```
print("Python Programmming Fundamentals")
```

```
Python Programmming Fundamentals
```

```
print("Hello", "Class")
```

```
Hello Class
```

```
print(100)
```

```
100
```

```
print(100,200,300,400)
```

```
100 200 300 400
```

```
print(100, "i am text")
```

```
100 i am text
```

```
print(100,200,300,400, sep="####")
```

```
100####200####300####400
```

Data

- Numbers
- Text

Data Types

```
- Numbers
    - Whole    : itnegers  >>> int
    - Decimal  : flaots    >>> float

- Text
     - "cahracter/word/sentence/paragraph":  strings : str
- Boolean     : true /false : bool (True /False)
```

```
print("Hello we are  learning Python")
```

```
Hello we are  learning Python
```

```
print("100")
```

```
100
```

```
print(23.6)
```

```
23.6
```

```
print(200)
```

```
200
```

```
print(type(200))
```

```
<class 'int'>
```

```
print(type(23.6))
```

```
<class 'float'>
```

```
print(type("Hello we are  learning Python"))
```

```
<class 'str'>
```

```
print(type(True))
```

```
<class 'bool'>
```

Variables

```
course  = "Data  Sciences and AI"
```

```
course
```

```
'Data  Sciences and AI'
```

```
course = "Python for Ai and Data Sciences"
```

```
course
```

```
'Python for Ai and Data Sciences'
```

```
marks  = 100
```

```
type(marks)
```

```
int
```

Variable Naming

```
# variable name shpould not start with a number
1student = "Ali"
```

```
  Cell In[23], line 2
    1student = "Ali"
    ^
SyntaxError: invalid decimal literal
```

```
# no special charaters are allowed.
father@name = "asad"
father-name = "asad"
father.name = "asad"
father$name = "asad"
```

```
  Cell In[24], line 2
    father@name = "asad"
    ^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
```

```
#space not allowed in variable name
father name = "asad"
student name = "asad"

```

```
  Cell In[25], line 2
    father name = "asad"
           ^
SyntaxError: invalid syntax
```

Reserved words: no resrved words can be uaed as a variable name

image.png

Operator

```
- - (subtraction)
- + (addition)
- * (
- /
- //
- %
- =
- ** exponent
```

```
4+6
```

```
10
```

```
4 - 9
```

```
-5
```

```
5*5
```

```
25
```

```
5**3
```

```
125
```

```
# floating divsion
10 / 3
```

```
3.3333333333333335
```

```
10 // 3 # floor disivon integer
```

```
3
```

```
10.2//4
```

```
2.0
```

% modulus operator

```
    - remainder
```

```
12%6
```

```
0
```

Assignment Operator

```
total = 120

# a integer value 120 is asssigned to a variable named total
```

Comparision Operator

- ==

- !=

- >

- <

- >=

- <=

```
marks = 100
obtained = 67
```

```
marks > obtained
```

```
True
```

```
marks < obtained
```

```
False
```

```
marks >= obtained
```

```
True
```

```
marks<= obtained
```

```
False
```

```
marks == obtained
```

```
False
```

```
marks!=obtained
```

```
True
```

User Input

```
input("please tell me your age")
```

```
please tell me your age12
'12'
```

```
age  = input("please tell me your age")
```

```
please tell me your age120
```

```
type(age)
```

```
str
```

```
age + 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 1
----> 1 age + 10

TypeError: can only concatenate str (not "int") to str
```

Type Casting

```
int(133.5)
```

```
133
```

```
float(12)
```

```
12.0
```

```
int("100")
```

```
100
```

```
str(100)
```

```
'100'
```

```
float("45.6")
```

```
45.6
```

```
int(float("45.6"))
```

```
45
```

```
age  = int(input("Enter your age"))
age+10
```

```
Enter your age45
55
```

```
age = age+ 10
age
```

```
75
```

Incrment Operator / Decrement Operator

```
count  = 1
```

```
count = count+1
count
```

```
5
```

```
count +=5
count
```

```
30
```

```
count -=5
count
```

```
0
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
name  = "Imran Khan"
dob = 1952
citizenship = "Paksitani"
profession1 = "Cricketer"
profession2 = "Politician"
age  = 2024- dob
```

Double-click (or enter) to edit

String: Data in inverted comas

String is class

String is immutable (non changable)

```
course  = "artificial intelligence"
```

```
print(type(course))
```

```
<class 'str'>
```

```
course.capitalize()
```

```
'Artificial intelligence'
```

```
course.upper()
```

```
'ARTIFICIAL INTELLIGENCE'
```

```
course.lower()
```

```
'artificial intelligence'
```

```
course.title()
```

```
'Artificial Intelligence'
```

```
course.count("E")
```

```
0
```

```
text = "It’s typically more aggressive, and because basal-like cancers lack targeted treatments, it’s a promising focus for virtual scre
```

```
text.endswith("identification.")
```

```
True
```

```
text.endswith(".")
```

```
True
```

```
text.endswith("ion.")
```

```
True
```

```
text.endswith("Identification.")
```

```
False
```

```
course.startswith("ar")
```

```
True
```

```
marks = "100"
```

```
marks.isalnum()
```

```
True
```

```
marks.isalpha()
```

```
False
```

```
marks.islower()
```

```
False
```

String Concatenation: joining

```
+  operator is concatenation operator
```

```
first_name = "Ali"
last_name = "Hassan"
```

```
first_name + last_name
```

```
'AliHassan'
```

```
age = 30
```

```
first_name+last_name+age
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 first_name+last_name+age

TypeError: can only concatenate str (not "int") to str
```

```
first_name+last_name+str(age)
```

```
'AliHassan30'
```

```
name = input("Please tell me your good name: ")
fname = input("Dont mind abbu ka nam bataen:  ")
age = float(input("How old are you"))

print("Mr."+name+ " s/o "+ fname + " you are "+ str(age) + " old.")
```

```
Please tell me your good name: Rashid
Dont mind abbu ka nam bataen:  Amin
How old are you20
Mr.Rashid s/o Amin you are 20.0 old.
```

string formatting

```
print("Mr.{} S/O {} you are {} old.".format(name,fname,age))
# placeholders
```

```
Mr. Rashid S/O Amin you are 20.0 old.
```

```
print(f"Mr. {name} S/O {fname} you are {age} old.")
```

```
Mr. Rashid S/O Amin you are 20.0 old.
```

String indexing and Slicing

```
text = "it’s a promising focus for virtual screening and target identification." # single line double quotes
```

```
text = 'it’s a promising focus for virtual screening and target identification.' # single line single quotes
```

```
text = """it’s a promising focus for virtual
           screening and target
           identification.""" # multiple line tripple quotes
```

```
text = '''it’s a promising focus for virtual
           screening and target
           identification.''' # multiple line tripple quotes
```

length >>> len()

```
len(text)
```

```
8
```

```
text = "Paksitan"
#       01234567
#-8-7-6-5-4-3-2-1
```

```
# [] sqaure brackets are always used for indexing
text[2]
```

```
'k'
```

```
text[-3]
```

```
't'
```

Slicing: sub string

```
text = "it’s a promising focus for virtual" # sub string
```

```
text[0:11]  # [start : end: step]
```

```
'it’s a prom'
```

```
text[5:13]
```

```
'a promis'
```

```
country = "pakistan"
```

```
country[0:6:1]
```

```
'pakist'
```

```
country[0:7:2]
```

```
'pksa'
```

Python Collections/DataStrcutures

- String
- List
- Tuple
- Dictionary
- Set

LIST

```
- []
- many values
- many type values
- mutable
- index
- slice
- copy
```

```
student = "Ahmed"
```

```
#            0        1       2        3        4        5      6     7
students = ['Ali', 'Basit','Kaleem', 'Nasir', 'Babar','Asif','Asad','Saad']
#            -8      -7        -6       -5      -4     -3       -2    -1
```

```
print(type(students))
```

```
<class 'list'>
```

```
students[3]
```

```
'Nasir'
```

```
students[-2]
```

```
'Asad'
```

```
students[1], students[-7]
```

```
('Basit', 'Basit')
```

```
Start coding or generate with AI.
```

```
import pandas as pd
result = pd.read_excel("result.xlsx")
```

```
result = result[["Name","Email Address", "Score"]]
```

```
Start coding or generate with AI.
```

