In [30]:

```
print("tanoj")
```

```
tanoj
```

In [53]:

```
age = 20
print('age')
```

```
age
```

In [54]:

```
print(age)
```

```
20
```

In [57]:

```
FName,MName,LName = 'N','Satya','Tanoj'
print(FName)
print(MName)
print(LName)
```

```
N
Satya
Tanoj
```

In [ ]:

```

```

In [78]:

```
x=y=z = "abc"
print(x)
print(y)
print(z)
print(x+ 5*' ' +z)
```

```
abc
abc
abc
abc     abc
```

In [7]:

```
first_name = "Tanoj"
last_name = "N"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
Tanoj N
```

In [86]:

```
#Spaces

name = "tanoj"
print(name)
print(name.rjust(20))      # rjust gives spaces in the right. here its 20 spaces
print(name.ljust(20))      # ljust gives spaces in the left. here its 20 spaces
print(name.center(20))     # center gives spaces in the center. here its 20 spaces
print(' '.join(name))
```

```
tanoj
               tanoj
tanoj
       tanoj
t a n o j
```

In [8]:

```
## addressing whitespaces
```

In [9]:

```
print("Python")
```

```
Python
```

In [12]:

```
print("\tpython") # print using tab space
```

```
        python
```

In [13]:

```
print("\nPython") # print in new line
```

```

Python
```

In [14]:

```
print("Languages:\nPython\nScala\nJava")
```

```
Languages:
Python
Scala
Java
```

In [17]:

```
x = "value"
```

In [6]:

```
x = "  value   "

print(x)
```

```
  value
```

In [7]:

```
print(x.strip())    # removes the white spaces
```

```
value
```

In [21]:

```
# Numbers
```

In [22]:

```
2+1
```

Out[22]:

```
3
```

In [23]:

```
3/2
```

Out[23]:

```
1.5
```

In [101]:

```
x=1
y=2
z=3
print(x)
```

```
1
```

In [103]:

```
x,y,z,a=1,2.2,3j,'tj'  # advisible to declare a variable like this when we need to assi
gn more variables
```

In [105]:

```
print(type(x))    # type() give type of number provided
print(type(y))
print(type(z))
print(type(a))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
```

In [109]:

```
b = int(y)   #we can convert the data type from int-float, float-int, int-complex
c = int(z)
print(c)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-109-94c504487dc6> in <module>
      1 b = int(y)
----> 2 c = int(z)
      3 print(c)

TypeError: can't convert complex to int
```

In [ ]:

```

```

In [ ]:

```

```

In [27]:

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

In [33]:

```
import sys
```

In [1]:

```
#List - collection of items in a particular and it is mutuble daya type.
#String is an immutable data type

#it can be assigned by using []
```

In [10]:

```
bicycles = ['trek','redline','hero']
# elements will be accessing with 0 by indexing eg 0,1,2,3,....

print(bicycles[0])              # here to print 1st element
print(bicycles[0].title())      #capital first letter
```

```
trek
Trek
```

In [18]:

```
# assigning a new value i.e. change trek to ranger

bicycles[2] = 'ranger'  # if we replace 0 with 2 in the 2nd index place
print(bicycles)
```

```
['ranger', 'redline', 'ranger']
```

In [19]:

```
#insert and apend
print(bicycles)
```

```
['ranger', 'redline', 'ranger']
```

In [20]:

```
bicycles[2] = 'hero'
bicycles[0] = 'trek'
print(bicycles)
```

```
['trek', 'redline', 'hero']
```

In [22]:

```
bicycles.append('ranger')   # here it appends ranger to 4th position
print(bicycles)
```

```
['trek', 'redline', 'hero', 'ranger']
```

In [23]:

```
#include the element to desigred index
#to achive this use 'insert'
bicycles.insert(1,'ranger')    #index,value
print(bicycles)
```

```
['trek', 'ranger', 'redline', 'hero', 'ranger']
```

In [24]:

```
#how to delete a value from list
# pop() : to delete

bicycles.pop()
print(bicycles)
```

```
['trek', 'ranger', 'redline', 'hero']
```

In [25]:

```
bicycles.pop()
print(bicycles)    # from the above outputs pop() deletes only the LAST value
```

```
['trek', 'ranger', 'redline']
```

In [27]:

```
bicycles.pop(0)
print(bicycles)    #when we provide index value, it deletes the desired value
```

```
['ranger', 'redline']
```

In [36]:

```
cars = ['bmw','audi','jaugur','benz']

#sorting a list -- using sort()

cars.sort()                # default sort
cars.reverse()  # reverse sort .. we can also use "sort(reverse = True)"
print(cars)
print('cars:',cars )
```

```
['jaugur', 'bmw', 'benz', 'audi']
cars: ['jaugur', 'bmw', 'benz', 'audi']
```

In [37]:

```
# create an empty list

bikes = []
print("ini")
print(bikes)
```

```
ini
[]
```

In [8]:

```
bikes = [ 'yamaha','RE']
print(bikes)
```

```
['yamaha', 'RE']
```

In [ ]:

```
# Negative indexing

indexing is 0,1,2,3,4...
if there are 1000 elements and the 1000 th element will be indexed as -1
from last to first -1,-2,-3,-4.....
```

In [42]:

```
cars = ['audi','bmw','bugatti','ferrari']
print(cars[-1])          #-ve indexing
```

```
ferrari
```

In [45]:

```
# For loops in lists

Students = ['I','Me','Myself']
for student in Students:
    print(student)          # student is a temp variable
                            # space before print is called INDENTATION(maintaining a pr
oper format). It takes 4 spaces
```

```
I
Me
Myself
```

