Day_3 Python

Topics Covered:-

OOPs Tutorial

Exceptional Handling

Python OOPs :- Inheritance

Magic Method class

Python Generator Vs Iterator

Decorators

OOPs Tutorial in Python

Class → Real world object
- have attributes { features }
- functions like car is used for driving.
- Car. windows = 4 } features

proper way of defining class

```
class Car:                                  → [constructor] → directing the object
    def __init__(self, window, door, engine : type):
        self.windows = window
        self.door = door
        self.enginetype = enginetype
```

instance
```
car1 = Car(4, 5, "petrol")
print(car1.window)
```
output = 4

Exceptional Handling

Query: Difference b/w error and Exception ?

```
try:
    # code block where exception can occur
except:
    print("Some problem may have occured")
```

* we can write what we want to show ~~etc~~ instead of error to maintain readibility.

for ex: (i) Name error
(ii) Zero division error
(iii) Invalid syntax
so that this give good meaning for user.

for ex Custom Exception

```
class Error (Exception):
    pass
class dobException (Error):
    pass
class customgenric (Error
pass
year = int(input("Enter the year of Birth"))
age = 2021 - year
try:
if age <= 30 and cage >20:
    pass
else
    raise dobException
except dobException
    print("The year age is not valid")
```

Python OOPs - inheritance

↳ enheriting features from parent class

```
## BluePrint car
class Car():
    def __init__(self, windows, doors, enginetype):
        self.window = windows
        self.doors = doors
        self.enginetype = enginetype
    def drive():
        print ("The Person drive the car")

car = (4,5,"diesel")
car.drive()

class audi (car):
    def __init__ (self, windows, doors, enginetype, enableai):
        super().__init__(windows, door, enginetype)
        self.enableai = enable ai
```

from parent class:

Python OOPs - Magic Methods in classes

take some blue print of car

result: c = car(4.5, "Diesel")
output <__main__.Car at 0x2b15892

dir(c)

* Its automatically calling methods or it's just initialising magically.

Python Assert

It provides the assert statement to check if a given logical expression is true or false. Program execution proceed only if true or False the AssertionError when it is false

```
10 >= 10
True
```

ex
```
num = 10
assert num >= 10
True
```

Python Generators Vs Iterators

Generator

```
def square(n):
    for i in range(n):
        return i**2
```

yield

This basically is creating iterator
It is generator object

ex
```
a = square(3)
a -> provides location
=> next(a)
0
=> next(a)
1
```

* To create iterator we use iter() and to generator we use function along with Yield keyword.

* Yields save or store local variable.
* Generator in python help us to write code fast and compact
* Python iterator is much more memory efficient.

Python Decorators:
function copy
closures
decorators

```
def welcome():
    return "Hi How are You?"

wel = welcome()
del welcome
```

} Basically here copying is done.

Output: Hi How are You?

Closures:
When we write function inside function is called closure.
⇒ we can access enbuilt function from parent function to sub function.

Python Decorators

```
def main_welcome(func):
    def sub_welcome_class():
        print("Welcome")
        func()
        print("Please subscribe")
    return sub_welcome_class()

def channel_name():
    print("Aniket")
```