Example on Shape and Paint Cost using OOP

- get the dimension sof the shape
- calculate area
- get color from user
- return the cost of painting the shape

In [42]:
```
from abc import ABC, abstractmethod
class Shape(ABC) :

    prices = {"red" : 10, "blue" : 20, "green" : 30, "white" : 1}  # class variable

    def __init__(self, color = None):
        self.area = 0
        self.cal_area()
        self.color = color

    @abstractmethod
    def cal_area():
        pass

    def cal_cost(self, color = None):
        color = color if color else self.color
        cost = Shape.prices.get(color, 1)
        return self.area * cost

class Circle(Shape):

    def __init__(self, radius, **kwargs):
        self.radius = radius
        super().__init__(kwargs.get("color"))   # execute the parent constructor

    def cal_area(self):
        self.area = 3.14 * (self.radius ** 2)

class Rectangle(Shape) :
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth
        super().__init__()

    def cal_area(self):
        self.area = self.length * self.breadth
```

In [39]:
```
c = Circle(10)
c.cal_cost("red")
```

Out[39]: 3140.0

In [40]:
```
c = Circle(10, color = "red")
c.cal_cost()
```

Out[40]: 3140.0

In [41]:
```
c = Circle(10, color = "red")
c.cal_cost("blue")
```

Out[41]: 6280.0

In [20]:
```
Shape()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 Shape()

TypeError: Can't instantiate abstract class Shape without an implementation for abst
ract method 'cal_area'
```

Assigning attributes at run time

In [57]:
```
class Employee:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self) :
        return f"{self.name} | {self.age}"

    @staticmethod
    def is_adult(age):
        return age>= 18

emp1 = Employee("Jane", 30)
emp2 = Employee("Jack", 25)
emp2.mob = 9876543
emps = [emp2, emp1]

for e in emps :
    print(e.name, e.age, e.mob)
```

```
Jack 25 9876543
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[57], line 20
     17 emps = [emp2, emp1]
     19 for e in emps :
---> 20     print(e.name, e.age, e.mob)

AttributeError: 'Employee' object has no attribute 'mob'
```

Example on Overlaoding using classmethod and staticmethods

In [62]:
```
class Employee:
```

```
    company = "Oracle"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self) :
        return f"{self.name} | {self.age}"

    @staticmethod
    def is_adult(age):
        print(Employee.company)
        return age>= 18

emp1 = Employee("Jane", 30)
emp2 = Employee("Jack", 25)
```

In [63]:
```
emp1.is_adult(18)
```

```
Oracle
```

Out[63]: True

In [61]:
```
Employee.is_adult(24)
```

Out[61]: True

In [70]:
```
class Employee:

    company = "Oracle"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    @classmethod
    def from_string(cls, emp_strg):
        name,age = emp_strg.split(",")
        return cls(name, int(age))

    @classmethod
    def from_dict(cls, emp_dict):
        return cls(**emp_dict)

    def __repr__(self) :
        return f"{self.name} | {self.age}"

emp1 = Employee("Jane", 30)
emp2 = Employee.from_string("Jack,25")
emp3 = Employee.from_dict({"name" : "Rosie", "age" : 35})
```

In [66]:
```
emp1
```

Out[66]: Jane | 30

In [67]:
```
emp2
```

Out[67]:
```
Jack | 25
```

In [71]:
```
emp3
```

Out[71]:
```
Rosie | 35
```

In [73]:
```
e = Employee  # class object
e
```

Out[73]:
```
__main__.Employee
```

In [74]:
```
e("anna", 25)
```

Out[74]:
```
anna | 25
```

Mutiple Inheritance

In [80]:
```
class E():
    pass
class A():
    pass
class C(A):
    pass
class B(E, A):
    pass
class D(B, C):
    pass

D.mro()
```

Out[80]:
```
[__main__.D, __main__.B, __main__.E, __main__.C, __main__.A, object]
```

Regular Expressions

Regular expressions are used for matching text patterns for searching, replacing and parsing text with complex patterns of characters.

