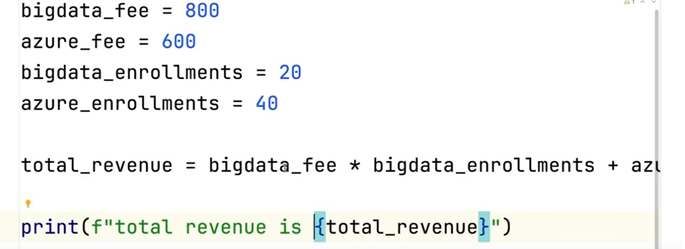
**Python by Sumit Sir**



+

\*

-

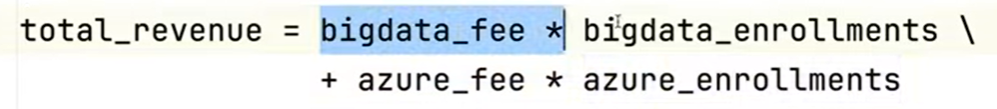
/

%

//

If we write the code and go to next line, we need to add **“\”**

**Ex:**

****

**To print in format**

**print(f“ total revenue is {variable}”)**

**Average\_price = (bigdata\_fee\*bigdata\_Enroll + azure\_fee\*azure\_enroll)/(bigdata\_enroll+azure\_enroll)**

**print(5+3\*8) 🡪 29**

**print( (5+3)\*8) 🡪 64**

**print(15/4) 🡪 3.75**

**print(15//4) 🡪 3 (Integer value only i.e the value before decimal place)**

**print(15%4) 🡪 3 (remainder)**

**Power**

**print( 2\*\*3) 🡪 8**

**Right side binding**

**print( 2\*\*2\*\*3) 🡪**

**(2\*\*3) 🡪 8**

**(2\*\*8) 🡪 256**

****

**2 is an operand and \*\* is the operator**

**Increment**

**fee = fee + 50**

**fee += 50**

**Multiplication**

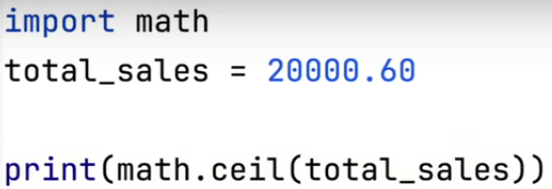
**fee \*= 2**

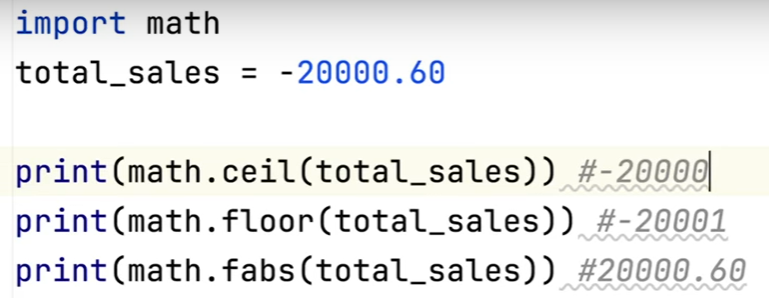
**To use round off/ round off to the value by applying the rule of 5 or greater number after decimal**