	Name	Email Address	Score
0	NaN	NaN	34
1	Muhammad Zeeshan	zeeshanayaz1@gmail.com	36
2	hamza parekh	hamzakashifparekh2009@gmail.com	27
3	Muhammad Ahmed Muazan	muazanqureshi3@gmail.com	30
4	Sufyan Abdul Rahman	sufyan.abdulrahman55@gmail.com	35
...	...	...	...
129	Faiza noor	balochfaizan8989@gmail.com	29
130	Abdul Rehman	tygif397@gmail.com	36
131	Muhammad Sharjeel Mughal	mastermughal6969@gmail.com	33
132	Muhammad Hamza	hamzaazam2076@gmail.com	32
133	Hanzala jahangir	lyneskroff2189@gmail.com	26

134 rows × 3 columns

```
result.to_excel("result.xlsx")
```

List functions/Methods

```
students = ['Ali', 'Basit','Kaleem', 'Nasir', 'Babar','Asif','Asad','Saad']
```

Adding more memebers to an existing list

- append(): appends a member at very last
- insert(): insert a member at given index
- extend(): adds a collection to list
-
  - : concats to list

```
# adds a single value or member in last of the list
students.append("Rashid")
```

```
students
```

```
['Ali', 'Basit', 'Kaleem', 'Nasir', 'Babar', 'Asif', 'Asad', 'Saad', 'Rashid']
```

```
students.append("Rashid", "Zahid")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 students.append("Rashid", "Zahid")

TypeError: list.append() takes exactly one argument (2 given)
```

```
# add Majid on 3rd index
students.insert(3,"Majid")
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid']
```

```
# add Majid on 100 index
students.insert(100,"Shani")
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani']
```

```
staff = ["wahid", 'kamil', "hammad", "munib"]
```

```
students.append(staff)
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib']]
```

```
students.extend(staff)
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib']
```

```
students.extend("Hello")
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o']
```

```
students.extend("100")
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0']
```

```
students.extend("1")
```

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0',
 '1']
```

```
students + staff
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0',
 '1',
 'wahid',
 'kamil',
 'hammad',
 'munib']
```

```
staff
```

```
['wahid', 'kamil', 'hammad', 'munib']
```

Deleting a member from existin list

- del statement deletes parmanently via index
- remove(): deletes parmanently via value
- pop(): deletes parmanently from last index

  ```
  can delete a member from given index

  it returns deleted
  ```

  - clear()

```
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
 ['wahid', 'kamil', 'hammad', 'munib'],
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0',
 '1']
```

```
del students[12]
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Rashid',
 'Shani',
```

```
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0',
 '1']
```

```
del students[10]
students
```

```
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Shani',
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l',
 'o',
 '1',
 '0',
 '0',
 '1']
```

```
students.remove('o')
```

```
students.remove('0')
```

```
students.remove('hello')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[39], line 1
----> 1 students.remove('hello')

ValueError: list.remove(x): x not in list
```

```
print(students)
students.pop()
students
```

```
['Ali', 'Basit', 'Kaleem', 'Sajid', 'Majid', 'Nasir', 'Babar', 'Asif', 'Asad', 'Saad', 'Shani', 'wahid', 'kamil', 'hammad', 'munib',
['Ali',
 'Basit',
 'Kaleem',
 'Sajid',
 'Majid',
 'Nasir',
 'Babar',
 'Asif',
 'Asad',
 'Saad',
 'Shani',
 'wahid',
 'kamil',
 'hammad',
 'munib',
 'H',
 'e',
 'l',
 'l']
```

```
students = ['Ali', 'Basit', 'Kaleem', 'Sajid', 'Majid', 'Nasir', 'Babar', 'Asif', 'Asad']
```

```
deleted_values =  students.pop()
```

```
deleted_values
```

```
'Asif'
```

```
deleted_values =  students.pop(5)
```

```
deleted_values
```

```
'Nasir'
```

```
students
```

```
['Ali', 'Basit', 'Kaleem', 'Sajid', 'Majid', 'Babar']
```

```
del students[3:6]
students
```

```
['Ali', 'Basit', 'Kaleem']
```

```
students.clear()
```

```
students
```

```
[]
```

```
students.index("hamid")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[52], line 1
----> 1 students.index("hamid")

ValueError: 'hamid' is not in list
```

```
students.extend('2ab5h')
students
```

```
['2', 'a', 'b', '5', 'h']
```

```
students.sort()
```

```
students
```

```
['2', '5', 'a', 'b', 'h']
```

```
chr(99)
```

```
'c'
```

```
ord('c')
```

```
99
```

```
students.append(100)
```

```
students.insert(3,200)
```

```
students
```

```
['2', '5', 'a', 200, 'b', 'h', 100]
```

```
students.sort()
```

```
---------------------------------------------------------------------------
```

```
students.reverse()
```

```
----> 1 students.sort(
```

```
students
```

```
[100, 'h', 'b', 200, 'a', '5', '2']
```

```
new_students = students.copy()
```

```
new_students
```

```
[100, 'h', 'b', 200, 'a', '5', '2']
```

```
old_students = students
```

```
old_students
```

```
[100, 'h', 'b', 200, 'a', '5', '2']
```

Pass by Value and Pass by reference (deep copy and shaLLOW COPY)

```
students
```

```
[100, 'h', 'b', 200, 'a', '5', '2']
```

```
del students[3]
students
```

```
[100, 'h', 'b', 'a', '5', '2']
```

```
old_students
```

```
[100, 'h', 'b', 'a', '5', '2']
```

```
id(old_students)
```

```
2554048404416
```

```
id(students)
```

```
2554048404416
```

```
new_students
```

```
[100, 'h', 'b', 200, 'a', '5', '2']
```

```
Start coding or generate with AI.
```

List in List

```
numbers  = [2,3,4,5, [11,12,13,14], 7,8]
```

```
numbers
```

```
[2, 3, 4, 5, [11, 12, 13, 14], 7, 8]
```

```
len(numbers)
```

```
7
```

```
numbers[4]
```

```
[11, 12, 13, 14]
```

```
numbers[4][2]
```

```
13
```

```
numbers.append("Pakistan")
```

```
numbers
```

```
[2, 3, 4, 5, [11, 12, 13, 14], 7, 8, 'Pakistan']
```

```
numbers[-1][-1]
```

```
'n'
```

```
numbers[-1][-3]
```

```
't'
```

```
numbers[-1][2:6]
```

```
'kist'
```

```
# repalce the value at index -2 with new value 800
numbers[-2] = 800
numbers
```

```
[2, 3, 4, 5, [11, 12, 13, 14], 7, 800, 'Pakistan']
```

```
numbers[-1][6:2:-1]
```

```
'atsi'
```

Dictionary

- {key:value}
- Mutable
- Iterable
- More descriptive than a list

```
profile = {}
```

```
type(profile)
```

```
dict
```

```
len(profile)
```

```
0
```

```
profile = {'name':'Ali', "name":"Papoo", 'age':28, 'height':5.8, 'weight':60,'qualification':'Graduate'}
```

Note: Keys can be string, int, tuple

```
Values: int, float, string, boolean, list, dictionary, tuple, set
```

```
len(profile)
```

```
5
```

Accessing Member in a Dictionary

```
profile['name']
```

```
'Ali'
```

```
profile['age']
```

```
28
```

```
profile['height']
```

```
5.8
```

Adding member in existing dictionary

```
# if key exists, it will replace the value with new value
# if key doesnot exist , new key:value pair will be created
profile['email'] = 'abc@gmail.com'
profile
```

```
{'name': 'Papoo',
 'age': 28,
 'height': 5.8,
 'weight': 60,
 'qualification': 'Graduate',
 'email': 'abc@gmail.com'}
```

```
profile.keys()
```

```
dict_keys(['name', 'age', 'height', 'weight', 'qualification', 'email'])
```

```
profile.values()
```

```
dict_values(['Papoo', 28, 5.8, 60, 'Graduate', 'abc@gmail.com'])
```

```
profile.items()
```

```
dict_items([('name', 'Papoo'), ('age', 28), ('height', 5.8), ('weight', 60), ('qualification', 'Graduate'), ('email',
'abc@gmail.com')])
```

```
profile.pop()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[75], line 1
----> 1 profile.pop()

TypeError: pop expected at least 1 argument, got 0
```

```
profile.pop('weight')
```

```
60
```

```
profile
```

```
{'name': 'Papoo',
 'age': 28,
 'height': 5.8,
 'qualification': 'Graduate',
 'email': 'abc@gmail.com'}
```

```
profile.popitem()
```

```
('email', 'abc@gmail.com')
```

```
profile
```

```
{'name': 'Papoo', 'age': 28, 'height': 5.8, 'qualification': 'Graduate'}
```

```
profile.popitem()
```

```
('qualification', 'Graduate')
```

```
reserach_areas = {'CompSc':"Ai", "Phy":"QunatumTheory"}
```

```
reserach_areas
```

```
{'CompSc': 'Ai', 'Phy': 'QunatumTheory'}
```

```
profile['research'] = reserach_areas
profile
```

```
{'name': 'Papoo',
 'age': 28,
 'height': 5.8,
 'research': {'CompSc': 'Ai', 'Phy': 'QunatumTheory'}}
```

```
profile.update(reserach_areas)
```

```
profile
```

```
{'name': 'Papoo',
 'age': 28,
 'height': 5.8,
 'research': {'CompSc': 'Ai', 'Phy': 'QunatumTheory'},
 'CompSc': 'Ai',
 'Phy': 'QunatumTheory'}
```

```
del profile['research']
```

```
profile
```

```
{'name': 'Papoo',
 'age': 28,
 'height': 5.8,
 'CompSc': 'Ai',
 'Phy': 'QunatumTheory'}
```

List in a Dictionary

Dictionary in a list

Dictioary in a Dictioary

List in a dictionary

```
report  = {'name':['asad','ali','danish','anas','riaz','nasir'],
           'roll':[1,2,3,4,5,6],
           'course':['python', 'stats','ai', 'python','ML'],
           'marks':[90,67,78,90,65,87]}
```

```
report
```

```
{'name': ['asad', 'ali', 'danish', 'anas', 'riaz', 'nasir'],
 'roll': [1, 2, 3, 4, 5, 6],
 'course': ['python', 'stats', 'ai', 'python', 'ML'],
 'marks': [90, 67, 78, 90, 65, 87]}
```

```
report['course'][2]
```

```
'ai'
```

```
report['name'][1]
```

```
'ali'
```

Dictionary in List

```
students = [ {'name':'Asad', 'roll':123, 'course':'Python'},
             {'name':'Danial', 'roll':124, 'course':'Ai'},
             {'name':'Yasir', 'roll':100, 'course':'Python'},
             {'name':'Umer', 'roll':113, 'course':'ML'}  ]
```

```
len(students)
```

```
4
```

```
len(students[1])
```

```
3
```

```
students[3]['name']
```

```
'Umer'
```

```
students[3]['course']
```

```
'ML'
```

Dictionary in Dictionary

```
student = {
             "names":{"first_name":"Nasir", "last_name":"Hussain"}  ,
             "course":{"enrolled":"Python", 'dropped':'Ai'} ,
             "emails":{"current":"abc@gmail.com", "previous":"xyz@gmail.com"}
           }
```

```
len(student)
```

```
3
```

```
student['course']['dropped']
```

```
'Ai'
```

Operators :

in: membership checking operator

is: identity check operator

not: negation

and: logical and

or: logical

```
name=['asad','ali','danish','anas','riaz','nasir']
```

```
name
```

```
['asad', 'ali', 'danish', 'anas', 'riaz', 'nasir']
```

in opaerator:

```
checks if a member is in collection or not.
```

```
"Anas" in name
```
```
False
```
```
"Anas" not in name
```
```
True
```
```
"a" in name[3]
```
```
True
```
```
'emails' in student
```
```
True
```
```
'dropped' in student
```
```
False
```
```
a=10
```
```
b= 20
```
```
a is b
```
```
False
```
```
c = a
```
```
id(a)
```
```
140722955299544
```
```
id(c)
```
```
140722955299544
```
```
a is c
```
```
True
```
```
c is a
```
```
True
```
```
d = 20
```
```
b is d
```
```
True
```
```
a = 9
c = a
```
```
a = 10
```
```
c
```
```
9
```
```
id(a)
```
```
140722955299512
```
```
id(c)
```

```
140722955299512
```

```
a=100
b=200
c=300

a<b or c<b
```

```
True
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
a<b and c<b
```

```
False
```

```
12 or 50 or 20
```

```
12
```

```
12 and 50 and 20
```

```
20
```

Truthy : every value has truthy value

```
bool("hello")
```

```
True
```

```
bool("")
```

```
False
```

```
bool(0)
```

```
False
```

```
bool(1)
```

```
True
```

```
bool(12)
```

```
True
```

```
bool([1,3])
```

```
True
```

```
bool([])
```

```
False
```

```
12 or 50 or 20
```

```
12
```

```
12 and 50 and 20
```

```
20
```

```
Start coding or generate with AI.
```

```
a =2
b= 3
c =4
```

```
x1,x2 =   (-b+(b**2-4*a*c)**(0.5) )/2*a        , (-b-(b**2-4*a*c)**(0.5) )/2*a
```

```
x1
```

```
(-2.9999999999999996+4.795831523312719j)
```

```
x2
```

```
(-3.0000000000000004-4.795831523312719j)
```

Tuple

```
- ()
- immutable (unchangable)
- tuple iterable
- index
```

```
atuple = (11,22,33,44,55,66)
```

```
print(type(atuple))
```

```
<class 'tuple'>
```

```
len(atuple)
```

```
6
```

```
atuple[3]
```

```
44
```

```
atuple[2:5]
```

```
(33, 44, 55)
```

```
atuple[0] = 1000
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 atuple[0] = 1000

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

```
del atuple[4]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 del atuple[4]

TypeError: 'tuple' object doesn't support item deletion
```

```
a = (12,)
print(type(a))
```

```
<class 'tuple'>
```

Set

- {}

```
aset = {3}
```

```
aset= {}
```

```
type(aset)
```

```
dict
```

```
bset={1,2,32,3,4,55,6,7,8,8,8,9}
```

```
len(bset)
```

```
10
```

Conditional Statements/ If Clause/ Conditionals

```
- if statement
- if statment else
- elif
- nested if's
```

```
age = int(input("Enter your age: "))
if age <18:
    print("Abhi tum chotay ho")
    print("Abhi vote nah dal sakte")
    print("Car dribe nahi karsakte")
else:
    print("Abhi tum bare hogae ho")
    print("Zara hosh karo")
    print("Career pe dhyan do")
```

```
Enter your age:  18
Abhi tum bare hogae ho
Zara hosh karo
Career pe dhyan do
```

```
numbers = int(input("Enter your numbers in Python test"))
if numbers >=25:
    print("Pass")
    print("Next test ki tayari karen")
else:
    print("Fail")
    print("Mehnat karen")
```

```
Enter your numbers in Python test 24
Fail
Mehnat karen
```

```
# take a number input and check if the number is even or odd
num = int(input("Enter a number:  "))
if num%2==0:
    print(f"{num} is a Even number")
else:
    print(f"{num} is an odd number")
```

```
Enter a number:   10
10 is a Even number
```

```
# take total shopping amount as input and if the amount is greater than 10000, calculate 15% discount
# and show the total and total after discount.
total = int(input("Enter total shopping amount: "))
if total >=10000:
    discount = total *0.15
    print(f"""
             Total Amount Before Discount {total}
             15% Discount is {discount}
             Total Payable : {total-discount}
             """)
else:
    print(f"Total amount payable is {total}")
```

```
Enter total shopping amount:  9000
Total amount payable is 9000
```

```
friends_lst = ['ali','bilal','wajahat', 'shani', 'raza', 'sohaib', 'waway', 'asim']
```

```
name  = input("Please tell me your name: ").lower()
if name in friends_lst:
    print("Your are Welcome!")
else:
    print("Please wait, your name is missing")
```

```
Please tell me your name:  wALI
Please wait, your name is missing
```