Regexes are used for four main purposes -

- To validate if a text meets some criteria; Ex. a zip code with 6 numeric digits
- Search substrings. Ex. finding texts that ends with abc and does not contain any digits
- Search & replace everywhere the match is found within a string; Ex. search "fixed deposit" and replace with "term deposit"
- Split a string at each place the regex matches; Ex. split everywhere a @ is encountered

Raw python string

It is recommended that you use raw strings instead of regular Python strings. Raw strings begin with a prefix, r, placed before the quotes

In [81]:
```
print("ABC \n PQR")
```

```
ABC
 PQR
```

In [82]:
```
print(r"ABC \n PQR")
```

```
ABC \n PQR
```

Importing re module

In [15]:
```
import re
```

Functions in re Module

The "re" module offers functionalities that allow us to match/search/replace a string

- `re.match()` - The match only if it occurs at the beginning of the string
- `re.search()` - First occurrence of the match if there is a match anywhere in the string
- `re.findall()` - Returns a list containing all matches in the string
- `re.split()` - Returns a list where the string has been split at each match
- `re.sub()` - Replaces one or many matches with a string
- `re.finditer()` - Returns a collectable iterator yielding all non-overlapping matches

In [86]:
```
text = "Jack and Jill went up the hill"
re.match(r"Jack", text)  # starts with
```

Out[86]:
```
<re.Match object; span=(0, 4), match='Jack'>
```

In [104…
```
text = "Jack and Jill went up the hill"
re.search(r"Jill", text)  # contains
```

Out[104…
```
<re.Match object; span=(9, 13), match='Jill'>
```

In [94]:
```
obj = re.search(r"Jill", text)
type(obj)
```

Out[94]:
```
re.Match
```

In [ ]:
```
obj.span()
```

Out[ ]:
```
(9, 13)
```

In [103…
```
obj.group()
```

Out[103…
```
'Jill'
```

In [100…
```
text = "She sells sea shells on the sea shore"
re.findall(r"s", text)
```

Out[100…
```
['s', 's', 's', 's', 's', 's', 's']
```

In [105…
```
text = "She sells sea shells on the sea shore"
re.split(r" ", text)
```

Out[105…
```
['She', 'sells', 'sea', 'shells', 'on', 'the', 'sea', 'shore']
```

In [112…
```
expr = "10 + 5 * 20 - 10"
list(map(int, re.split(r" [+*-] ", expr)))
```

Out[112…
```
[10, 5, 20, 10]
```

In [113…
```
text = "She sells sea shells on the sea shore"
re.sub(r"[aeiou]", "*", text)
```

Out[113…
```
'Sh* s*lls s** sh*lls *n th* s** sh*r*'
```

Basic Characters

- `^` - Matches the expression to its right at the start of a string. It matches every such instance before each line break in the string
- `$` - Matches the expression to its left at the end of a string. It matches every such instance before each line break in the string
- `p|q` - Matches expression p or q

Character Classes

- `\w` - Matches alphanumeric characters: a-z, A-Z, 0-9 and _
- `\W` - Matches non-alphanumeric characters. Ignores a-z, A-Z, 0-9 and _
- `\d` - Matches digits: 0-9
- `\D` - Matches any non-digits
- `\s` - Matches whitespace characters, which include the \t, \n, \r, and space characters
- `\S` - Matches non-whitespace characters
- `\A` - Matches the expression to its right at the absolute start of a string (in single or multi-line mode)
- `\t` - Matches tab character
- `\Z` - Matches the expression to its left at the absolute end of a string (in single or multi-line mode)
- `\n` - Matches a newline character
- `\b` - Matches the word boundary at the start and end of a word
- `\B` - Matches where \b does not, that is, non-word boundary

Groups and Sets

- `[abc]` - Matches either a, b, or c. It does not match abc
- `[a\-z]` - Matches a, -, or z. It matches - because \ escapes it
- `[^abc]` - Adding ^ excludes any character in the set. Here, it matches characters that are NOT a, b or c
- `()` Matches the expression inside the parentheses and groups it
- `[a-zl` - Matches any alphabet from a to z
- `[a-z0-9]` - Matches characters from a to z and O to 9
- `[(+*)]` - Special characters become literal inside a set, so this matches (+ * and)
- `(?P=name)` - Matches the expression matched by an earlier group named "name"