**fee = 200.60**

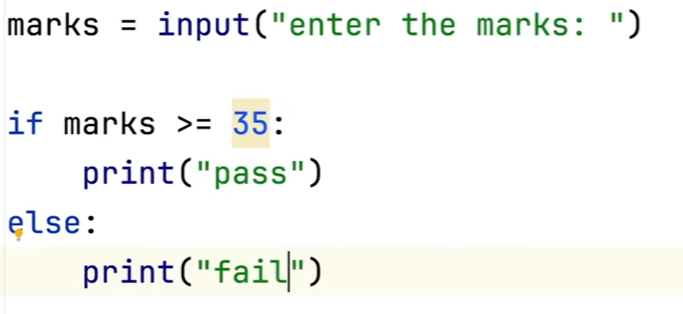
**print( math.ceil(fee)) 🡪 201**

**Print(math.floor(fee)) 🡪 200**

****

****

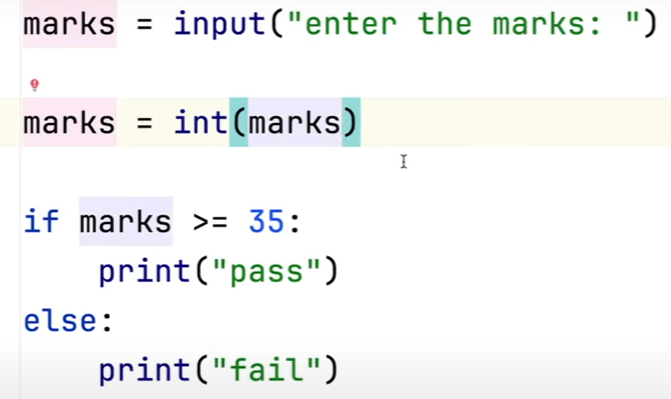
**Conditional statements**

****

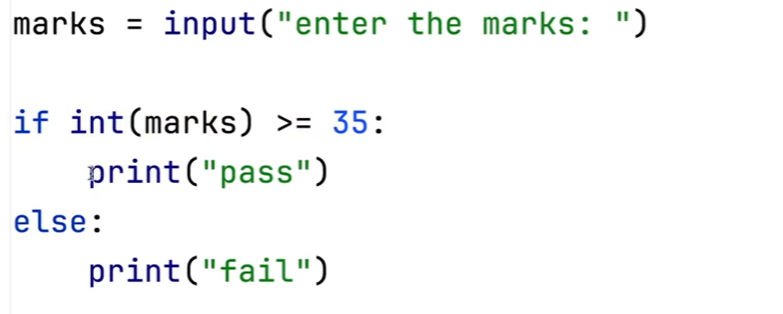
**Error**

**We need to convert marks into int**

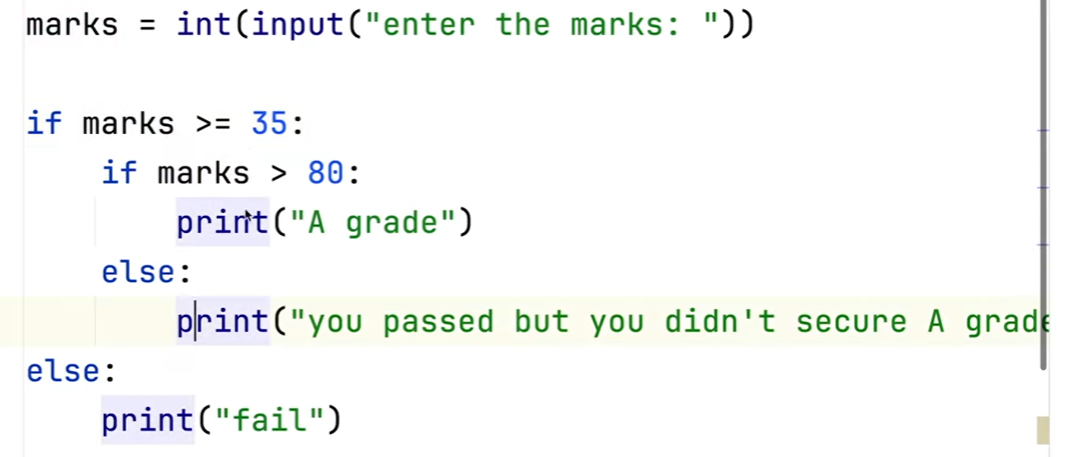
**marks = int(marks)**

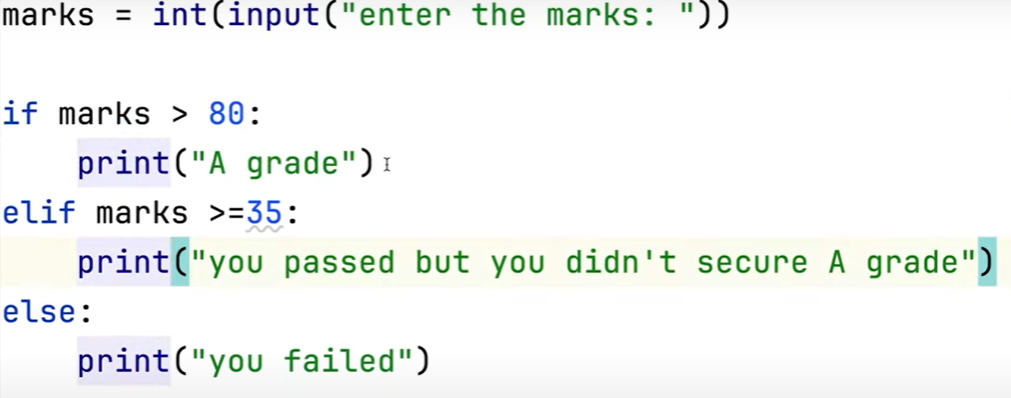
****

****

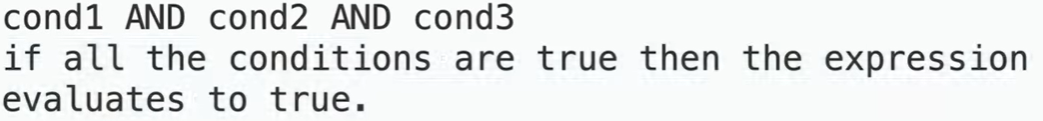
****

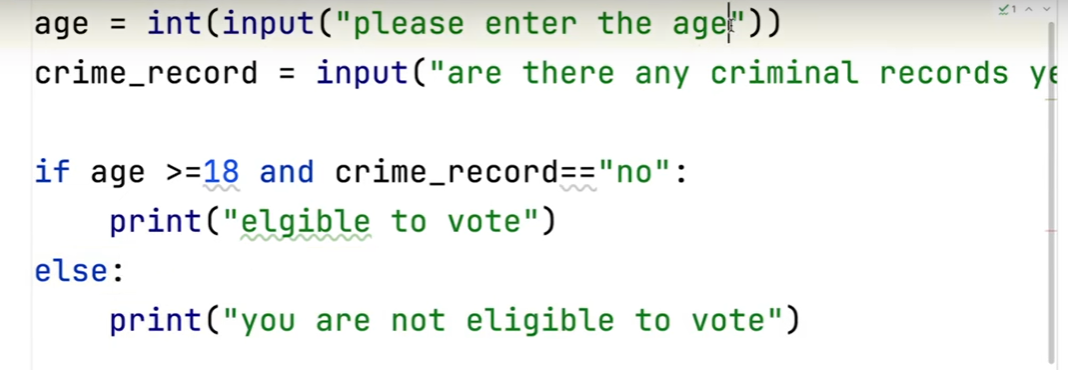
**Nested if statements**

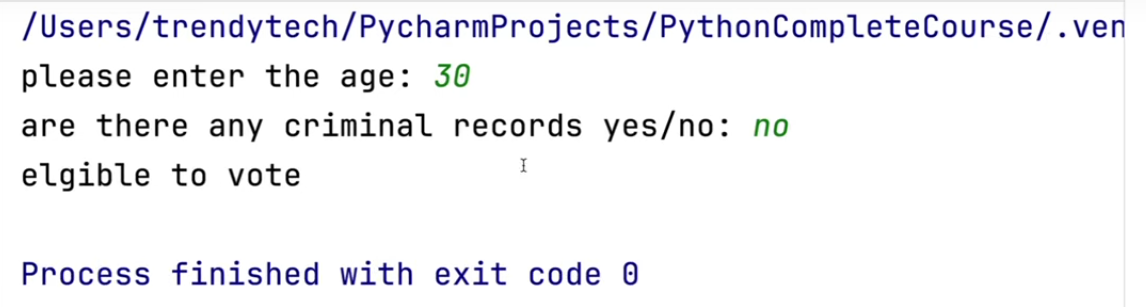