In [49]:

```
Students = ['I','Me','Myself']
for y in Students:
    print(f"{y.title()}, attendent of python")
#Eg: getting a bday wishes from ol stores. Message will be given and for will be names
```

```
I, attendent of python
Me, attendent of python
Myself, attendent of python
```

In [50]:

```
Students = ['I','Me','Myself']
for a in Students:
    print(f"{a.title()}, attendent of python")   # f string

print("Thnq for participation")   # this print is out of for loop
```

```
I, attendent of python
Me, attendent of python
Myself, attendent of python
Thnq for participation
```

In [89]:

```
Android = ['One+','Moto']
IOS = ['Apple']
for phone in Android:
    print(f"{phone.title()}, is Android")
for phone in IOS:
    print(f"{phone.title()}, is IOS")

print("Are popular brands")
```

```
One+, is Android
Moto, is Android
Apple, is IOS
Are popular brands
```

In [53]:

```
Students = ['I','Me','Myself']
for a in Students
    print(f"{a.title()}, attendent of python")
```

```
  File "<ipython-input-53-f5dc8d216456>", line 2
    for a in Students
                     ^
SyntaxError: invalid syntax
```

In [54]:

```
Students = ['I','Me','Myself']
for a in Students:
print(f"{a.title()}, attendent of python")
```

```
  File "<ipython-input-54-fc9c72351e11>", line 3
    print(f"{a.title()}, attendent of python")
        ^
IndentationError: expected an indented block
```

In [58]:

```
# working with numerical list

numbers = list(range(1,6))   #RANGE -- inbuilt function to give the range
print(numbers)
```

```
[1, 2, 3, 4, 5]
```

In [63]:

```
for value in range(1,11):
    print(value)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [39]:

```
squares = []  # create an empty list
for value in range(1,16):
    square = value**2     # * multiplies once and ** multiplies twice
    squares.append(square)

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225]
```

In [38]:

```
#slicing of list:

places = ['vizag','hyderabad','banglore','munnar','alleppy','chennai']
print(places[0:2])
print(places[2:4])
print(places[2:])
print(places[:2])      #it takes index of 0,1
```

```
['vizag', 'hyderabad']
['banglore', 'munnar']
['banglore', 'munnar', 'alleppy', 'chennai']
['vizag', 'hyderabad']
```

In [41]:

```
print('---PLACES---')
for place in places[:2]:
    print(place.title())
```

```
---PLACES---
Vizag
Hyderabad
```

In [51]:

```
# copying a list

my_drinks = ['coke','thumbsup','fanta','due','pepsi']
friends_drinks = soft_drinks[:]      #copied
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
<ipython-input-51-213f580b0073> in <module>
      2
      3 my_drinks = ['coke','thumbsup','fanta','due','pepsi']
----> 4 friends_drinks = soft_drinks[:]      #copied

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

In [36]:

```
print(friends_drinks)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
<ipython-input-36-50634a40d2ea> in <module>
----> 1 print(friends_drinks)

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

In [34]:

```
my_drinks.append('mazza')

friends_drinks.append('sprite')

print(my_drinks)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
<ipython-input-34-ee8f73e2ece6> in <module>
----> 1 my_drinks.append('mazza')
      2
      3 friends_drinks.append('sprite')
      4
      5 print(my_drinks)

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

In [ ]:

```
# introduction to Tuple - Data type in python
# once assigned, it cannot be changed i.e t cant be appended or changed. Its immutable
 -- defined by ()
```

In [20]:

```
dimensions= (10,20)

print(dimensions[0])
print(dimensions[1])
```

```
10
20
```

In [22]:

```
dimensions[0] = 30   # as it is immutable, we can not reassign the value. we need to de
clare again to achive this
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-22-a5db7d9e1978> in <module>
----> 1 dimensions[0] = 30

TypeError: 'tuple' object does not support item assignment
```

In [24]:

```
dimensions = (30,40)  # if there is a request from business to change, we have to decla
re again

print(dimensions[0])
print(dimensions[1])
```

```
30
40
```

In [110]:

```
print(dimensions[-1])  #-ve indexing i.e from last value
```

```
40
```

In [112]:

```
dimensions = (50,60)   # redefining a tuple. we need to declare from 1st again to redef
ine
print(dimensions)
```

```
(50, 60)
```

In [121]:

```
#for loopinf in tuple

diminsions = (10,20,30)
print("Diminsion defined")
for diminsion in diminsions:
    print(diminsion)

diminsions = (50,60,70)

print("\nModified diminsion value")
for diminsion in diminsions:
    print(diminsion)
```

```
Diminsion defined
10
20
30

Modified diminsion value
50
60
70
```

In [ ]:

```
# if statements -- test the conditions defined
#= is assigning a value
#== is equality condition
```

In [120]:

```
bikes = ['enfield','yamaha','duke','suziki','renegade']

for bike in bikes:
    if bike == 'renegade':
        print(bike.upper())
    else:
        print(bike.title())
```

```
Enfield
Yamaha
Duke
Suziki
RENEGADE
```

In [124]:

```
#Python is case-sensitive

name = "Tanoj"

name == "Tanoj"   #checks if true or false
```

Out[124]:

```
True
```

In [126]:

```
name.lower() == 'tanoj'    #here we r converting to lower case
```

Out[126]:

```
True
```

In [129]:

```
# != is not equal to

req_topping = "Mashroom"

if req_topping != 'Chicken':
    print("get chicken")
```

```
get chicken
```

In [ ]:

In [ ]:

In [ ]: