In [ ]:
```
# object oriented programming

paytm:
    phonenum
    password
    kyc
    email

        RECHARGE
        ADD MONEY TO
```

In [86]:
```
class Emp:
    count = 0
    emps =[]


    @classmethod
    def incr_count(cls):
        cls.count +=1


    @classmethod
    def add_emp(cls,obj):
        cls.emps.append(obj)

    @staticmethod


    def __init__(self,id,name,email,contact):
        self.id = id  #instancce variable
        self.name = name
        self.email = email
        self.contact = contact
        Emp.incr_count()
        Emp.add_emp(self)



    def set_dept(self,dept_name):
        self.dept = dept_name



    def set_salary(self,sal):
        self.sal = sal

    def incr_sal(self,per):
        self.sal = self.sal + (self.sal * per)/100


    def __str__(self):
        return " {} {} {}".format(self.id,self.name,self.contact)



emp1 =  Emp(101,"abc","abc@xyz.com","0987654332")

emp2 = Emp(102,"acd","2abc@xyz.com","0333654332")

emp3 = Emp(103,"bcd","3abc@xyz.com","0333654332")

emp4 = Emp(104,"cbcd","4abc@xyz.com","0333654332")


print(Emp.count)
```

```
emp1.set_dept("HR")
emp2.set_dept("HR")
emp3.set_dept("HR")
emp4.set_dept("HR")

emp2.set_salary(10000)
emp1.set_salary(20000)
emp3.set_salary(30000)
emp4.set_salary(50000)




emp1.incr_sal(30)

emps = [emp1,emp2,emp3,emp4]

for emp in emps:
    if emp.sal > 50000 and emp.dept == "HR":
        print(emp)


print(Emp.emps)









print(emp1.__dict__)
```

```
4
[<__main__.Emp object at 0x0587B2D0>, <__main__.Emp object at 0x0587B390>, <_
_main__.Emp object at 0x0587B3B0>, <__main__.Emp object at 0x0587B3D0>]
{'id': 101, 'name': 'abc', 'email': 'abc@xyz.com', 'contact': '0987654332',
'dept': 'HR', 'sal': 26000.0}
```

In [ ]:

In [ ]:
```
course
course  id
        name
        author
        languge
        catergory



tutorial

t_id
t_name
t_content
t_course = Object of cllass course

all the tutotial of author abc

course and total numbe of course in them
```

In [82]:
```
#class coursee



class Course:
    courses = []
    def __init__(self,id,name,author,category):
        self.c_id = id
        self.c_name = name
        self.c_author = author
        self.c_category = category
        Course.courses.append(self)



class Tutorial():
    tuts = []
    def __init__(self,id,name,content,course):
        self.t_id = id
        self.t_name = name
        self.t_content = content
        self.t_course = course
        Tutorial.tuts.append(self)

def __str__(self):
        return " {} {} {} {}".format(self.id,self.name,self.author,self.catego
ry)


course1 =  Course(1,"Python","ABC","Programming")
course2 =  Course(2,"Java","ABC","Programming")
course3 =  Course(3,"ANgualr","ABC","Programming")

tut1 = Tutorial(101,"Loops","Looping smt",course1)
tut2 = Tutorial(102,"list","Methods",course2)
tut3 = Tutorial(103,"framework","design ui/ux",course3)
tut1 = Tutorial(104,"DJnago","web development",course1)


for tut in Tutorial.tuts:
    if tut.t_course.c_author =="ABC":
        print(tut.t_name,tut.t_content)

tut_count ={}

for tut in Tutorial.tuts:
        if tut.t_course.c_name in tut_count:
            tut_count[tut.t_course.c_name] +=1
        else:
            tut_count[tut.t_course.c_name] =1

print(tut_count)
```

```
Loops Looping smt
list Methods
framework design ui/ux
DJnago web development
{'Python': 2, 'Java': 1, 'ANgualr': 1}
```

In [ ]:

```
user:
    id
    firstname
    lastname
    eail
    password

profile:
    contact number
    aadhar number
    linkedin link
```

In [101]:
```
class User:
    def __init__(self,id,name,email,password):
        self.id = id
        self.name = name
        self.email = email
        self.password = password



    def get_details(self):
        return "{} {}".format(self.id,self.name)


class Admin:
    def get_details(self):
        return "aAmdmin ne kuch beheja dekho"


class Profile(User,Admin):
    def __init__(self,id,name,email,password,contact,linkedin):
        super().__init__(id,name,email,password)
        self.contact = contact
        self.linkedin = linkedin


    def get_details(self):
        name_details = super().get_details()
        return "{} {} {}".format(name_details,self.contact,self.linkedin)



p1 = Profile(1,"ABC","abc@gmail.com","19dhiadha",9480888784,"jaiprince17")

print(p1.get_details())


print(p1.__dict__)
```

```
1 ABC 9480888784 jaiprince17
{'id': 1, 'name': 'ABC', 'email': 'abc@gmail.com', 'password': '19dhiadha',
'contact': 9480888784, 'linkedin': 'jaiprince17'}
```

exception handling

In [ ]:
```
```

In [102]:
```
l = [10,20,30,40]
print(dir(__builtin__))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'Blo
ckingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProces
sError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedErro
r', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'En
vironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundErro
r', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'Impor
tError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedErro
r', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'Mem
oryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError',
'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'Pending
DeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionErro
r', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'S
topAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemE
rror', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'Unboun
dLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'Un
icodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warnin
g', 'WindowsError', 'ZeroDivisionError', '__IPYTHON__', '__build_class__', '_
_debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__',
'__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytea
rray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copy
right', 'credits', 'delattr', 'dict', 'dir', 'display', 'divmod', 'enumerat
e', 'eval', 'exec', 'filter', 'float', 'format', 'frozenset', 'get_ipython',
'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int',
'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'ma
p', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'po
w', 'print', 'property', 'range', 'repr', 'reversed', 'round', 'set', 'setatt
r', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'typ
e', 'vars', 'zip']
```

In [114]:
```
try:
    print(a)
    l = [10,20,30,40,50]
    l =l[100]
except IndexError as e:
    print("invalid index",e)
except Exception as e:
    print(e)
```

```
invalid index list index out of range
```

user define exception

In [ ]:
```

```