****

****

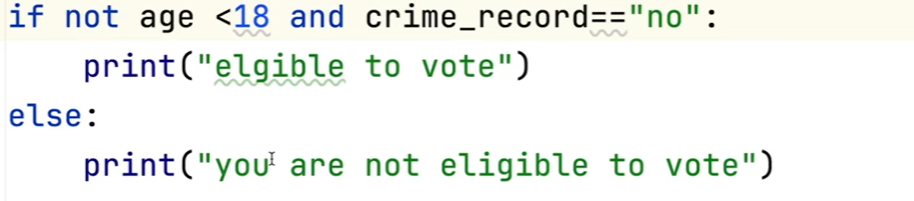
**Print(True and False) 🡪 False**

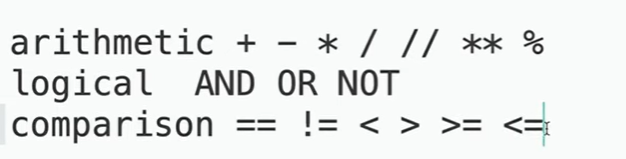
****

****

****

**Not operator**

****

****

**If any string is present in the variable, use “in” keyword**

name = "Sumit Mittal"  
print("Sumit" in name) *#True*print("sumit" in name) *#False*

**Working with Strings**

‘ ’

“ ”

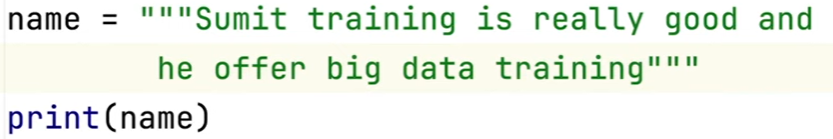
‘’’ ’’’

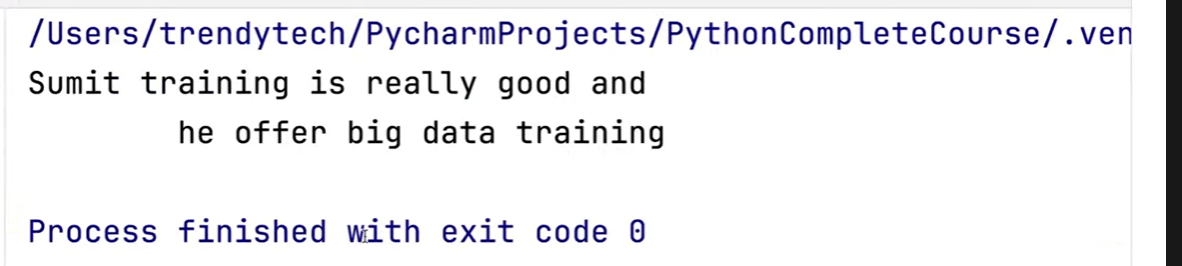
Sentence1 = “Jack’s batting is very good”

Sentence2 = ‘Jacks” batting is very good’