Multiple if's

```
# tamator: agr price >500, matlena, p>300, 1/2kg, 200>1kg 100>2kg 5kg
tomato_price = int(input("Tamator kia hisab hen?"))
if tomato_price >=500:
    print("Rehne do bhai")
elif tomato_price >=300:
    print("1/2 Kg deden")
elif tomato_price >=200:
    print("1 Kg deden")
elif tomato_price>=100:
    print("2kg deden")
else:
    print("5 Kg")
```

```
Tamator kia hisab hen? 90
5 Kg
```

```
tomato_price = int(input("Tamator kia hisab hen?"))
if tomato_price <100:
    print("5 kg")
elif tomato_price<=200:
    print("2 Kg")
elif tomato_price<=300:
    print("1Kg")
elif tomato_price<=400:
    print("1/2 Kg")
else:
    print("mat lena")
```

```
Tamator kia hisab hen? 450
mat lena
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
sub1 = float(input("Enter marks in subject1: "))
sub2 = float(input("Enter marks in subject2: "))
sub3 = float(input("Enter marks in subject3: "))
sub4 = float(input("Enter marks in subject4: "))
sub5 = float(input("Enter marks in subject5: "))
obtianed = sub1+sub2+sub3+sub4+sub5
total  = 500
grade = None
percentage  = obtianed/total * 100
if percentage>=90:
    grade = "A*"
elif percentage>=80:
    grade = "A"
elif percentage>=70:
    grade = "B"
elif percentage>=60:
    grade = "C"
elif percentage>=50:
    grade = "D"
else:
    grade ="Fail"

print(f"""
        Subject1 : {sub1}
        Subject2 : {sub2}
        Subject3 : {sub3}
        Subject4 : {sub4}
        Subject5 : {sub5}

        Total Marks Obtained: {obtianed}
        Percentage Obtained : {round(percentage,2)}
        Grade Achieved      : {grade}

""")
```

```
Enter marks in subject1:  0
Enter marks in subject2:  89
Enter marks in subject3:  90
Enter marks in subject4:  89
Enter marks in subject5:  90

        Subject1 : 0.0
        Subject2 : 89.0
        Subject3 : 90.0
        Subject4 : 89.0
        Subject5 : 90.0

        Total Marks Obtained: 358.0
        Percentage Obtained : 71.6
        Grade Achieved      : B
```

```
round(123.15634546565655, 3)
```

```
123.156
```

```
Start coding or generate with AI.
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
total_purchase = int(input("Enter the total purchase amt: "))
if total_purchase >100:
    discount = total_purchase*.10
    print(f"Total Purchase Amt: ${total_purchase} ")
    print(f"Discount Availed ${discount}")
    print(f"Amount After Discount is ${total_purchase-discount}")
else:
    print(f"Amount Payable is ${total_purchase}")
```

```
Enter the total purchase amt:  1200
Total Purchase Amt: $1200
Discount Availed $120.0
Amount After Discount is $1080.0
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
items = int(input("Enter the number of items purchased: "))
total_price = int(input("Enter the total price: "))
if items >5:
    discount = total_price*.15
    print(f"Total Price is {total_price-discount}")
else:
    print(f"Total Price is {total_price}")
```

```
Enter the number of items purchased:  6
Enter the total price:  100
Total Price is 85.0
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
price = 10000
is_member = input("Are you a memeber or not (Y/N): ").lower()
if is_member=='y':
    print(f"Price: {price*0.80}")
else:
    print(f"Price: {price*0.95}")
```

```
Are you a memeber or not (Y/N):  n
Price: 9500.0
```

Nested ifs'

```
class_day  = input("Do you have a class today? ")
if class_day =="yes":
    id_card = input("Do you have your ID Card? ")
    if id_card =="yes":
        attendance = input("Is your attendance is complete?")
        if attendance == "yes":
            assignment = input("Have you done the assignment? ")
            if assignment =="yes":
                print("You can attend the class")
            else:
                print("You can not attend the class assginement not done")
        else:
            print("You can not attend the class if attendance is incompete")
    else:
        print("You can not attend the class if dont have id card")
else:
    print("You can not attend the class if not a class day")
```

```
Do you have a class today?  yes
Do you have your ID Card?  yes
Is your attendance is complete? yes
Have you done the assignment?  yes
You can attend the class
```

```
sub1 = float(input("Enter marks in subject1: "))
sub2 = float(input("Enter marks in subject2: "))
sub3 = float(input("Enter marks in subject3: "))
sub4 = float(input("Enter marks in subject4: "))
sub5 = float(input("Enter marks in subject5: "))
obtianed = sub1+sub2+sub3+sub4+sub5
total  = 500
```

```
grade = None
percentage  = obtianed/total * 100
result = None
if sub1>=50:
    if sub2>=50:
        if sub3>=50:
            if sub4>=50:
                if sub5>=50:
                   result = "Pass"
                else:
                    result = "Fail"

            else:
                result = "Fail"
        else:
            result = "Fail"
    else:
        result = "Fail"

else:
    result = "Fail"

if percentage>=90:
    grade = "A*"
elif percentage>=80:
    grade = "A"
elif percentage>=70:
    grade = "B"
elif percentage>=60:
    grade = "C"
elif percentage>=50:
    grade = "D"
else:
    grade ="Fail"

print(f"""
        Subject1 : {sub1}
        Subject2 : {sub2}
        Subject3 : {sub3}
        Subject4 : {sub4}
        Subject5 : {sub5}

        Total Marks Obtained: {obtianed}
        Percentage Obtained : {round(percentage,2)}
        Grade Achieved      : {grade}
        Result:               {result}

""")
```

```
Enter marks in subject1:  78
Enter marks in subject2:  98
Enter marks in subject3:  45
Enter marks in subject4:  89
Enter marks in subject5:  56

        Subject1 : 78.0
        Subject2 : 98.0
        Subject3 : 45.0
        Subject4 : 89.0
        Subject5 : 56.0

        Total Marks Obtained: 366.0
        Percentage Obtained : 73.2
        Grade Achieved      : B
        Result:               Fail
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
lst = input("Enter memebrs of list").split(",")

print(lst[::2])
```

```
Enter memebrs of list 1,2,3,4,5,6,7,8,9,0
['1', '3', '5', '7', '9']
```

```
"helleo".split('e')
```

```
['h', 'll', 'o']
```

```
lst
```

```
['nasir', ' talham kamran', ' faisal']
```

```
",".join(lst)
```

```
'nasir, talham kamran, faisal'
```

```
"*".join(["1","2","3","4","5","6","6"])
```

```
'1*2*3*4*5*6*6'
```

```
'1*2*3*4*5*6*6'.split("*")
```

```
['1', '2', '3', '4', '5', '6', '6']
```

```
mamueasyshop =  {"charger":500,"datacable":250, "backcover":300,
            "handsfree":800,"protector":200,"sim":500, "powerbank":1000,
            "usb":500, "battery":1500,"mouse":300,"keyboard":750}
```

```
item_name = input("Baji kia chahye? ")
if item_name in mamueasyshop:
    quantity = int(input(f"How many {item_name}"))
    total = quantity * mamueasyshop['item_name']
    print(f"")
else:
    print("Sorry Baji")
```

```
Baji kia chahye?  q
```

```
Start coding or generate with AI.
```

