```
# To check data type
print(type([]))
```

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

```
# pass keyword used to terminate class without body
class sample:
  pass

x = sample()

print(type(x))
```

```
<class '__main__.sample'>
```

```
# Example of class and object
class Dog:
  def __init__(self, breed):
    self.breed = breed

Tommy = Dog(breed='Lab')
Tisan = Dog(breed='Huskie')

Tommy.breed
```

```
{"type":"string"}
```

```
Tisan.breed
```

```
{"type":"string"}
```

```
#Another example of class and objects
class Dog:
  species = 'Mammal'
  def __init__(self, breed,name):
    self.breed=breed
    self.name=name



My_Dog = Dog('Lab','jerry') # objects

My_Dog.name
```

```
{"type":"string"}
```

```
My_Dog.breed
```

```
{"type":"string"}
```

```
My_Dog.species
```

```
{"type":"string"}
```

```python
#Find Area and circumference of circle with the help of classs and
objects
class circle:
  pi = 3.14
  def __init__(self, radius=1):
    self.radius = radius
    self.area = radius * radius * circle.pi

  def setRadius(self,new_radius):
    self.radius = new_radius
    self.area = new_radius * new_radius * self.pi

  def getcircumference(self):
    return self.radius * self.pi * 2

c1 = circle()    #Finf area and circumference of cirle with default
value of radius
print('The Radius is : ', c1.radius)
print('The Area is : ', c1.area)
print('The Circumference is : ', c1.getcircumference())
```

```
The Radius is :  1
The Area is :  3.14
The Circumference is :  6.28
```

```python
c1.setRadius(4)     #Finf area and circumference of cirle with  value
of radius = 4
print('The Radius is : ', c1.radius)
print('The Area is : ', c1.area)
print('The Circumference is : ', c1.getcircumference())
```

```
The Radius is :  4
The Area is :  50.24
The Circumference is :  25.12
```

```python
# Track the object from its coordinates
class track:
  def __init__(self, coor1,coor2):
    self.coor1 = coor1
    self.coor2 = coor2


  def distance(self):
    x1,y1 = self.coor1
    x2,y2 = self.coor2
    return ((x2 - x1)**2 + (y2 - y1)**2)**0.5


  def slope(self):
    x1,y1 = self.coor1
    x2,y2 = self.coor2
```

```
    return (y2 - y1)/(x2 - x1)
```

```
c1 = (12,9)
c2 = (3,8)

find = track(c1,c2)

find.distance() # To find out Distance of the object
```

```
9.055385138137417
```

```
find.slope()  # To find out Slope of the object
```

```
0.1111111111111111
```

```
# Manage Deposit and Withdraw money with customer Account
class Account:
  def __init__(self,owner,balance=0):
    self.owner = owner
    self.balance = balance


  def __str__(self):
    return f'Account Owner:          {self.owner}\nAccount Balance:
${self.balance}'


  def deposit(self,dep_amt):
    self.balance += dep_amt
    print("Deposit Accepted")


  def withdraw(self,wd_amt):
    if self.balance>= wd_amt:
      self.balance -= wd_amt
      print("Withdraw Accepted")

    else:
      print("Funds Unavailable")

acc1 = Account('Rahul', 20000) # To creat account

print(acc1)   # To display info of account holder
```

```
Account Owner:          Rahul
Account Balance:        $20000
```

```
acc1.owner # to display account owner name
```

```
{"type":"string"}
acc1.balance # To display account holder name
20000
  acc1.deposit(5000)  # To add money in account holder account
Deposit Accepted
acc1.balance # To check account balance
25000

acc1.withdraw(30000) # To withdraw money from the account holder but



Funds Unavailable
acc1.withdraw(19000)
Withdraw Accepted
acc1.balance
6000
```