**Jack\’s”**  🡪 \ is an escape character

If we have string in multiple lines then we need to use “”” “””





**\n ->** escape character for new line

Sentense3 = “Jack’s batting is awesome \n as he scored some valuable runs”

\

\n 🡪 new line

\t 🡪 tab

fname = input("Enter first name ")  
lname = input("Enter last name ")  
print(fname+" "+lname)

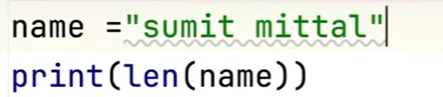
O/p:

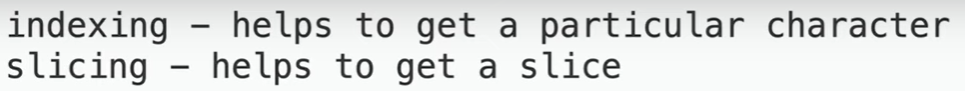
Enter first name Jack

Enter last name Will

Jack Will

To find the length of a string 🡪 len()





Index starts from 0 from left to right.

If we start indexing from right to left then it will start from -1.

Indexing

* Left to right 🡪 0
* Right to left 🡪 -1

String is immutable. We cannot modify the string.

**Slicing**

string1 = "test match"  
print(string1[0:2]) *#te*print(string1[5:10]) *#match*

print(string1[5:len(string1)]) *#match*

print(string1[5:]) *#match*

print(string1[:4]) *#test*

To print last 3 characters

print(string1[-3:len(string1)]) *#tch*

print(string1[-5:-2]) *#mat*

To print everything after the space

index = string1.find(" ")  
print(string1[index+1:len(string1)]) *#match*

Print the first word (it will be before first space)

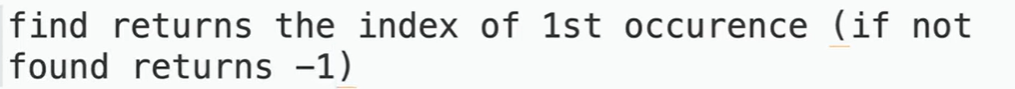
index = string1.find(" ")  
print(string1[0:index]) *#test*

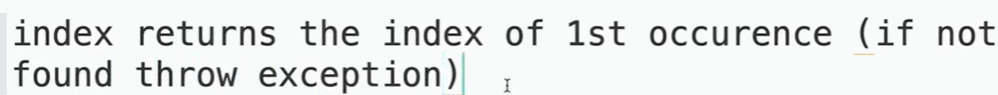
Print the alternate character

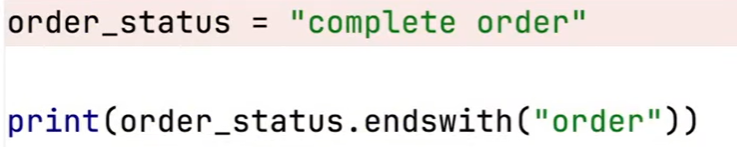
string2 = "abcdefghijk"  
print(string2) *#abcdefghijk*print(string2[1:4]) *#bcd*print(string2[1:9:2]) *#bdfh*print(string2[1::2]) *#bdfhj*

Print a string in reverse order

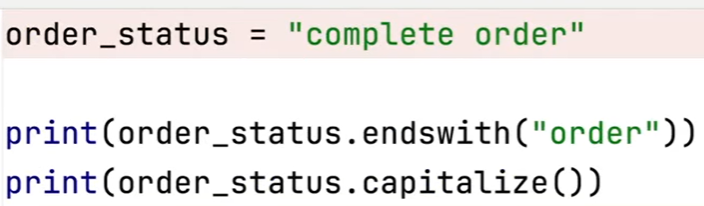
string3 = "Will jacks"  
print(string3[::-1]) *#skcaj lliW*