Loops:

```
- For in
- While
```

iterables: string, list, dic, set ,tuple

```
for b in "hello":
    print(b.upper())
```

```
H
E
L
L
O
```

```
range(10)
```

```
range(0, 10)
```

```
list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print("Num","Sqr","Cube")
for num in range(10):

    print(num, num**2, num**3)
```

```
Num Sqr Cube
0 0 0
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
```

```
for a in range(1,11,2):
    print(a,end="")
```

```
13579
```

```
print("hello", "Pakistan",sep="\t\t")
```

```
hello           Pakistan
```

```
mamueasyshop =  {"charger":500,"datacable":250, "backcover":300,
            "handsfree":800,"protector":200,"sim":500, "powerbank":1000,
            "usb":500, "battery":1500,"mouse":300,"keyboard":750}
```

Iterating over a dictionary

```
# by default loop iterated over key in a dictionary
for a in mamueasyshop:
    print(a)
```

```
charger
datacable
backcover
handsfree
protector
sim
powerbank
usb
battery
mouse
keyboard
```

```
for a in mamueasyshop.keys():
    print(a)
```

```
charger
datacable
backcover
handsfree
protector
sim
powerbank
usb
battery
mouse
keyboard
```

```
mamueasyshop.keys()
```

```
dict_keys(['charger', 'datacable', 'backcover', 'handsfree', 'protector', 'sim', 'powerbank', 'usb', 'battery', 'mouse',
'keyboard'])
```

```
for a in mamueasyshop.values():
    print(a)
```

```
500
250
300
800
200
500
1000
500
1500
300
750
```

```
for a in mamueasyshop.items():
    print(a)
```

```
('charger', 500)
('datacable', 250)
('backcover', 300)
('handsfree', 800)
('protector', 200)
('sim', 500)
('powerbank', 1000)
('usb', 500)
('battery', 1500)
('mouse', 300)
('keyboard', 750)
```

Packing and Unpacking

```
# packing
alist = [12,24,36,48]
```

```
#unpack
mangoes, oranges, apples, cherries = alist
```

```
mangoes
```

```
12
```

```
oranges
```

```
24
```

```
for k,v in mamueasyshop.items():
    print(k , v )
```

```
charger 500
datacable 250
backcover 300
handsfree 800
protector 200
sim 500
powerbank 1000
```

```
usb 500
battery 1500
mouse 300
keyboard 750
```

```
relatives  = ['ali', 'asad','arif', 'Amer', 'hashir', 'faiz', 'zaiton']
```

```
for relative in relatives:
    if relative == 'hashir':
        print(f"{relative} you are not invited.")
    else:
        print(f"{relative} you are invited.")
```

```
ali you are invited.
asad you are invited.
arif you are invited.
Amer you are invited.
hashir you are not invited.
faiz you are invited.
zaiton you are invited.
```

```
for relative in relatives:
    if relative.startswith('a'):
        print(relative, 'Invited')
    else:
        print(relative, "Not Invited")
```

```
ali Invited
asad Invited
arif Invited
Amer Not Invited
hashir Not Invited
faiz Not Invited
zaiton Not Invited
```

```
# i need the prices of item costing less than 500
mamueasyshop
```

```
{'charger': 500,
 'datacable': 250,
 'backcover': 300,
 'handsfree': 800,
 'protector': 200,
 'sim': 500,
 'powerbank': 1000,
 'usb': 500,
 'battery': 1500,
 'mouse': 300,
 'keyboard': 750}
```

```
for k,v in mamueasyshop.items():
    if v <=500:
        print(k, v)
```

```
charger 500
datacable 250
backcover 300
protector 200
sim 500
usb 500
mouse 300
```

```
prod = input("What do you wana buy? ")
if prod in mamueasyshop.keys():
    print(f"The price of the {prod} is {mamueasyshop[prod]}")
else:
    print(f"{prod} no in shop.")
```

```
What do you wana buy?  usb
The price of the usb is 500
```

```
mamueasyshop['protector']
```

```
200
```

```
mamueasyshop
```

```
{'charger': 500,
 'datacable': 250,
 'backcover': 300,
 'handsfree': 800,
 'protector': 200,
 'sim': 500,
 'powerbank': 1000,
 'usb': 500,
 'battery': 1500,
 'mouse': 300,
 'keyboard': 750}
```

```
age  = 10
count = 0
while age < 100:
    print(count, "Happy Birthday")
    count+=1
    age+=10
```

```
0 Happy Birthday
1 Happy Birthday
2 Happy Birthday
3 Happy Birthday
4 Happy Birthday
5 Happy Birthday
6 Happy Birthday
7 Happy Birthday
8 Happy Birthday
```

```
count = 0
while count < 10:
    print(count)
    count+=1
```

```
0
1
2
3
4
5
6
7
8
9
```

Break and Continue

```
for a in range(10):
    if a==5:
        break
    else:
        print(a)
```

```
0
1
2
3
4
```

```
for a in range(10):
    if a==5:
        continue
    else:
        print(a)
```

```
0
1
2
3
4
6
7
8
9
```

```
count = 5
cart = []
amt = 0
while count>0:
    prod = input("Wht do you want to buy? ")
    if prod in mamueasyshop.keys():
        print(f"price of the {prod} is {mamueasyshop[prod]}")
        print("Product added in the Cart")
        amt +=mamueasyshop[prod]
        cart.append(prod)
    else:
        print(f"Sorry {prod} not in shop")
    count-=1
print("Total", amt)
print("Cart",cart)
```

```
Wht do you want to buy?  usb
price of the usb is 500
Product added in the Cart
Wht do you want to buy?  mouse
price of the mouse is 300
Product added in the Cart
Wht do you want to buy?  charger
price of the charger is 500
Product added in the Cart
Wht do you want to buy?  keyboard
price of the keyboard is 750
Product added in the Cart
Wht do you want to buy?  protector
price of the protector is 200
Product added in the Cart
Total 2250
Cart ['usb', 'mouse', 'charger', 'keyboard', 'protector']
```

```
amt
```

```
1750
```

```
cart
```

```
['mouse', 'usb', 'keyboard', 'protector']
```

```


cart = []
amt = 0
while True:
    prod = input("Wht do you want to buy? or press q to quit ").lower()
    if prod == 'q':
        break
    else:
        if prod in mamueasyshop.keys():
            if prod in cart:
                print(f"{prod} already in Cart")
            else:

                print(f"price of the {prod} is {mamueasyshop[prod]}")
                print("Product added in the Cart")
                amt +=mamueasyshop[prod]
                cart.append(prod)
        else:
            print(f"Sorry {prod} not in shop")
        count-=1


print("Total", amt)
print("Cart",cart)
```

```
Wht do you want to buy? or press q to quit  usb
price of the usb is 500
Product added in the Cart
Wht do you want to buy? or press q to quit  usb
usb already in Cart
Wht do you want to buy? or press q to quit  eggs
Sorry eggs not in shop
Wht do you want to buy? or press q to quit  keyboard
price of the keyboard is 750
Product added in the Cart
Wht do you want to buy? or press q to quit  sim
price of the sim is 500
Product added in the Cart
Wht do you want to buy? or press q to quit  charger
price of the charger is 500
Product added in the Cart
Wht do you want to buy? or press q to quit  handsfree
price of the handsfree is 800
Product added in the Cart
Wht do you want to buy? or press q to quit  backcover
price of the backcover is 300
Product added in the Cart
Wht do you want to buy? or press q to quit  datacable
price of the datacable is 250
Product added in the Cart
Wht do you want to buy? or press q to quit  q
```

```
Total 3600
Cart ['usb', 'keyboard', 'sim', 'charger', 'handsfree', 'backcover', 'datacable']
```

```
Start coding or generate with AI.
```

While Loop

```
squared_alist = []
alist = [1,2,3,4,5,6,7,8,9,10]
# using for loop
for i in alist:
    squared_alist.append(i**2)
squared_alist
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
squared_alist = []
alist = [11,12,13,14,15,16,17,18,19,20,21]
# using while
a=0
while a<len(alist):
    squared_alist.append(alist[a]**2)
    a+=1
squared_alist
```

```
[121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441]
```

```
favourite_food= []
for a in range(5):
    ff = input("Favourite Food: press q to quit ")
    if ff =="q":
        break
    else:
        favourite_food.append(ff)
favourite_food
```

```
Favourite Food: press q to quit  biryani
Favourite Food: press q to quit  q
['biryani']
```

```
favourite_food= []
for a in range(5):
    ff = input("Favourite Food: press q to quit ")
    if ff =="q":
        break
    else:
        favourite_food.append(ff)
favourite_food
```

```
Favourite Food: press q to quit  Biryani
Favourite Food: press q to quit  Karahi
Favourite Food: press q to quit  Tikkka
Favourite Food: press q to quit  Nihari
Favourite Food: press q to quit  Paye
['Biryani', 'Karahi', 'Tikkka ', 'Nihari', 'Paye']
```

```
favourite_food= []
a = 0
while a<5:
    ff = input("Favourite Food: press q to quit ")
    if ff =="q":
        break
    else:
        favourite_food.append(ff)
        a+=1
favourite_food
```

```
Favourite Food: press q to quit  a
Favourite Food: press q to quit  s
Favourite Food: press q to quit  v
Favourite Food: press q to quit  b
Favourite Food: press q to quit  g
['a', 's', 'v', 'b', 'g']
```

```
favourite_food= []
flag = True
while flag:
    ff = input("Favourite Food: press q to quit ")
    if ff =="q":
        flag= False
    else:
        favourite_food.append(ff)
        a+=1
favourite_food
```

```
Favourite Food: press q to quit  q
[]
```

Functions

```
    - Open functions >> print, type input, len, range, id,
    - class function >> list function, dic, set tuple, strng
    - user defined function:
```

```
def greeteveryone():
    print("Welcome!")
```

```
greeteveryone()
```

```
Welcome!
```

```
def add():
    a =10
    b = 20
    print(a+b)
```

```
add()
```

```
30
```

```
add()
```

```
30
```

```
# parameter less function
# parameterised function
def greeteveryone(name):##parameter
    print(f"Welcome Mr.{name}!")
```

```
greeteveryone()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[104], line 1
----> 1 greeteveryone()

TypeError: greeteveryone() missing 1 required positional argument: 'name'
```

```
greeteveryone('Bilal')#argument
```

```
Welcome Mr.Bilal!
```

```
greeteveryone('Jamal')#argument
```

```
Welcome Mr.Jamal!
```

```
greeteveryone("Nasir")
```

```
Welcome Mr.Nasir!
```

```
def add(a,b):
    print(a+b)
```

```
add()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[114], line 1
----> 1 add()

TypeError: add() missing 2 required positional arguments: 'a' and 'b'
```

```
add(2,3)
```

```
5
```

```
add(5,6)
```

```
11
```

```
add("hassan", "nasir")
```

```
hassannasir
```

```
add(2,3,4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[122], line 1
----> 1 add(2,3,4)

TypeError: add() takes 2 positional arguments but 3 were given
```

```
def add(a,b,c=100,d):# default value
    print(a+b+c+d)
```

```
  Cell In[124], line 1
    def add(a,b,c=100,d):# default value
                      ^
SyntaxError: parameter without a default follows parameter with a default
```

```
def add(a,b,d,c=100):# default value
    print(a+b+c+d)
```

```
add(1,2,3,4)
```

```
10
```

```
add(1,2,3)
```

```
106
```

```
def add(a=0,b=0,d=0,c=0):# default value
    print(a+b+c+d)
```

```
add(1,3,4)
```

```
8
```

keyword argument

```
def generate_profile(name, age, qual, prof, gender):
    profile={'name':name, 'age':age, "qualification":qual, 'profession':prof,
            'gender':gender}
    print(profile)
```

```
generate_profile(age='23',prof="Doctor",gender='Female', name="Shabana",
                qual="MBBS")
```

```
{'name': 'Shabana', 'age': '23', 'qualification': 'MBBS', 'profession': 'Doctor', 'gender': 'Female'}
```

```
generate_profile('bilal', "Matric", 'PythonDev', gender='male',age=23)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[88], line 1
----> 1 generate_profile('bilal', "Matric", 'PythonDev', gender='male',age=23)

TypeError: generate_profile() got multiple values for argument 'age'
```

Arbitrary Arguments

```
def add(a,b=0, *other):
    print(a+b+sum(other))
```

```
add(11,2,3,4,5,5,6,7,7,8,8,9,9,0)
```

```
84
```

```
add(1,2,3,4,5,6,7,8,9,10)
```

```
1
2
(3, 4, 5, 6, 7, 8, 9, 10)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[161], line 1
----> 1 add(1,2,3,4,5,6,7,8,9,10)

Cell In[147], line 5, in add(a, b, *other)
      3 print(b)
      4 print(other)
----> 5 print(a+b+other)

TypeError: unsupported operand type(s) for +: 'int' and 'tuple'
```

```
sum((3, 4, 5, 6, 7, 8, 9, 10))
```

```
52
```

```
def profile_builder(name, age, qualification, income, *other_info):
    profile = {}
    profile['name'] = name
    profile['age'] = age
    profile['qualification'] = qualification
    profile['income'] = income
    profile['other_info'] = other_info

    print(profile)
```

```
profile_builder('arish',24,'Graduate', 40000, 'karachi', '5.8"', 'Fair', 'datascientsit')
```

```
{'name': 'arish', 'age': 24, 'qualification': 'Graduate', 'income': 40000, 'other_info': ('karachi', '5.8"', 'Fair', 'datascientsit
```

```
def profile_builder(name, age, qualification, income, **other_info):
    profile = {}
    profile['name'] = name
    profile['age'] = age
    profile['qualification'] = qualification
    profile['income'] = income
    profile['other_info'] = other_info

    print(profile)
```

```
profile_builder('arish',24,'Graduate', 40000, city='karachi', height='5.8"', complexion='Fair', profession='datascientsit')
```

```
{'name': 'arish', 'age': 24, 'qualification': 'Graduate', 'income': 40000, 'other_info': {'city': 'karachi', 'height': '5.8"', 'comp
```

```
def profile_builder(name, age, qualification, income, **other_info):
    profile = {}
    profile['name'] = name
    profile['age'] = age
    profile['qualification'] = qualification
    profile['income'] = income
    # profile['city'] = other_info['city']
    profile.update(other_info)
    # profile['other'] = other_info

    print(profile)
```

```
profile_builder('arish',24,'Graduate', 40000, city='karachi', height='5.8"', complexion='Fair', profession='datascientsit')
```

```
{'name': 'arish', 'age': 24, 'qualification': 'Graduate', 'income': 40000, 'city': 'karachi', 'height': '5.8"', 'complexion': 'Fair
```

Start coding or generate with AI.