Quantifiers

- `.` - Matches any character except newline
- `?` - Matches the expression to its left O or 1 times
- `{n}` - Matches the expression to its left n times
- `(,m)` - Matches the expression to its left up to m times
- `*` - Matches the expression to its left O or more times
- `+` - Matches the expression to its left 1 or more times
- `{n,m}` - Matches the expression to its left n to m times
- `{n, }` - Matches the expression to its left n or more times

Examples -

Ex. Extract all digits from the text

In [114…
```
text = "The stock price was 456 yesterday. Today, it rose to 564"
re.findall(r"\d", text)
```

Out[114…
```
['4', '5', '6', '5', '6', '4']
```

Ex. Extract all numbers from the text

In [115…
```
text = "The stock price was 456 yesterday. Today, it rose to 564"
re.findall(r"\d+", text)
```

Out[115…
```
['456', '564']
```

Ex. Retrive the dividend from the text

In [116…
```
text = "On 25th March, the company declared 17% dividend."
re.findall(r"\d+%", text)
```

Out[116…
```
['17%']
```

Ex. Retrieve all uppercase characters

In [123…
```
text = "Stocks like AAPL GOOGL BMW are the preferred ones"
re.findall(r"[A-Z]", text)
```

Out[123…
```
['S', 'A', 'A', 'P', 'L', 'G', 'O', 'O', 'G', 'L', 'B', 'M', 'W']
```

Ex. Retrive all stock names

In [120…
```
text = "Stocks like AAPL GOOGL BMW are the preferred ones"
re.findall(r"\b[A-Z]+\b", text)
```

Out[120…
```
['AAPL', 'GOOGL', 'BMW']
```

Ex. Retrieve the phone numbers with country code only

In [124…
```
text = "My number is 65-11223344 and 65-91919191. My other number is 44332211"
re.findall(r"\d+-\d+", text)
```

Out[124…
```
['65-11223344', '65-91919191']
```

Ex. Retrieve the phone numbers with or without country code

In [129…
```
text = "My number is 65-11223344 and 65-91919191. My other number is 44332211"
re.findall(r"\d+-\d+|\d+", text)
```

Out[129…
```
['65-11223344', '65-91919191', '44332211']
```

In [128…
```
re.findall(r"\d*-*\d+",text)
```

Out[128…
```
['65-11223344', '65-91919191', '44332211']
```

Ex. Retrieve the phone numbers without country code

In [133…
```
text = "My number is 65-11223344 and 65-91919191. My other number is 44332211"
re.findall(r"\d{3,}",text)
```

Out[133…
```
['11223344', '91919191', '44332211']
```

In [138…
```
re.findall(r"[^-]\b\d{3,}\b",text)
```

Out[138…
```
[' 44332211']
```

Ex. Retrieve the zip codes with 2 alphabets in the beginning

In [140…
```
text = "The zipcodes are AB4567, TX2323, 310120, NY1210, 734001 "
re.findall(r"[A-Z]{2}\d+", text)
```

Out[140…
```
['AB4567', 'TX2323', 'NY1210']
```

Ex. Retrieve the dates

In [141…
```
text = "Temasek Holdings was founded on 25/06/1974. It turns 47 on 25/6/2021"
re.findall(r"\d+/\d+/\d+", text)
```

Out[141…
```
['25/06/1974', '25/6/2021']
```

Ex. Retrieve the email IDs

In [144…
```
text = "Email us at contact@gobledy.com or info@info.net or tryus@python.az "
re.findall(r"\w+@\w+.\w+", text)
```

Out[144…
```
['contact@gobledy.com', 'info@info.net', 'tryus@python.az']
```

Ex. Replace values as given in the dict

In [152…
```
text = "Stocks like ORC AAPL GOOGL BMW are the preferred ones"
mapping = {"AAPL" : "APPLE", "GOOGL" : "GOOGLE"}
re.findall(r"\b[A-Z]+\b", text)
```

Out[152…
```
['ORC', 'AAPL', 'GOOGL', 'BMW']
```

In [146…
```
re.sub(r"\b[A-Z]+\b", "*", text)
```

Out[146…
```
'Stocks like * * * are the preferred ones'
```

In [157…
```
re.sub(r"\b[A-Z]+\b", lambda obj : mapping.get(obj.group(), obj.group()), text)
```

Out[157…
```
'Stocks like ORC APPLE GOOGLE BMW are the preferred ones'
```

In [156…
```
text = "Stocks like ORC AAPL GOOGL BMW are the preferred ones"
obj = re.search(r"\b[A-Z]+\b", text)
mapping.get(obj.group(), obj.group())  # returns matching value from mapping dict i
```

Out[156…
```
'ORC'
```

Ex. Replace first occurence of pattern with 1 second with 2 third with 3 and so on...