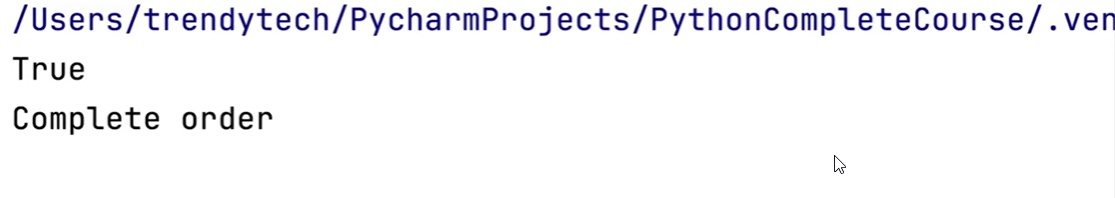






True





Strip() 🡪 it will remove all the left side and right side spaces of the string

lStrip() 🡪 it will remove all the left side spaces of the string

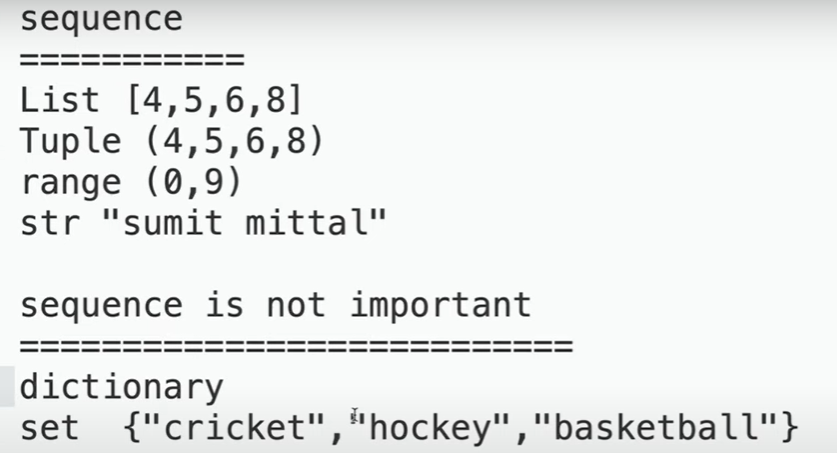
rStrip() 🡪 it will remove all the left side spaces of the string

replace() 🡪 will replace a substring from the string

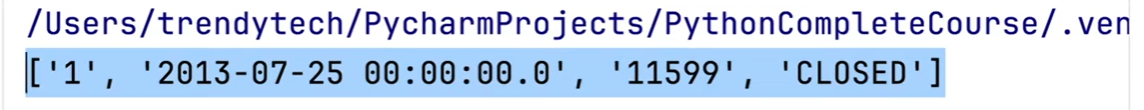


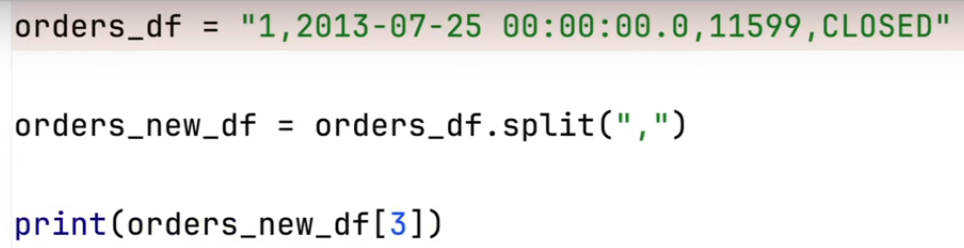
Data strutures

* List
* Tuple
* range
* str
* dictionary
* set









* CLOSED

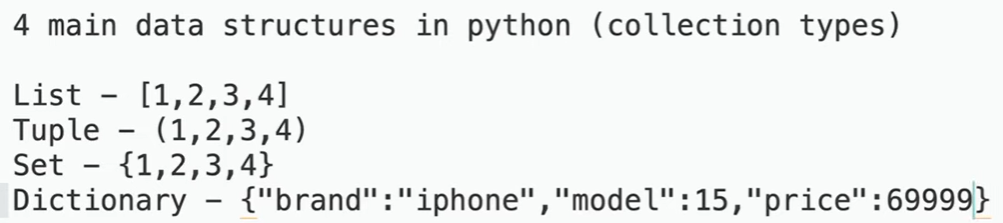
If the string is having extra spaces added like this

“\*\*\*\*\*, CLOSED “

We can use strip() function

Print(orders\_new\_df[3].strip()) 🡪 CLOSED

**List and Tuples**



For the Set and Dictionary, the sequence doesn’t matter

Sequence type – List, Tuple, String

Mutable vs Immutable

Mutable – value can be changed

Immutable – value cannot be changed

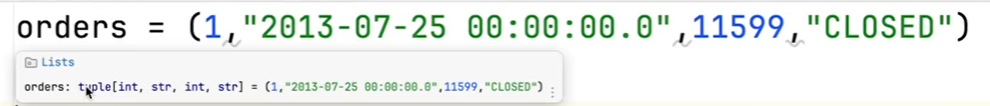
String - Immutable

Tuple - Immutable

Set – Mutable

Dictionary – Mutable

orders = [1, "2024-02-02",112233 ,"CLOSED"]  
print(type(orders))  
*# O/p - <class 'list'>*



orders = (1, "2024-02-02",112233 ,"CLOSED")  
print(type(orders))  
*# O/p - <class 'tuple'>*

