

In []: #Dictionary

```
In [7]: booking={  
    "1021": {"flight_name": "indigo", "seat_no": 102, "status": "Sucesss"},  
    "1022": {"flight_name": "indigo", "seat_no": 103, "status": "Pending"},  
    "1023": {"flight_name": "indigo", "seat_no": 104, "status": "Failed"}  
}  
  
for item in booking:  
    print(f"boking_id:{item},details:{booking[item]}")
```

```
boking_id:1021,details:{'flight_name': 'indigo', 'seat_no': 102, 'status': 'Sucesss'}  
boking_id:1022,details:{'flight_name': 'indigo', 'seat_no': 103, 'status': 'Pending'}  
boking_id:1023,details:{'flight_name': 'indigo', 'seat_no': 104, 'status': 'Failed'}
```

In []: #if statement

```
In [6]: num=10  
if None:    #it will not allow to execute the Loop [None,0,False,""]  
    print("None")  
print("this print always")
```

this print always

```
In [9]: if -1:    #it will not allow to execute the Loop  
    print("None")  
print("this print always")
```

None

this print always

```
In [12]: score=67  
if score>90 and score<=100:  
    print("grade=A")  
elif score>80 and score<=90:  
    print("grade=B")  
elif score>70 and score<=80:  
    print("grade=C")  
elif score>60 and score<=70:  
    print("grade=D")  
elif score>50 and score<=60:  
    print("grade=E")  
else:  
    print("grade=F")
```

grade=D

In []: num=46

```
if num>30 and num<=50:  
    if num%2==0:  
        print("even")  
    else:  
        print("odd")
```

```
In [ ]: #Range rang(start,stop,step)
```

```
In [13]: x=[10,20,30,40]
for num in x:
    print(num)
else:
    print("execute after for loop exit")
```

```
10
20
30
40
execute after for loop exit
```

```
In [15]: x=[10,20,30,40]
for num in x:
    print(num)
    if num%2==0:
        print("running")
else:
    print("execute after for loop exit")
```

```
10
running
20
running
30
running
40
running
execute after for loop exit
```

```
In [ ]:
```

```
In [33]: num=7
flag=0
for i in range(2,int(num**0.5)+1):
    if num%i==0:
        flag=1
        break
if flag==0:
    print("prime number")
else:
    print("not a prime number")
```

```
prime number
```

```
In [ ]: value1=int(input("Enter the first Value"))
value2=int(input("Enter the first Value"))
def prime(num):
    if num<2:
        return True
    for i in range(2,int(num**0.5)+1):
        if num%i==0:
            return False
    return True
print(f"write prime number between {value1},{value2}")
for i in range(value1,value2+1):
    if prime(i):
        print(i)
```

```
In [ ]: #continue => allows only false condition to process or loop items ( does not allow
#break=> it come out from the loop, once condition true else it will not execute )
```

```
In [7]: x=[10,20,30,40]
for i in x:
    if i%2==0:
        continue
    print(num)
else:
    print("even number")
```

even number

```
In [9]: #Find element greater than 50
x=[50,70,30,80,10,24,56]
for i in x:
    if i>50:
        print(i)
    else:
        continue
```

70
80
56

```
In [5]: shop=["Hp","dell","lenovo","mac"]
while True:
    laptop=input("Enter the laptop brand")
    if laptop in shop:
        print("Allowed to purchase")
        break
    else:
        print("Try searching another brand")
        continue
```

Try searching another brand
Allowed to purchase

```
In [2]: #Append and Extend

list1=[1,2,3,4,5]
list2=["blue","green","Orange","Red"]
list1.append(list2)
print(list1)
```

[1, 2, 3, 4, 5, ['blue', 'green', 'Orange', 'Red']]

```
In [3]: list1=[1,2,3,4,5]
list2=["blue","green","Orange","Red"]
list1.extend(list2)
print(list1)
```

[1, 2, 3, 4, 5, 'blue', 'green', 'Orange', 'Red']

```
In [8]: #Overridden
lst=[20,405,5049,39,404]
lst1=[]
lst1=lst
print(lst)
print(lst1)
lst1.append(4)
```

```
lst1.append(5)
print(lst)
print(lst1)

[20, 405, 5049, 39, 404]
[20, 405, 5049, 39, 404]
[20, 405, 5049, 39, 404, 4, 5]
[20, 405, 5049, 39, 404, 4, 5]
```

In [12]:

```
#shallow copy
lst=[20,44,5994,5953]
lst1=[]
lst1.copy(lst)
print(lst)
print(lst1)
lst1.append(4)
lst1.append(5)
print(lst)
print(lst1)
```

In [13]:

```
lst=[20,304,45,60,79,80]
print(lst[-4])
```

45

In [14]:

```
lst1=["blue","Yellow","Red","Orange"]
lst2=[1,3,5,6,3,45]
lst3=lst1+lst2
print(lst3)
```

['blue', 'Yellow', 'Red', 'Orange', 1, 3, 5, 6, 3, 45]

In [17]:

```
power=[i**2 for i in range(10)]
print(power)
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

In [24]:

```
#using Fuunction and without Function Transformation
matrix=[
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12]
]
# transpose=[[matrix[j] for j in range(len(matrix))] for i in range(len(matrix[0]))]

transpose=[]
for i in range():
print(transpose)

[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]], [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]], [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]]
```

In [20]:

```
#nested tuple
tup=(1,(1,2,3),["Green","Yello"],True)
print(type(tup))
print(tup[2])
print(tup[1][-1])

<class 'tuple'>
['Green', 'Yello']
```

3

```
In [25]: tup=("hello",)
print(tup)
```

('hello',)

```
In [26]: tup="hello"
print(tup)
```

hello

```
In [27]: lst=(1,3,["hello",2],True)
lst[2][0]="bye"
```

```
print(lst)
```

(1, 3, ['bye', 2], True)

```
In [28]: #Strings are immutable
st="Rithul"
st[5]="t"
```

TypeError

Traceback (most recent call last)

Cell In[28], line 3

```
1 #Strings are immutable
2 st="Rithul"
----> 3 st[5]="t"
```

TypeError: 'str' object does not support item assignment

```
In [31]: ls=(("hello",)*5)
print(ls)
```

('hello', 'hello', 'hello', 'hello', 'hello')

```
In [33]: # In tuple we cannot delete or remove element
# Delete entire tuple using del
lst=(1,3,4,5,7)
del(lst)
```

```
In [34]: #tuple indexing
tup=(1,44,5,6,7,8,2,5)
print(tup.index(5))
```

2

```
In [36]: tup=(1,44,5,6,7,8,2,5)
if 5 in tup:
    print("present")
if 9 not in tup:
    print("not present")
```

present

not present

```
In [39]: tup=(1,44,5,6,7,8,2,5)
tup2=sorted(tup,reverse=True)
print(tup2)
```

[44, 8, 7, 6, 5, 5, 2, 1]

```
In [40]: import statistics as lst  
tup=(1,44,5,6,7,8,2,5)  
print(lst.mean(tup))
```

9.75

```
In [41]: print(lst.median(tup))
```

5.5

```
In [42]: print(lst.mode(tup))
```

5

```
In [46]: import math  
math.factorial(5)
```

Out[46]: 120

```
In [47]: lst={1,2,2,4,6,6,3}  
print(lst)
```

{1, 2, 3, 4, 6}

```
In [51]: ##### ASSIGNMENT 2 #####  
#Copying element without changing the memory address (Shallow copy)  
lst=[4,5,6,7,8]  
lst1=[]  
  
lst1=list.copy(lst)  
print(lst1)  
  
print(id(lst))  
print(id(lst1))
```

[4, 5, 6, 7, 8]

2358141847168

2358141844224

```
In [52]: lst={12,34,6,7,3,3,57393,45}  
print(lst)  
lst.add(45) #set is unordered  
print(lst)
```

{57393, 34, 3, 6, 7, 12, 45}

{57393, 34, 3, 6, 7, 12, 45}

```
In [53]: lst.update([8,4,56,2]) # add multiple elements  
print(lst)
```

{2, 3, 4, 6, 7, 8, 12, 34, 45, 57393, 56}