More on Functons

```
- return
- global and local vaiable
```

```
name = "test"   # globbl
def profile_builder(name, age, qualification, income, **other_info):
    profile = {} # local varaible
    profile['name'] = name
    profile['age'] = age
    profile['qualification'] = qualification
    profile['income'] = income
    # profile['city'] = other_info['city']
    profile.update(other_info)
    return profile
```

```
profile_builder('arish',24,'Graduate', 40000, city='karachi', height='5.8"', complexion='Fair', profession='datascientsit')
```

```
{'name': 'arish',
 'age': 24,
 'qualification': 'Graduate',
 'income': 40000,
 'city': 'karachi',
 'height': '5.8"',
 'complexion': 'Fair',
 'profession': 'datascientsit'}
```

```
print(profile)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 print(profile)

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

```
def add(a,b):
    print(a+b)
```

```
add(10,20) + 30
```

```
30
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 add(10,20) + 30

TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

```
def add(a,b):
    return a+b
```

```
print(add(2,3))
```

```
5
```

```
ans
```

```
5
```

```
def add(a,b):
    return a+b, "Pakistan Zindabad"
```

```
add(4,5)
```

```
(9, 'Pakistan Zindabad')
```

```
# salesman: commmsionCalc
# taxCalc
# Payment
```

```
def TaxCalc(income):
    if income>=50000:
        return income*0.10
    elif income>=40000:
        return income*0.05
    elif income>=30000:
        return income*0.03
    else:
        return income*0
```

```
def CommisionCalc(units):
    if units >=500:
        return 10000
    elif units>=250:
        return 5000
    elif units>=100:
        return 2000
    else:
        return 1000
```

```
def SalaryCalculator(basic, units_sales):
    comision = CommisionCalc(units_sales)
    total = basic+comision
    tax = TaxCalc(total)
    net  = total - tax
    return net
```

```
SalaryCalculator(50000, 120)
```

```
46800.0
```

```
def sum_list(lst):
    s = 0
    for a in lst:
        s+=a
    return s
```

```
sum_list([9,8,7,6,5,4,3,2,1,10])
```

```
55
```

```
def max_list(lst):
    max_num = lst[0]
    for n in lst:
        if n > max_num:
            max_num = n
    return max_num
```

```
max_list([12,34,34,23,145,65,78,90,34,2])
```

```
145
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
def square_dic(n):
    sq = {}
    for a in range(1,n+1):
        square = a**2
        sq[a] = square
    return sq
```

```
square_dic(10)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

```
Question:
Write a program which accepts a sequence of comma separated 4 digit binary numbers
as its input and then check whether they are divisible by 5 or not.
The numbers that are divisible by 5 are to be printed in a comma separated sequence.
Example:
0100,0011,1010,1001
```

```
Then the output should be:
1010
```

```
  Cell In[25], line 6
    0100,0011,1010,1001
    ^
SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o prefix for octal integers
```

```
bn = input("Enter 4 comma separated 4digit binary numbers").split(",")
for n in bn:
    if int(n)%5==0:
```

```
Enter 4 comma separated binary numbers 1010,1,0101011,11101,1110
```

```
bn
```

```
['1010', '1', '0101011', '11101', '1110']
```

```
int(0b1010)
```

```
10
```

```
bin(10)
```

```
'0b1010'
```

```
Start coding or generate with AI.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[43], line 1
----> 1 int(bin(int('1010')))

ValueError: invalid literal for int() with base 10: '0b1111110010'
```

```
Start coding or generate with AI.
```

The roots of the quadratic equation ax2 + bx + c = 0, a ≠ 0 are given by the following formula: In this formula, the term b2 - 4ac is called the discriminant. If b2 - 4ac = 0, then the equation has two equal roots. If b2 - 4ac > 0, the equation has two real roots. If b2 - 4ac < 0, the equation has two complex roots. Write a program that prompts the user to input the value of a (the coefficient of x2), b (the coefficient of x), and c (the constant term) and outputs the roots of the quadratic equation.

```
x1,x2 = -b+(b**2-4*a*c)**0.5/(2*a),-b-(b**2-4*a*c)**0.5/(2*a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-f626a94e7c75> in <cell line: 1>()
----> 1 x1,x2 = -b+(b**2-4*a*c)**0.5/(2*a),-b-(b**2-4*a*c)**0.5/(2*a)

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

```
def dicrimant(a,b,c):
  if (b**2 -4*a*c) > 0:
    return "Real"
  elif (b**2 -4*a*c) == 0:
    return "Egual"
  else:
    return "Imaginary"
dicrimant(1,2,5)
```

```
'Imaginary'
```

```
Start coding or generate with AI.
```

Modules

```
import math
```

```
import datetime
```

```
import os
```

```
Start coding or generate with AI.
```

```
import myutility
```

```
%run myutility.py
```

```
myutility.max_list([2,43,56,78,76,89,90,113])
```

```
113
```

```
from myutility import SalaryCalculator
```

```
SalaryCalculator(10000,500)
```

```
20000
```

```
%run myutility.py
```

```
from myutility import email
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-18-09e6e91ad195> in <cell line: 1>()
----> 1 from myutility import email

ImportError: cannot import name 'email' from 'myutility' (/content/myutility.py)

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES

```
Start coding or generate with AI.
```

Exception Handling

```
num1 = int(input("Enter a number "))
num2 = int(input("Enter another number "))

print(num1/num2)
```

```
Enter a number 12
Enter another number 0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-22-306cb0cda495> in <cell line: 4>()
      2 num2 = int(input("Enter another number "))
      3
----> 4 print(num1/num2)

ZeroDivisionError: division by zero
```

```
try:
    num1 = int(input("Enter a number "))
    num2 = int(input("Enter another number "))
    print(num1/num2)
except ZeroDivisionError:
  print("You can't divide by zero")
except ValueError:
  print("You can't enter a string")
```

```
Enter a number True
You can't enter a string
```

```
try:
    num1 = int(input("Enter a number "))
    num2 = int(input("Enter another number "))
    print(num1/num2)
except Exception as e:
  print(e)
```

```
Enter a number 12
Enter another number 0
Invalid input
```

```
users = []

while True:
  user_name = input("Enter a username or press q to quit")
  if user_name == "q":
    break
  try:
    if user_name in users:
      raise Exception("Username already exists")
    users.append(user_name)

  except Exception as e:
    print(e)


print(users)
```

```
Enter a username or press q to quitasad
Enter a username or press q to quitsaad
Enter a username or press q to quitasad
Username already exists
Enter a username or press q to quitq
['asad', 'saad']
```

```
try:
  name = input("Enter name")
  age  = int(input("Enter your age "))

  if age < 18 :
    raise Exception("You are not old enough")
  elif age >=100:
    raise Exception("You are too old")
except Exception as e:
  print(e)
```

```
Enter nameasad
Enter your age 200
You are too old
```

```
try:
  num1 = int(input("Enter a number "))
  num2 = int(input("Enter another number "))

  result = num1/num2

except Exception as e:
  print(e)


else:
   print(result)
finally:
  print("Koi chaly na chalay finally to chalega")
```

```
Enter a number 3
Enter another number two
invalid literal for int() with base 10: 'two'
Koi chaly na chalay finally to chalega
```

Start coding or generate with AI.

15. Write a Python program that prompts the user to enter a base number and an

```
exponent, and then calculates the power of the base to the exponent. The program
should not use the exponentiation operator (**) or the math.pow() function. The
program should handle both positive and negative exponents.
```

```
def calculate_exponent(num, exp):
    s = 1# 5,25,125
    for a in range(1,exp+1):
        s= s*num
    return s
```

```
calculate_exponent(3,4)
```

```
81
```

File : Reading and writing Text Files I/O Stream

```
try:
    f = open('sample.txt', 'r')
    cont = f.read()
    print(cont)

except FileNotFoundError:
    print("The file you are looking for is not in this directory")
finally:
    f.close()
```

```
The information in this email is confidential and may be legally privileged. Access to this email by

 anyone other than the intended addressee is unauthorized. If you are not the intended recipient of

 this message, any review, disclosure, copying, distribution, retention, or any action taken or omitted

 to be taken in reliance on it is prohibited and may be unlawful. If you are not the intended recipient,

 please reply to or forward a copy of this message to the sender and delete the message, any

 attachments, and any copies thereof from your system.
```

```
try:
    f = open('sample.txt', 'w')
    f.write("This is a sample file")

except FileNotFoundError:
    print("The file you are looking for in not in this directory")
finally:
    f.close()
```

```
try:
    f = open('sample.txt', 'r')
    cont = f.read()
    print(cont)

except FileNotFoundError:
    print("The file you are looking for in not in this directory")
finally:
    f.close()
```

