OOPS ==> Object Oriented Programming

Used for complex programming.

As like Dynammic Programming.

Imagine a self driving car. there are many section like, camera for identifying, maps for giving routes and lane assistance for identifying whether the car is following the lane or not.

for these like examples, there are large complex codes.. to organize these code, OOP comes into action.

Here each work is assigned to someone. so that it may be easier to handle the task.

For example:

Consider, we need to create a Virtual Restaurant. When we create a Virtual restaurant, we need to create Virtual chef, virtual waiter, virtual cleaner.

Lets model a waiter :

there are two things we need to see there. [what it has], [what it does].

[what it has]:

we can assign a variable ==> holding_a_plate = True (or) False.

or which table is it responsible for ==> tables_responsible = [4,5,6].

[what it does]:

may be it can take order to the chef and may it also need to be take payments and add money to the restaurant. in that case, we can code like this:

def take_order(table,order): #takes order to chef

def take_payment(amount): #add money to the restaurant

here we can technicaly say the [what is has] ==> **attributes** and [what it does] ==> **methods** and the [waiter] ==> **model.**

attribute is basically a variable that is attached to the particular model.

method is a function that a particular model can do.

Point to remember

These are not free floating function or free floating variable. beacuse the particular attribute and the particular method is assigned to the particular model.

We can able to generate multiple versions of the object.

example: waiter ==> henry and betty

here waiter is called as **class** and individual characters which are genrated using the blueprint **(class)** is called as **object**. here, **henry and betty are objects**.

Let consider a car.

so the blueprint of the car is said to be **class** and it may contain:

```
-> colour

-> wheels

-> milege

-> abiltiy to drive

-> ability to break

etc.,
```

now we can create a **object** from the blueprint **object** is like

```
==> car = CarBlueprint()
```

Here **car** is the **object**, and the **CarBlueprint()** is the **class**. because we created **car** from the **CarBlueprint().**

Note: classes should be in capital.

Let we can construct a code using class. Here we are going to use the class which is predefined by someone.

Example: Turtle is a predifned class wich can able to aint or draw something.. it is defined as the turtle with a paintbrush in its back. it is having different brush, colors, etc.,

turtle documentation: https://docs.python.org/3/library/turtle.html

pip install {packagename as per Pypi}

↑ To install a package

In [ ]:
```
# import turtle

# paint = turtle.Turtle()
# Turtle is the class in the turtle package and
# The Class is always started using uppercase
# here "paint" the object. just like assigning variable to a function.
```

```
# we can also use this code ↓

from turtle import Turtle, Screen
paint = Turtle()

print(paint)  #<- an seperate window will be opened with a arrow.
paint.shape("triangle") # triangle shaped brush had been created.
paint.color('coral') # changes the colour of the pen
paint.forward(100) # creating a box
paint.right(90) # turning 90 degree right
paint.forward(100)
paint.right(90)
paint.forward(100)
paint.right(90)
paint.forward(100)
paint.right(90)

my_screen = Screen()
print(my_screen.canvheight)
my_screen.exitonclick()   #exit the code when i am clicking on the screen
```

In [ ]:

```
# creating a code which provides table in a ascii format
# using prettytable package

# pretty table package is downloaded from the extension in pycharm

from prettytable import PrettyTable

table = PrettyTable()    # table as object, PrettyTable() as class
# print(table)   # printing an empty table

'''output:
    ++
    ||
    ++
    ++
'''

# refer the documentation of the package before using it from pypi website

table.add_column("Name",["hritick","aswin","karthi"])
table.add_column("Reg_no",["21BAD203","21BAD204","21BAD207"])

print(table)


'''
output:
    +---------+----------+
    |   Name  |  Reg_no  |
    +---------+----------+
    | hritick | 21BAD203 |
    |  aswin  | 21BAD204 |
    |  karthi | 21BAD207 |
    +---------+----------+
'''
```

Project of the day

Coffee Machine using OOP

menu.py

In [ ]:
```
'''
class MenuItem:
    """Models each Menu Item."""
    def __init__(self, name, water, milk, coffee, cost):
        self.name = name
        self.cost = cost
        self.ingredients = {
            "water": water,
            "milk": milk,
            "coffee": coffee
        }


class Menu:
    """Models the Menu with drinks."""
    def __init__(self):
        self.menu = [
            MenuItem(name="latte", water=200, milk=150, coffee=24, cost=2.5),
            MenuItem(name="espresso", water=50, milk=0, coffee=18, cost=1.5),
            MenuItem(name="cappuccino", water=250, milk=50, coffee=24, cost=3),
        ]

    def get_items(self):
        """Returns all the names of the available menu items"""
        options = ""
        for item in self.menu:
            options += f"{item.name}/"
        return options

    def find_drink(self, order_name):
        """Searches the menu for a particular drink by name. Returns that item i
        for item in self.menu:
            if item.name == order_name:
                return item
        print("Sorry that item is not available.")
'''
```

money_machine.py

In [ ]:
```
'''
class MoneyMachine:

    CURRENCY = "$"

    COIN_VALUES = {
        "quarters": 0.25,
        "dimes": 0.10,
        "nickles": 0.05,
```

```
        "pennies": 0.01
    }

    def __init__(self):
        self.profit = 0
        self.money_received = 0

    def report(self):
        """Prints the current profit"""
        print(f"Money: {self.CURRENCY}{self.profit}")

    def process_coins(self):
        """Returns the total calculated from coins inserted."""
        print("Please insert coins.")
        for coin in self.COIN_VALUES:
            self.money_received += int(input(f"How many {coin}?: ")) * self.COIN
        return self.money_received

    def make_payment(self, cost):
        """Returns True when payment is accepted, or False if insufficient."""
        self.process_coins()
        if self.money_received >= cost:
            change = round(self.money_received - cost, 2)
            print(f"Here is {self.CURRENCY}{change} in change.")
            self.profit += cost
            self.money_received = 0
            return True
        else:
            print("Sorry that's not enough money. Money refunded.")
            self.money_received = 0
            return False
'''
```

coffee_maker.py

In [ ]:

```
'''
class CoffeeMaker:
    """Models the machine that makes the coffee"""
    def __init__(self):
        self.resources = {
            "water": 300,
            "milk": 200,
            "coffee": 100,
        }

    def report(self):
        """Prints a report of all resources."""
        print(f"Water: {self.resources['water']}ml")
        print(f"Milk: {self.resources['milk']}ml")
        print(f"Coffee: {self.resources['coffee']}g")

    def is_resource_sufficient(self, drink):
        """Returns True when order can be made, False if ingredients are insuffi
        can_make = True
        for item in drink.ingredients:
            if drink.ingredients[item] > self.resources[item]:
                print(f"Sorry there is not enough {item}.")
                can_make = False
```

```
        return can_make

    def make_coffee(self, order):
        """Deducts the required ingredients from the resources."""
        for item in order.ingredients:
            self.resources[item] -= order.ingredients[item]
        print(f"Here is your {order.name} ☕. Enjoy!")

'''
```

main.py

In [ ]:

```
# same of Day_015 project

# here you need to get hands on practice about classes and objects

# menu.py, money.py , money_machine.py, coffee maker.py
# was provided
# in main.py, you can get hint about what are the classes we are going to use.

# refer each py file to see what each class is doing.


# 1 -> Print report
# 2 -> is resources sufficient
# 3 -> Process coins
# 4 -> check transaction successfull
# 5 -> make coffee

from menu import Menu, MenuItem
from coffee_maker import CoffeeMaker
from money_machine import MoneyMachine

# 1 -> Print report
machine_money = MoneyMachine()
coffee_maker = CoffeeMaker()
menu = Menu()
is_on = True
# menu_item = MenuItem()

while is_on:
    options = menu.get_items()
    selected_option = input(f"What should you need from ({options}) ? :")
    if selected_option == "off":
        is_on = False
    # 1 -> Print report
    elif selected_option == "report":
        coffee_maker.report()
        machine_money.report()
    # 2 -> is resources sufficient
    else:
        drink = menu.find_drink(selected_option)        # return object type
        if coffee_maker.is_resource_sufficient(drink):
            # 3 -> Process coins, # 4 -> check transaction successfull
            if machine_money.make_payment(drink.cost):
                # 5 -> make coffee
                coffee_maker.make_coffee(drink)
```

```
# print(menu.find_drink("latte"))
# print(menu_item.name("latte"))
# print(coffee_maker.is_resource_sufficient("latte"))


'''
output:
-------

What should you need from (latte/espresso/cappuccino/) ? :latte
Please insert coins.
How many quarters?: 10
How many dimes?: 0
How many nickles?: 0
How many pennies?: 0
Here is $0.0 in change.
Here is your latte ☕. Enjoy!
What should you need from (latte/espresso/cappuccino/) ? :report
Water: 100ml
Milk: 50ml
Coffee: 76g
Money: $2.5
What should you need from (latte/espresso/cappuccino/) ? :cappuccino
Sorry there is not enough water.
What should you need from (latte/espresso/cappuccino/) ? :espresso
Please insert coins.
How many quarters?: 15
How many dimes?: 0
How many nickles?: 0
How many pennies?: 0
Here is $2.25 in change.
Here is your espresso ☕. Enjoy!
What should you need from (latte/espresso/cappuccino/) ? :report
Water: 50ml
Milk: 50ml
Coffee: 58g
Money: $4.0
What should you need from (latte/espresso/cappuccino/) ? :espresso
Please insert coins.
How many quarters?: 5
How many dimes?: 0
How many nickles?: 00
How many pennies?: 0
Sorry that's not enough money. Money refunded.
What should you need from (latte/espresso/cappuccino/) ? :espresso
Please insert coins.
How many quarters?: 10
How many dimes?: 0
How many nickles?: 0
How many pennies?: 0
Here is $1.0 in change.
Here is your espresso ☕. Enjoy!
What should you need from (latte/espresso/cappuccino/) ? :espresso
Sorry there is not enough water.
What should you need from (latte/espresso/cappuccino/) ? :report
Water: 0ml
Milk: 50ml
```

```
Coffee: 40g
Money: $5.5
What should you need from (latte/espresso/cappuccino/) ? :off
'''
```