DAY 8 OF 200 DAY'S PYTHON CHALLENGE

OBJECT ORIENTED PROGRAMMING SYSTEM(OOPS)

Object-oriented programming (OOP) is a programming paradigm that uses objects and classes in programming. OOPs, concepts in python, aim to implement real-world entities like inheritance, polymorphisms, encapsulation, etc., in the programming.

Day 8.py

```
class test:
   pass
a = test()
print(type(a))
```

EXAMPLES

Day 8.py

```
class Practice:
  def welcome_msg(self):
    print("Welcome to 200 day's Challenge")

abhi=Practice()
abhi.welcome_msg()
```

Day 8.py

```
class accademy:
  def __init__(self, Ph_no, email_id, st_id):
  # self keyword is bounding any variable into the class
    self.Ph_no = Ph_no
    self.email_id = email_id
    self.st_id = st_id

  def return_msg(self):
    return self.Ph_no, self.email_id, self.st_id
```

POLYMORPHISM

Polymorphism is the ability for an object to take on many forms. This allows for different objects to be treated in the same way, even if they have different implementations.

Day 8.py

```
class data_science:
  def syllabus(self):
    print("Data Science")
class web_dev:
  def syllabus(self):
    print("Web Dev")
def class_parcer(class_obj):
  for i in class_obj:
    i.syllabus()
data_science = data_science()
web_dev = web_dev()
class_obj = [data_science,web_dev]
class_parcer(class_obj)
```

ENCAPSULATION

Encapsulation is the bundling of data and methods into a single unit, called an object. This bundling helps to protect the data from outside interference and ensures that the data is used in a consistent manner.

Day 8.py

```
class test:
    def __init__(self,year,speed):
        self.__year = year
        self.__speed = speed

     def set_speed(self,speed):
        self.__speed = 0 if speed < 0 else speed

     def get_speed(self):
         return self.__speed
```

BANK MANAGEMENT

Day 8.py

```
class bank_account:

  def __init__(self, balance):
    self.__balance = balance

  def deposit(self, amount):
    self.__balance = self.__balance + amount

  def withdraw(self, amount):
    if self.__balance >= amount:
      self.__balance = self.__balance - amount
      return True
    else:
      return False

  def get_balance(self):
    return self.__balance
```

FOLLOW FOR MORE

FOLLOW ME