Variables

- variables are used to store the value
- and the values can be change

In [1]:
```
number=100  # value 100 stored in a variable name called 'number'
number      # shift+enter
```

Out[1]: 100

In [2]:
```
number=200 # value 200 stored in a variable name called 'number'
number
```

Out[2]: 200

Note

- python is step by step process
- python wil take last output as the latest value
- in above code we are using same variable: number
- but number value changed two times
- first time 100
- second time 200
- latest value 200

In [3]:
```
NUMBER=200
NUMBER
```

Out[3]: 200

In [4]:
```
number, NUMBER
```

Out[4]: (200, 200)

In [5]:
```
NUMber=300
NUMber
```

Out[5]: 300

In [6]:
```
number123=400
number123
```

Out[6]: 400

In [7]:
```
123number=500
123number
```

```
  Cell In[7], line 1
    123number=500
      ^
SyntaxError: invalid decimal literal
```

In [8]:
```
numb%er=600
numb%er
```

```
  Cell In[8], line 1
    numb%er=600
    ^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of
'='?
```

In [9]:
```
number one=800
number one
```

```
  Cell In[9], line 1
    number one=800
           ^
SyntaxError: invalid syntax
```

In [10]:
```
number_one=900
number_one
```

Out[10]: 900

In [11]:
```
_=1000
_
```

Out[11]: 1000

In [12]:
```
n1=100,200
n1
```

Out[12]: (100, 200)

In [13]:
```
n2,n3=200
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 n2,n3=200

TypeError: cannot unpack non-iterable int object
```

In [17]:
```
n4,n5=100,200
print(n4)
print(n5)
```

```
100
200
```

In [18]:
```
n4,n5=100,200
n4
n5
```

Out[18]: 200

In [ ]:
```
number=200  # w
NUMBER=300  # w
NUMber=400  # w
number123=500 # w
123number=600  # F
num%er=700    # F
number one=800 #F
number_one=900 # W
_=1000 # W
n1=100,200 # W
n1,n2=100  # F
n1,n2=100,200 # W
```

In [19]:
```
n1,_=200,300
```

In [20]:
```
if=100
```

```
  Cell In[20], line 1
    if=100
       ^
SyntaxError: invalid syntax
```

In [ ]:
```
if
while
else
elif
for
True
False
```

- Python variables are case sensitive
- Variables can be declare as capital letters
- Variables can be declare as small letters
- Variables can not start with Numbers (prefix)
- Numbers can be suffix of the variables
- Variables can not have special charcters except **underscore**
- Variables does not have space between the names
- Only undesrcore is possibile
- Keywords or reserved words can not use as variables
- two values can be assign to a single variable
- one value can not assign to two variables

In [ ]:
```
True
```

```
true
```