List is mutable and Tuple is immutable

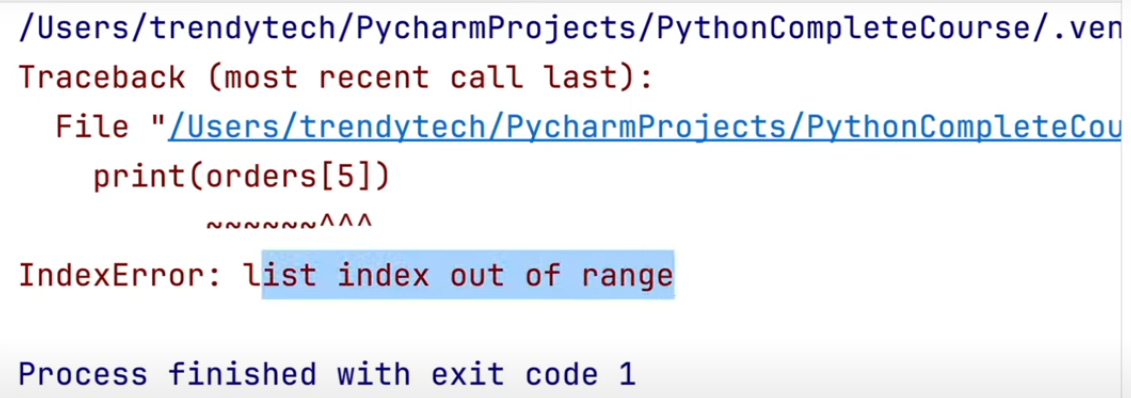
**To get element at 2nd index**

orders = [1, "2024-02-02",112233 ,"CLOSED"]

print(orders[2]) # 112233

**Element at 5th index**

orders = [1, "2024-02-02",112233 ,"CLOSED"]  
print(orders[5])

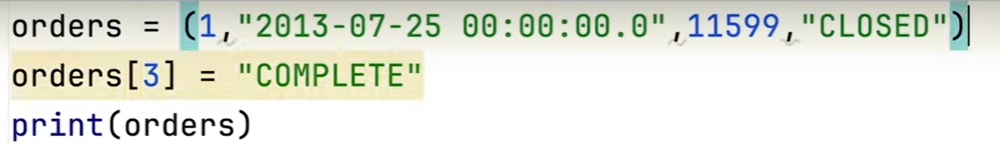


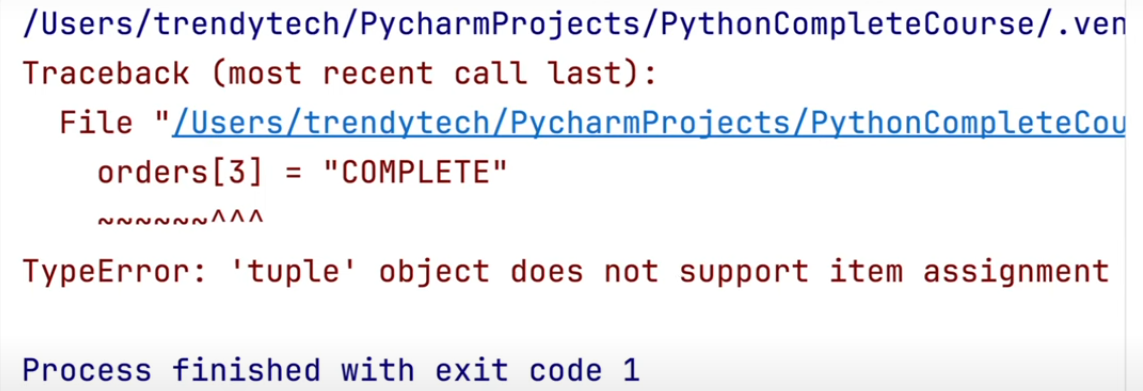
**Modify the string at 3rd index for List**

orders = [1, "2024-02-02",112233 ,"CLOSED"]  
orders[3] = 'COMPLETE'  
print(orders)

*#[1, '2024-02-02', 112233, 'COMPLETE']*

Modify the string at 3rd index for tuple

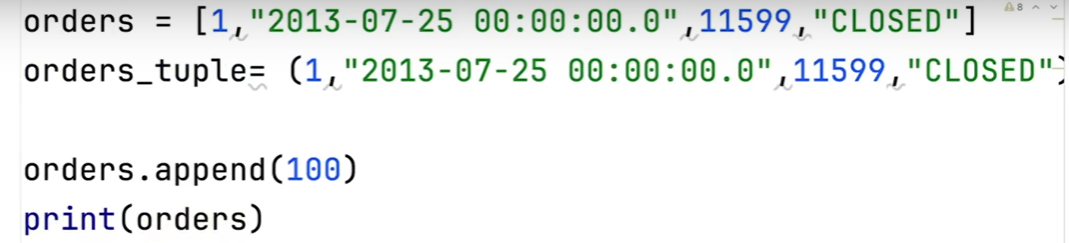


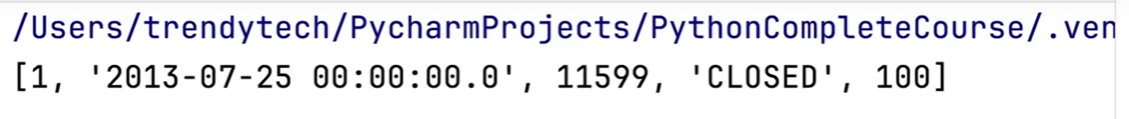


**For the list, to add element at the end 🡪 append()**

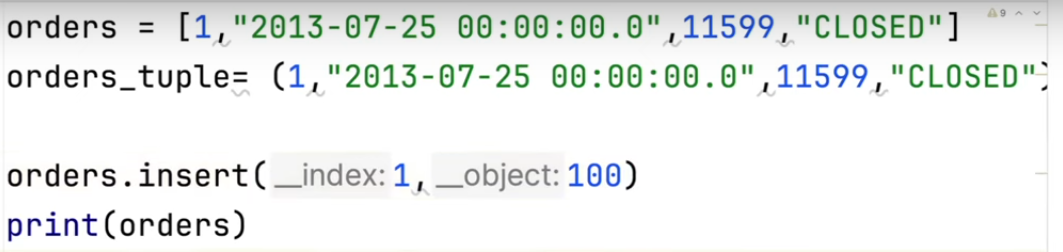
orders = [1, "2024-02-02",112233 ,"CLOSED"]  
orders.append(100)  
print(orders)  
*#[1, '2024-02-02', 112233, 'CLOSED', 100]*

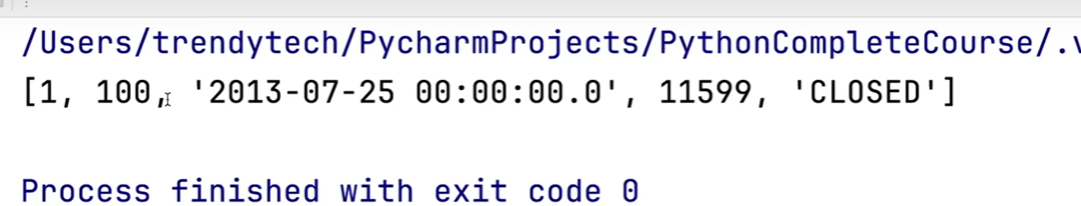
Append is applicable only for the List. For Tuple it won’t applicable.





To add element at particular index, **insert()** method is used.

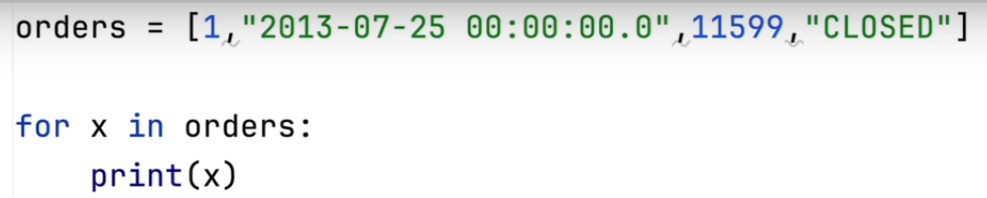




To get total number of elements present in the List 🡪 **len()**

orders = [1, "2024-02-02",112233 ,"CLOSED"]  
orders.append(100)  
print(orders)  
*#[1, '2024-02-02', 112233, 'CLOSED', 100]*print(len(orders)) *#5*

To print each element in the list



**To get type of element**

for x in orders

print(type(x))

orders = [1, "2024-02-02",112233 ,"CLOSED"]  
*#orders.pop();  
#print(orders) #[1, '2024-02-02', 112233]*orders = [1, "2024-02-02",112233 ,"CLOSED"]  
'''for x in orders:  
 print(x)  
 1  
2024-02-02  
112233  
CLOSED'''  
  
for x in orders:  
 print(type(x))  
'''<class 'int'>  
<class 'str'>  
<class 'int'>  
<class 'str'>'''

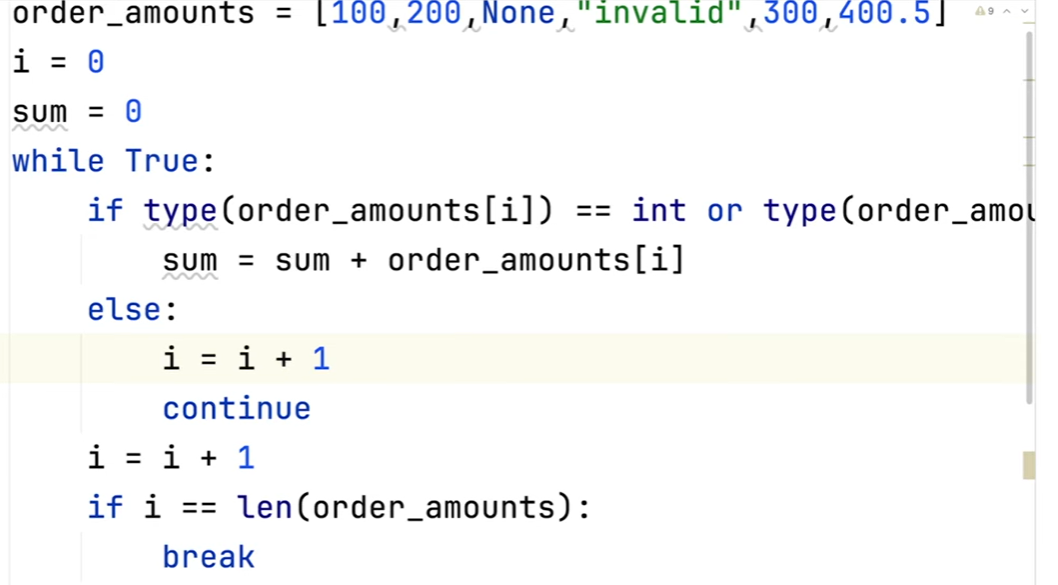
numbers = range(1,11)  
for x in numbers:  
 print(x)  
'''1  
2  
3  
4  
5  
6  
7  
8  
9  
10'''

