#split()
         b = 'welcome to my blog'
         b.split()
Out[70]: ['welcome', 'to', 'my', 'blog']
In [71]: | b = 'welcome; to; my; blog'
         b.split(';')
Out[71]: ['welcome', 'to', 'my', 'blog']
In [72]: |#isupper()
         a = 'PYTHON'
         a.isupper()
Out[72]: True
In [73]: #isupper()
         a = 'pYTHON'
         a.isupper()
Out[73]: False
```





```
a = 'python'
         a.islower()
Out[74]: True
In [75]: #isdigit()
         a = '123'
         a.isdigit()
Out[75]: True
In [76]: #istitle()
         a = 'Welcome To Cttc'
         a.istitle()
Out[76]: True
In [77]: #isalnum()
         a = '123'
         a.isalnum()
Out[77]: True
In [78]: b = 'ahjsd'
         a.isalnum()
Out[78]: True
In [79]: | c = '12sajhf'
         c.isalnum()
Out[79]: True
In [80]: | d = '12fae@'
         d.isalnum()
Out[80]: False
In [81]: #isspace()
         d = ' '
         d.isspace()
Out[81]: True
```



!pip install pywhatkit

Requirement already satisfied: pywhatkit in c:\users\msi 1\anaconda3\lib\site -packages (5.4)

Requirement already satisfied: Pillow in c:\users\msi 1\anaconda3\lib\site-pa ckages (from pywhatkit) (9.4.0)

Requirement already satisfied: pyautogui in c:\users\msi 1\anaconda3\lib\site -packages (from pywhatkit) (0.9.54)

Requirement already satisfied: requests in c:\users\msi 1\anaconda3\lib\site-packages (from pywhatkit) (2.31.0)

Requirement already satisfied: wikipedia in c:\users\msi 1\anaconda3\lib\site -packages (from pywhatkit) (1.4.0)

Requirement already satisfied: Flask in c:\users\msi 1\anaconda3\lib\site-pac kages (from pywhatkit) (2.2.2)

Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\msi 1\anaconda3\lib\site-packages (from Flask->pywhatkit) (2.2.3)

Requirement already satisfied: Jinja2>=3.0 in c:\users\msi 1\anaconda3\lib\si te-packages (from Flask->pywhatkit) (3.1.2)

Requirement already satisfied: itsdangerous>=2.0 in c:\users\msi 1\anaconda3 \lib\site-packages (from Flask->pywhatkit) (2.0.1)

Requirement already satisfied: click>=8.0 in c:\users\msi 1\anaconda3\lib\sit e-packages (from Flask->pywhatkit) (8.0.4)

Requirement already satisfied: pymsgbox in c:\users\msi 1\anaconda3\lib\site-packages (from pyautogui->pywhatkit) (1.0.9)

Requirement already satisfied: pytweening>=1.0.4 in c:\users\msi 1\anaconda3 \lib\site-packages (from pyautogui->pywhatkit) (1.2.0)

Requirement already satisfied: pyscreeze>=0.1.21 in c:\users\msi 1\anaconda3 \lib\site-packages (from pyautogui->pywhatkit) (0.1.30)

Requirement already satisfied: pygetwindow>=0.0.5 in c:\users\msi 1\anaconda3 \lib\site-packages (from pyautogui->pywhatkit) (0.0.9)

Requirement already satisfied: mouseinfo in c:\users\msi 1\anaconda3\lib\site -packages (from pyautogui->pywhatkit) (0.1.3)

Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\msi 1\ana conda3\lib\site-packages (from requests->pywhatkit) (2.0.4)

Requirement already satisfied: idna<4,>=2.5 in c:\users\msi 1\anaconda3\lib\s ite-packages (from requests->pywhatkit) (3.4)

Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\msi 1\anaconda3 \lib\site-packages (from requests->pywhatkit) (1.26.16)

Requirement already satisfied: certifi>=2017.4.17 in c:\users\msi 1\anaconda3 \lib\site-packages (from requests->pywhatkit) (2025.1.31)

Requirement already satisfied: beautifulsoup4 in c:\users\msi 1\anaconda3\lib \site-packages (from wikipedia->pywhatkit) (4.12.2)

Requirement already satisfied: colorama in c:\users\msi 1\anaconda3\lib\site-packages (from click>=8.0->Flask->pywhatkit) (0.4.6)

Requirement already satisfied: MarkupSafe>=2.0 in c:\users\msi 1\anaconda3\lib\site-packages (from Jinja2>=3.0->Flask->pywhatkit) (2.1.1)

Requirement already satisfied: pyrect in c:\users\msi 1\anaconda3\lib\site-pa ckages (from pygetwindow>=0.0.5->pyautogui->pywhatkit) (0.2.0)

Requirement already satisfied: soupsieve>1.2 in c:\users\msi 1\anaconda3\lib \site-packages (from beautifulsoup4->wikipedia->pywhatkit) (2.4)

Requirement already satisfied: pyperclip in c:\users\msi 1\anaconda3\lib\site -packages (from mouseinfo->pyautogui->pywhatkit) (1.8.2)



```
[83]: import pywhatkit as kit
```



```
In [84]: kit.playonyt('Ishq de')
```

Out[84]: 'https://www.youtube.com/watch?v=rnyrDk4G68g\\\u0026pp=ygUHSXNocSBkZQ%3D%3D'

```
In [85]: kit.info('What is python')
```

Python is a high-level, general-purpose programming language. Its design phil osophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked and garbage-collected.

```
In [86]: kit.search('rose flower')
```

List Datatype :- It is a sequence of items, inside a [], items are separated by comma.

```
A list is mutable : items can be modified.
A list is indexed : items have index numbers that can be used for ind exing and slicing.
```

```
In [89]: lis = [4,5,6,7,8,34]
type(lis)

Out[89]: list

In [90]: #slicing
lis[0:6:2]

Out[90]: [4, 6, 8]

In []:
```

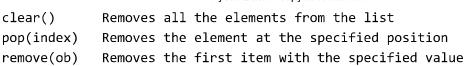
List Attributes:-

Adding items to list:

```
append(ob) Adds an element at the end of the list extend(seq) Add the elements of a list (or any iterable), to the end of the current list insert(index,ob) Adds an element at the specified position
```

Removing items from list:







Miscellaneous functions:

```
copy()

Returns a copy of the list

count(ob)

Returns the number of elements with the specifie

d value

index(ob)

Returns the index of the first element with the

specified value

reverse()

Reverses the order of the list
```

Adding elements to list.

```
In [91]:
         #append()
         lis = [4,5,6,7,8,34]
         lis.append('cttc')
         print(lis)
         [4, 5, 6, 7, 8, 34, 'cttc']
In [92]: #extend()
         lis = [4,5,6,7,8,34]
         lis.extend(['cttc','bbsr','ctc'])
         print(lis)
         [4, 5, 6, 7, 8, 34, 'cttc', 'bbsr', 'ctc']
In [93]: #insert()
         lis = [4,5,6,7,8,34]
         lis.insert(5,'cttc')
         print(lis)
         [4, 5, 6, 7, 8, 'cttc', 34]
```

Deleting element from a list.

```
In [95]: #pop()
lis = [4,5,6,7,8,34]
lis.pop(4)
print(lis)
[4, 5, 6, 7, 34]
```





```
#remove()
         lis = [4,5,6,7,8,34]
         lis.remove(8)
         print(lis)
         [4, 5, 6, 7, 34]
In [98]: #clear()
         lis.clear()
         print(lis)
         []
 In [ ]:
         Other Operations on list
 In [ ]:
```