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

this is postgressql

relational:

oracle,msql,posgres,sqlite,db2

non relational

mongodb,cassandra

no structre to data

In [115]:
```
import psycopg2 as db

conn =db.connect(user="postgres",password="qwerty",host="localhost",database="panda",port=5432)

print(conn)
```

```
<connection object at 0x06492E90; dsn: 'user=postgres password=xxx dbname=panda host=localhost port=5432', closed: 0>
```

In [116]:
```
cur = conn.cursor()

print(cur)
```

```
<cursor object at 0x06A25608; closed: 0>
```

In [129]:
```
cur.execute("""CREATE TABLE qwe (ID INT,NAME VARCHAR(32),EMAIL VARCHAR(32))""")
```

```
---------------------------------------------------------------------------
DuplicateTable                            Traceback (most recent call last)
<ipython-input-129-4be188df7b63> in <module>
----> 1 cur.execute("""CREATE TABLE qwe (ID INT,NAME VARCHAR(32),EMAIL VARCHAR(32))""")

DuplicateTable: relation "qwe" already exists
```

In [170]:
```
conn.rollback()
```

In [164]:
```
cur.execute("""CREATE TABLE new1 (ID INT,NAME VARCHAR(32),EMAIL VARCHAR(32))""")
```

In [171]:
```
cur.execute(""" INSERT INTO qwe(ID,NAME,EMAIL) VALUES (101,"ABC","ABC@XYZ.CO
M") """)
```

```
---------------------------------------------------------------------------
UndefinedColumn                           Traceback (most recent call last)
<ipython-input-171-9f0435cf5ab7> in <module>
----> 1 cur.execute(""" INSERT INTO qwe(ID,NAME,EMAIL) VALUES (101,"ABC","ABC
@XYZ.COM") """)

UndefinedColumn: column "ABC" does not exist
LINE 1:  INSERT INTO qwe(ID,NAME,EMAIL) VALUES (101,"ABC","ABC@XYZ.C...
                                                    ^
```

In [6]:
```
import mysql.connector as my

db1 = my.connect(
  host="localhost",
  user="root",
  passwd="",
  database="db",
    port:3306
)
```

```
  File "<ipython-input-6-62f06c22849c>", line 8
    port:3306
        ^
SyntaxError: invalid syntax
```

In [4]:

```
---------------------------------------------------------------------------
ProgrammingError                          Traceback (most recent call last)
<ipython-input-4-1568d9de3158> in <module>
      4   passwd="user",
      5   database="db",
----> 6     port=3306
      7 )
      8

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\__init__.py in connect(*args, **kwargs)
    177         return CMySQLConnection(*args, **kwargs)
    178     else:
--> 179         return MySQLConnection(*args, **kwargs)
    180 Connect = connect  # pylint: disable=C0103
    181

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\connection.py in __init__(self, *args, **kwargs)
     93
     94         if len(kwargs) > 0:
---> 95             self.connect(**kwargs)
     96
     97     def _do_handshake(self):

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\abstracts.py in connect(self, **kwargs)
    714
    715         self.disconnect()
--> 716         self._open_connection()
    717         self._post_connection()
    718

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\connection.py in _open_connection(self)
    208         self._do_auth(self._user, self._password,
    209                       self._database, self._client_flags, self._chars
et_id,
--> 210                       self._ssl)
    211         self.set_converter_class(self._converter_class)
    212         if self._client_flags & ClientFlag.COMPRESS:

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\connection.py in _do_auth(self, username, password, database,
 client_flags, charset, ssl_options)
    142             auth_plugin=self._auth_plugin)
    143         self._socket.send(packet)
--> 144         self._auth_switch_request(username, password)
    145
    146         if not (client_flags & ClientFlag.CONNECT_WITH_DB) and databa
se:

c:\users\crypto\appdata\local\programs\python\python37-32\lib\site-packages\m
ysql\connector\connection.py in _auth_switch_request(self, username, passwor
d)
    175                 auth_data = self._protocol.parse_auth_more_data(packe
t)
```

```
    176         elif packet[4] == 255:
--> 177             raise errors.get_exception(packet)
    178
    179     def _get_connection(self, prtcls=None):

ProgrammingError: 1045 (28000): Access denied for user 'root'@'localhost' (us
ing password: YES)
```

In [14]:
```
import sqlite3

conn = sqlite3.connect(database = "test_db.db")

cur = conn.cursor()

cur.execute(""" CREATE TABLE IF NOT EXISTS EMP(ID INT,NAME VARCHAR2(32),EMAIL
 VARCHAR2(32))""")

cur.execute(""" INSERT INTO EMP(ID,NAME,EMAIL) VALUES (101,"ABC","ABC@XYZ.CO
M") """)
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-14-5fd61b40e8f3> in <module>
      5 cur = conn.cursor()
      6
----> 7 cur.execute(""" CREATE TABLE EMP(ID INT,NAME VARCHAR2(32),EMAIL VARCH
AR2(32))""")
      8
      9 cur.execute(""" INSERT INTO EMP(ID,NAME,EMAIL) VALUES (101,"ABC","ABC
@XYZ.COM") """)

OperationalError: table EMP already exists
```

In [36]:
```
cur.execute(""" INSERT INTO EMP(ID,NAME,EMAIL) VALUES (444,"vvv","ABC@XYZ.CO
M") """)
```

Out[36]:
```
<sqlite3.Cursor at 0x5168120>
```

In [59]:
```
p = cur.execute("Select * from EMP")
print(p.fetchall())
```

```
[(444, 'vvv', 'ABC@XYZ.COM'), (444, 'vvv', 'ABC@XYZ.COM'), (444, 'vvv', 'ABC@
XYZ.COM'), (444, 'vvv', 'ABC@XYZ.COM')]
```

In [56]:
```
cur.execute(""" UPDATE EMP SET NAME='RST' WHERE ID=101 """)
conn.commit()
```

In [32]:
```
print(next(p))
```

```
(101, 'ABC', 'ABC@XYZ.COM')
```

In [31]:
```
conn.commit()
```

In [58]:
```
cur.execute(""" DELETE FROM EMP WHERE ID=101""")
```

Out[58]:
```
<sqlite3.Cursor at 0x5168120>
```