*#To find addition of first 10 natural numbers*numbers = range(1,11)  
sum=0  
for x in numbers:  
 sum= sum+x  
print(sum) *#55*

*# Take inputs from user and find sum of first 10 natural numbers*starting\_range = int(input("Starting range: "))  
ending\_range = int(input("ending\_range: "))  
numbers = range(starting\_range, ending\_range+1)  
sum=0  
for x in numbers:  
 sum = sum+x  
print(sum) #55

*#Do the addition for the values having type as int or float*data = [100,200,None,"Jack",300,400.5]  
sum=0  
for x in data:  
 if type(x)==int or type(x)==float:  
 sum = sum+x  
 else:  
 continue  
print(sum) *#1000.5*

data = [100,200,None,"Jack",300,400.5]  
i= 0  
sum = 0  
while i < len(data):  
 if type(data[i]) == int or type(data[i])==float:  
 sum = sum+data[i]  
 else:  
 i=i+1  
 continue  
 i=i+1  
print(sum) *#1000.5*



data = [100,200,None,"Jack",300,400.5]  
print(data.index(200)) *#1*

'''data = [100,200,None,"Jack",300,400.5]  
#print(data.index(200)) #1  
print(200 in data) #True  
print(201 in data) #False'''  
  
data = [50,50,40,50,70]  
print(data.count(50)) *#3*print(data.count(10)) *#0*

data = [50,50,40,50,70]  
data.sort()  
print(data) *#[40, 50, 50, 50, 70]*

data.reverse()  
print(data) *#[70, 50, 50, 50, 40]*

orders = [10,20,30,40,50]  
orders\_new = orders  
print(orders\_new) *#[10, 20, 30, 40, 50]*

orders = [10,20,30,40,50]  
orders\_new = orders.copy()  
orders\_new[1] = 200  
print(orders) *#[10,20,30,40,50]*

*#To get the unique values from the list*orders = [10,20,10,30,30,20,40,50]  
  
unique\_orders=[]  
for x in orders:  
 if x in unique\_orders:  
 continue  
 else:  
 unique\_orders.append(x)  
  
print(unique\_orders) #[10, 20, 30, 40, 50]

*#It can be done with the help of set. As set consist only unique values*orders = [10,20,10,30,30,20,40,50]  
unique\_orders = set(orders)  
print(unique\_orders) *#{40, 10, 50, 20, 30}*

orders\_list = [

[101, '2024-03-01', 112233, 'CLOSED'],

[102, '2024-03-02', 112234, 'PENDING\_PAYMENT'],

[103, '2024-03-03', 112235, 'COMPLETED']

]

# List Comprehension

completed\_orders = [order for order in orders\_list if order[3] == 'COMPLETED']

#print(completed\_orders) #[[103, '2024-03-03', 112235, 'COMPLETED']]

#Unpacking

orders = [101, '2024-03-01', 112233, 'CLOSED']

order\_id, order\_date, customer\_id, order\_status = orders

# This is equivalent to

# order\_id = orders[1], order\_date = order[2], customer\_id=order[3], order\_status=order[4]

#print(order\_status) #CLOSED

#Generate a list of 100 numbers

list\_numbers = list(range(1,101))

#print(list\_numbers) #[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

customer\_data=[1, "Richard Brook", 9456789000, "NYC"]

#print(customer\_data[0:2]) #[1, 'Richard Brook']

#We want to remove columns 3 and 4

customer\_data=[1, "Richard Brook", "XXXXX", "XXXXX", 9456789000, "NYC"]

#print(customer\_data[0:2]+customer\_data[4:]) #[1, 'Richard Brook', 9456789000, 'NYC']

#Append 2 different lists

##Note: We can modify List. We can not edit tuple.

orders = [101, '2024-03-01', 112233, 'CLOSED']

customer\_data=[1, "Richard, Brook", "XXXXX", "XXXXX", 9456789000, "NYC"]

#orders.append(customer\_data)

#print(orders) #[101, '2024-03-01', 112233, 'CLOSED', [1, 'Richard Brook', 'XXXXX', 'XXXXX', 9456789000, 'NYC']]

#print(orders[4]) #[1, "Richard, Brook", "XXXXX", "XXXXX", 9456789000, "NYC"]

#print(orders[4][1]) #Richard, Brook

# To get the flattened list - extend

orders.extend(customer\_data)

print(orders) #[101, '2024-03-01', 112233, 'CLOSED', 1, 'Richard, Brook', 'XXXXX', 'XXXXX', 9456789000, 'NYC']