```
This is a sample file
```

```
try:
    f = open('sample2.txt', 'w')
    f.write("This is a sample file\n\n")
    f.write("This ia sample file line2")

except FileNotFoundError:
    print("The file you are looking for in not in this directory")
finally:
    f.close()
```

```
try:
    f = open('sample2.txt', 'r')
    cont = f.read()
    print(cont)

except FileNotFoundError:
    print("The file you are looking for in not in this directory")
finally:
    f.close()
```

```
This is a sample file

This ia sample file line2
```

Context Manager

```
with open("samplefile3.txt","w") as f:
    f.write("Hello1 \n")
    f.write("Hello2 \n")
    f.write("Hello3 \n")
    f.write("Hello4 \n")
    f.write("Hello5 \n")
```

```
with open("samplefile3.txt","r") as f:
    print(f.read())
```

```
Hello1
Hello2
Hello3
Hello4
Hello5
```

```
with open('samplefile3.txt', 'a') as f:
    f.write("This Hello6")
```

```
with open("samplefile3.txt","r") as f:
    print(f.read())
```

```
Hello1
Hello2
Hello3
Hello4
Hello5
This Hello6
```

```
with open('samplefile3.txt') as f:
    print(f.readline())
```

```
Hello1
```

```
with open('samplefile3.txt') as f:
    print(f.readlines())
```

```
['Hello1 \n', 'Hello2 \n', 'Hello3 \n', 'Hello4 \n', 'Hello5 \n', 'This Hello6']
```

```
with open('samplefile3.txt') as f:
    for line in f:
        if "5" in line:
            continue
        print(line, end='')
```

```
Hello1
Hello2
Hello3
Hello4
This Hello6
```

```
with open("samplefile3.txt","r") as f:
    print(f.read())
```

```
Hello1
Hello2
Hello3
```

```
Hello4
Hello5
This Hello6
```

```
with open("samplefile3.txt","a") as f:
    f.writelines(['Say1\n', 'Say2\n', "Say3\n"])
```

```
with open("samplefile3.txt","r") as f:
    print(f.read())
```

```
Hello1
Hello2
Hello3
Hello4
Hello5
This Hello6Say1Say2Say3Say1
Say2
Say3
```

```
with open("sample.txt", 'r+') as f:
    print(f.read())
    f.write("\n This is line2")
    print(f.read())
    f.write("\nThis is line3")
    print(f.read())
```

```
This is a sample file
 This is line2
 This is line2
 This is line2
This is line3
 This is line2
This is line3
```

```
with open("newsample.txt", 'w+') as f:
    f.write("This is new sample line1")
    f.seek(10)
    print(f.read())
```

```
w sample line1
```

You are a hotel manager:

```
guest check in
guest check out
```

```
guests = ['Ali', 'Imran', 'Asim','Bhutto','Asif']
with open("guests1.txt", 'w+') as f:
    for guest in (guests):
        f.write(f"{guest}\n")
    f.seek(0)
    print(f.read())
```

```
Ali
Imran
Asim
Bhutto
Asif
```

```
check_out = ['Asif', 'Bhutto']
```

```
temp_guests = []
with open("guests1.txt", 'r+') as f:
    for line in f:
        temp_guests.append(line)
    for guest in temp_guests:
        if f"{guest}\n" not in check_out:
            f.write(f"{guest}\n")
    f.seek(0)
    print(f.read())
```

```
Ali
Imran
Asim
Bhutto
Asif
Ali

Imran

Asim

Bhutto

Asif

Ali

Imran

Asim

Bhutto

Asif

Ali


Imran


Asim


Bhutto


Asif
```

```
temp_guests
```

```
[]
```

```
with open("guests1.txt", 'r+') as f:
    for line in f:
        temp_guests.append(line)
```

```
temp_guests
```

```
['1- Ali\n', '2- Imran\n', '3- Asim\n', '4- Bhutto\n', '5- Asif\n']
```

```
checked_out=["Ali", "Bhutto"]
temp_list=[]

with open("guests1.txt", 'r') as guests:
    for g in guests:
        temp_list.append(g.strip())

with open("guests1.txt", 'w+') as guests:
    for name in temp_list:
        if name not in checked_out:
            guests.write(name + "\n")
```

```
with open("guests1.txt", 'r') as f:
    for line in f:
        print(line, end="")
```

```
Imran
Asim
Asif

Imran

Asim
```

```
Asif


Imran

Asim


Asif




Imran



Asim






Asif
```

```
Start coding or generate with AI.
```

Object Oriented Programming

Class :

```
Class is a map/blueprint/model of an object. Class is an implementation of object.
```

Object :

```
 Object is an instance of class. Drived from class. Will follow 100% to its class
```

image.pngimage.png

```
class Person():
    pass
```

```
p1 = Person()
```

```
print(type(p1))
```

```
<class '__main__.Person'>
```

```
num = 100
```

```
print(type(num))
```

```
<class 'int'>
```

```
name = 'asad'
print(type(name))
```

```
<class 'str'>
```

```
alist = []
print(type(alist))
```

```
<class 'list'>
```

```
adic = {}
print(type(adic))
```

```
<class 'dict'>
```

```
blist = list((1,2,3,4))
blist
```

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

```
blist = [1,2,3,4]
blist
```

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

```
Start coding or generate with AI.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 1
----> 1 bdic = dict(('name', 'asad'))
      2 bdic

ValueError: dictionary update sequence element #0 has length 4; 2 is required
```

```
class Student():
    name = "Asadullah"
    age = 30
    city = "Karachi"
```

```
s1 = Student()
```

```
s1.name
```

```
'Asadullah'
```

```
s1.age
```

```
30
```

```
s1.city
```

```
'Karachi'
```

```
s2 = Student()
```

```
s2.name = "Saad"
s2.age = 20
s2.city = "Lahore"
```

```
s1.name
```

```
'Asadullah'
```

```
s2.name
```

```
'Saad'
```

```
class Student():
    # initializer / constrcutor
    def __init__(self,name, age, city, course='Python'):
        self.name = name
        self.age = age
        self.city = city
        self.course  = course


    # functions / Methods
    def apper_in_exam(self):
        print(f"{self.name} is appearing in exam")

    # functions / Methods
    def pay_fees(self):
        print(f"{self.name} is paying fee")
```

```
s4 = Student()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[54], line 1
----> 1 s4 = Student()

TypeError: Student.__init__() missing 3 required positional arguments: 'name', 'age', and 'city'
```

```
s4 = Student('Taha', 20, "Karachi", 'Python')
```

```
s5 = Student('Wali', 23,'Lahore')
```

```
s4.apper_in_exam()
```

```
Taha is appearing in exam
```

```
s5.apper_in_exam()
```

```
Wali is appearing in exam
```

```
class Car():
    def __init__(s,model, make, color='Black'):
        s.model =model
        s.make = make
        s.color = color
        s.ac = 'Dawlance'
    def describe_car(s):
        print(f"""
             Model Name: {s.model}
             Car Make:   {s.make}
             Car Color:  {s.color}
             Car Ac: {s.ac}

             """)
```

```
c1 = Car('2024', "Honda", 'Black')
c2 = Car('2025', "Honda", 'White')
```

```
c1.describe_car()
```

```
                  Model Name: 2024
                  Car Make:   Honda
                  Car Color:  Black
                  Car Ac: Dawlance
```

```
c2.describe_car()
```

```
                  Model Name: 2025
                  Car Make:   Honda
                  Car Color:  White
                  Car Ac: Dawlance
```

```
Car.describe_car(c2)
```

```
                  Model Name: 2025
                  Car Make:   Honda
                  Car Color:  White
                  Car Ac: Dawlance
```

```
Car.describe_car(c1)
```

```
                  Model Name: 2024
                  Car Make:   Honda
                  Car Color:  Black
                  Car Ac: Dawlance
```

Start coding or generate with AI.