In [13]:
```
# collections -> list, tuple, set, dictionary
# list -> [], duplicates are allowed, ordered / indexed, mutable(part of data can be c

# names = [] # list()
# print(names, type(names))

# names = ['Abhinav', 'Piyush', 'jay', 'Nayeem','ratik' 'Shubam']
# # print(names)
# print(names[0], names[1], names[4])

nums = [1, 2, 3, 5, 4]
print(nums)
```

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

In [22]:
```
# values = [1, 'vasi', True, None, 10.65, 1+3j]
# print(values)

nums = [1, 1, 2, 3, 4, 3, 2]
print(nums, nums[0])

nums[0] = 10 # mutable
print(nums, nums[0])
```

```
[1, 1, 2, 3, 4, 3, 2] 1
[10, 1, 2, 3, 4, 3, 2] 10
```

In [44]:
```
players = ['virat', 'dhoni', 'rohit', 'sachin', 'sehwag']
# print(players)
# pop -> remove the last value by default
# removed_player = players.pop()
# print(players, removed_player)
# removed_player = players.pop()
# print(players, removed_player)

# players.pop(2) # index is passed, the value at that index will get removed
# print(players)


# index_to_be_removed = players.index('sachin')
# index_to_be_removed = players.index('raina')
# players.pop(index_to_be_removed)
# print(players)

# players.remove('dhoni')
# players.remove('raina') # not suggested
# print(players)

# membership operator
# print('raina' in players)
# print('sachin' in players)

# players.clear()
# print(players)
```

```
[]
```

In [57]:
```
# add values
players = ['virat', 'dhoni', 'rohit', 'sachin', 'sehwag']
```

```
# players.append('raina') # add value at the end
# players.append('hardik')
# print(players)

# players.insert(0, 'raina') # insert(position, value)
players.insert(20, 'raina') # insert(position, value)
print(players, players[5])
```

```
['virat', 'dhoni', 'rohit', 'sachin', 'sehwag', 'raina'] raina
```

In [67]:

```
class_A = ['piyush', 'manorath', 'ankur', 'atanaska']
class_B = ['Bicky', 'Govardhan', 'Guru', 'Jacob']
# ['piyush', 'manorath', 'ankur', 'atanaska', 'Bicky', 'Govardhan', 'Guru', 'Jacob']

# new_class = class_A + class_B
# print(new_class)

# extend
class_A.extend(class_B)
print(class_A)
```

```
['piyush', 'manorath', 'ankur', 'atanaska', 'Bicky', 'Govardhan', 'Guru', 'Jacob']
```

In [70]:

```
names = ['piyush', 'manorath', 'ankur', 'atanaska']
print(names)
names.reverse()
print(names)
```

```
['piyush', 'manorath', 'ankur', 'atanaska']
['atanaska', 'ankur', 'manorath', 'piyush']
```

In [92]:

```
# sort
names = ['Piyush', 'manorath', 'ankur', 'atanaska', 'Zebra', '10', '20', '37']
# ankur, atanaska, manorath, piyush
# names.sort()
print(names)

# names.sort(reverse=True) # desc order
# print(names)


# names.sort(key = str.upper) # (key=str.lower)
# print(names)

# names.sort(reverse=True, key = str.lower) # (key=str.upper)
# print(names)


# # nums = [1, 2, 3, 1, 4, 7, 7, 7]
# names = ['vasi', 'vasi', 'raj']
# print(nums.count('raj'))
```

```
['Piyush', 'manorath', 'ankur', 'atanaska', 'Zebra', '10', '20', '37']
```

In [104…

```
# tuple -> (), duplicate values are allowed, indexed/ordered, immutable
# nums = () # tuple()
# print(nums)


nums = (1, 2, 4, 3, 3)
# print(nums)
# print(nums[0], nums[1], nums[3])

# nums[0] = 10
# print(nums) # error

print(nums.count(3))
```

```
2
```

In [125…

```
# set -> {}, mutable, unordered / not indexed, duplicates are not allowed
# nums = set()
# print(nums, type(nums))


nums = {1, 2, 3, 4, 5, 5, 6}
# print(nums)

# add
# nums.add(7)
# print(nums)

# nums.add(8)
# print(nums)

# nums.remove(5)
# nums.remove(6)
# print(nums)

# nums.discard(10) # recommended
# nums.discard(6)
# print(nums)


# nums.clear()
# print(nums)
```

```
{1, 2, 3, 4, 5}
set()
```

In [136…

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7}
# print(set1.union(set2))
# print(set2.union(set1))

# print(set1.intersection(set2))
# print(set2.intersection(set1))


print(set1.difference(set2))
print(set2.difference(set1))
```

```
{1, 2, 3}
{6, 7}
```

In [139…

```
# dictionary -> key values, {}
person = {} # dict()
print(person, type(person))
```

```
{} <class 'dict'>
```

In [149…

```
person = {
    "name": "Rajni",
    "age": 70,
    "place": "chennai"
}


# print(person)
# print(person["name"], person["age"], person["place"])
# print(person.get("name"), person.get("age"), person.get("place", "India"))
```

```
Rajni 70 chennai
```

In [160…

```
person = {
    "name": "Rajni",
    "age": 70,
    "daughter": {
        "name": "Aishwarya",
        "husband": "Dhanush"
    }
}

# print(person.get("daughter").get("name"))

# remove
# person.get("daughter").pop("name")
# print(person.get("daughter"))

# clear
person.clear()
print(person)
```

```
{}
```

In [188…

```
# SLICING (start=0 or -1, end-1, step=1)
name = 'VASANTH'
# print(name[0:6])
# print(name[2:4])

# print(name[:6])
# print(name[2:])
# print(name[:])

# print(name[0:6:2])
# print(name[0:6:3])
# print(name[::2])

# print(name[-5:-2])
# print(name[-2:-5:-1])
# print(name[-6:-2:-2])
# print(name[1:-2])


# print(name[-2:0:-1])
print(name[::1])
print(name[::-1])
```

```
VASANTH
HTNASAV
```

In [191…

```
# conditional statements
```

```
is_voterid_available = False
if(is_voterid_available == True):
    print('You can vote')
else:
    print('You are minor')
```

```
You are minor
```

In [195…

```
# nested conditional statement
is_voterid_available = False
age = 16

if(is_voterid_available == True):
    if(age >= 18):
        print('You can vote')
    else:
        print('You are minor')
else:
    print('You are minor')
```

```
You are minor
```

In [198…

```
# nested conditionals statements with conditions
is_voterid_available = False
age = 16

if(is_voterid_available == True and age >= 18):
    print('You can vote')
else:
    print('You are minor')
```

```
You are minor
```

In [201…

```
bus = False
train = True

if(bus == True or train == True):
    print('You can reach office')
else:
    print('Boss is very happy')
```

```
You can reach office
```

In [206…

```
# n1, n2, n3 which is greater
num1 = 500
num2 = 100
num3 = 15

if(num1 > num2 and num1 > num3):
    print('num1 is greater')
elif(num2 > num1 and num2 > num3):
    print('num2 is greater')
else:
    print('num3 is greater')
```

```
num1 is greater
```

In [5]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 get_sum(4, 5)

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

```
In [ ]:
```