```
In [57]: lst={1,2,3,5,6}  
lst.remove(9)  
print(lst)
```

```

KeyError ..... Traceback (most recent call last)
Cell In[57], line 2
  1 lst={1,2,3,5,6}
----> 2 lst.remove(9)
  3 print(lst)

KeyError: 9

```

```
In [56]: lst.discard(9)
print(lst)
```

```
{1, 2, 3, 5, 6}
```

```
In [1]: lst={1,2,3,4,5,6,7,8,9}
a=lst.pop()
print(a)
b=lst.pop()
print(b)
print(lst)
```

```
1
2
{3, 4, 5, 6, 7, 8, 9}
```

```
In [ ]: ##### Assignment 3 #####
#reassign the Deleted data
```

```
In [2]: lst1={1,2,3,4,5,6}
lst2={5,6,7,8,9}
print(lst1 | lst2)
print(lst1.union(lst2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
In [3]: lst1=frozenset([12,3,4,5,6,67])
lst2=frozenset([2,3,5,6,7])
lst1.add(5)
print(lst1)
```

```

AttributeError ..... Traceback (most recent call last)
Cell In[3], line 3
  1 lst1=frozenset([12,3,4,5,6,67])
  2 lst2=frozenset([2,3,5,6,7])
----> 3 lst1.add(5)
  4 print(lst1)

AttributeError: 'frozenset' object has no attribute 'add'

```

```
In [1]: #Dictionary
my_dic={1:"Rithul",2:"Ayush"}
my_dic[1]
```

```
Out[1]: 'Rithul'
```

```
In [2]: my_dic[3] # it will raise error when key is not found
```

```

-----+-----+-----+-----+-----+-----+
KeyError ..... Traceback (most recent call last)
Cell In[2], line 1
----> 1 my_dic[3]

KeyError: 3

In [3]: my_dic.get(2)

Out[3]: 'Ayush'

In [4]: my_dic.get(3) # will handle the keyerror if key is not found in dictionary

In [5]: #Add or Modify
my_dic[5]=""

In [6]: #pop
my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}

my_dic.pop(2)

Out[6]: 'Ayush'

In [7]: my_dic

Out[7]: {1: 'Rithul', 3: 'Akash', 4: 'Amar'}

In [8]: my_dic.popitem()

Out[8]: (4, 'Amar')

In [9]: my_dic

Out[9]: {1: 'Rithul', 3: 'Akash'}

In [10]: #delete key
my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}

del my_dic[3]
print(my_dic)

{1: 'Rithul', 2: 'Ayush', 4: 'Amar'}

In [11]: #clear entire dictionary
my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}
my_dic.clear()
print(my_dic)

{}

In [12]: #delete entire dictionary never exist
my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}

del my_dic
print(my_dic)

```

```
NameError Traceback (most recent call last)
Cell In[12], line 5
      2 my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}
      3 del my_dic
----> 5 print(my_dic)

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

```
In [13]: #copy
my_dic={1:"Rithul",2:"Ayush",3:"Akash",4:"Amar"}
mydic2=my_dic.copy()
print(mydic2)

{1: 'Rithul', 2: 'Ayush', 3: 'Akash', 4: 'Amar'}
```

```
In [15]: print(id(my_dic))
print(id(mydic2))

2436919909440
2436919915520
```

```
In [16]: #dict.fromkeys()

sub=dict.fromkeys(["A","B","C"],0)
print(sub)

{'A': 0, 'B': 0, 'C': 0}
```

```
In [17]: squ={2:4,4:6,6:7,9:8}
print(squ.items())

dict_items([(2, 4), (4, 6), (6, 7), (9, 8)])
```

```
In [18]: print(squ.values())

dict_values([4, 6, 7, 8])
```

```
In [19]: print(squ.keys())

dict_keys([2, 4, 6, 9])
```

```
In [20]: lid={}
print(dir(lid))

['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

```
In [22]: #Dictionary operation

dic={"maths":89,"CS":90,"Science":45,"English":34}

dic1={k:v for k,v in dic.items() if v>50}
print(dic1)

{'maths': 89, 'CS': 90}
```

In []: