OOPs: object oriented Programming

Class is a Template or Blue Print which is actually describing what a particular object can have. Object can have Member Variables (Features/attributes) and Behaviour (functions/methods) Let's take a simple example:

```
class Server:
# Object initialization
# member variables serverId,name,ip,status,memory_usage
def __init__(self,serverId,name,ip):
self.serverId=serverId
self.name=name
self.ip=ip
self.status= "offline"
self.memory_usage=0
## Methods
def ping(self):
return f"pinging {self.ip}......."
def getStatus(self):
return f"{self.name} status is {self.status}"
def updateStatus(self):
if self.status=="offline":
self.status= "online"
else:
self.status = "offline"
print(f'{self.name} status updated successfully')
## Creating objects
web_server1 = Server("SI123456789","Webserver1","192.168.1.10")
result1=web_server1.ping()
print(result1)
web_server2 = Server("SI123456710","Webserver2","192.168.1.11")
print(web_server2.ping())
print(web_server1.getStatus())
web_server1.updateStatus()
print(web_server1.getStatus())
```

Here I have created a server class which is having init method which actually describing the initialization of Object

Means when the object is created it declared all the variables associated with that object in the methory.

When you trigger any function or trying to access the variable using that object name like obj1.status means it will take that objects data
from memory.

To understand that by oops system we are using self keyword here.

Inheritance:

To reuse the functionality of a class which is already created we can take help of inheritance.
Child can access all the functionalities of parent class.

Regex

re module provides functionalities to work with regex in Python
most common functions

re.search() : finds the first match anywhere in the provided text.
re.match(): trying match in the beginning
re.findall(): returns all matches as list
re.finditer(): return match objects as iterator
re.sub(): substitube pattern with new string
re.split(): splits a string by a pattern

example for match

```
import re

pattern = r"\d{3}-\d{2}-\d{4}" #SSN Pattern
text = "Sonam SSN is 123-45-6789"

match = re.search(pattern, text)
if match:
print("Match Found", match.group())
```

Using FindAll

```
import re

text = "Contact me at dev@pw.live.com or admin@pw.org"
pattern =r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]+\.[a-zA-Z]{2,}"
```

```
emails = re.findall(pattern, text)
print(emails)
```