In [169…
```
class Counter :
    cnt = 0

    @staticmethod
    def get_counter(obj):
        Counter.cnt += 1
        return str(Counter.cnt)

text = "Stocks like ORC AAPL GOOGL BMW are the preferred ones"
re.sub(r"\b[A-Z]+\b", Counter.get_counter, text)
```

In [ ]:
```
counter = 0
def get_counter(obj = None):
    global counter
    counter += 1
    return str(counter)

text = "Stocks like ORC AAPL GOOGL BMW are the preferred ones"
re.sub(r"\b[A-Z]+\b", get_counter, text)
```

Walrus Operator

In [178…
```
if x := 15 > 10 :  # initialize and use in the same expression
    print("yes")
```

```
yes
```

Handling data from external sources

Introduction to OS module

In [ ]:
```
import os
```

File Source

- The key function for working with files in Python is the `open()` function.
- The `open()` function takes two parameters; filename, and mode.
- There are four different methods (modes) for opening a file:
  - "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  - "a" - Append - Opens a file for appending, creates the file if it does not exist
  - "w" - Write - Opens a file for writing, creates the file if it does not exist

Ex. Read file `customers.txt`

In [ ]:
```

```

Ex. Print numbers of lines in the file

In [ ]:
```

```

Ex. Clean data read from the file and extract information about all `Pilots`.

In [ ]:
```

```

Ex. Write names of the pilots to `pilots.txt` file

In [ ]:
```

```

Using `with` keyword to read data and write data

In [ ]:

HTTPS Requests

In [19]:
```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\vaide\appdata\local\programs\pyt
hon\python312\lib\site-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\vaide\appdata\lo
cal\programs\python\python312\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\vaide\appdata\local\programs
\python\python312\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\vaide\appdata\local\pr
ograms\python\python312\lib\site-packages (from requests) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\vaide\appdata\local\pr
ograms\python\python312\lib\site-packages (from requests) (2024.6.2)
Note: you may need to restart the kernel to use updated packages.
```

In [18]:
```
import requests
```

In [31]:
```
response = requests.get(r"http://127.0.0.1:5000/tasks")
response
```

Out[31]:
```
<Response [200]>
```

In [32]:
```
data1 = response.json()
data1
```

Out[32]:
```
{'TaskNo': [1, 2],
 'Task': ['Flask Project', 'Meeting at 3'],
 'Created_date': ['2023-09-15 10:00:25.116253', '2023-09-15 13:49:46.580811'],
 'Due_date': ['2023-09-16 00:00:00', '2023-09-17 00:00:00'],
 'Status': ['In-Progress', 'In-Progress']}
```

Ex. Converting data into tabular format

In [39]:
```
import pandas as pd
df = pd.DataFrame(data1)
df["Created_date"] = pd.to_datetime(df["Created_date"], format="mixed").dt.date
df["Due_date"] = pd.to_datetime(df["Due_date"], format="mixed").dt.date
df
```

Out[39]:

	TaskNo	Task	Created_date	Due_date	Status
0	1	Flask Project	2023-09-15	2023-09-16	In-Progress
1	2	Meeting at 3	2023-09-15	2023-09-17	In-Progress

In [40]:
```
df.to_csv("tasks1.csv", index=False)
```

In [36]:
```
import os
os.getcwd()
```

Out[36]:
```
'T:\\Material_general\\Oracle\\Oracle_Oct_24\\Classwork_sample'
```

In [ ]:
```
os.chdir(r"path")
```

Ex. Converting data into tabular format

In [41]:
```
data2 = requests.get(r"http://127.0.0.1:5000/tasks1").json()
data2
```

Out[41]:
```
[{'TaskNo': 1,
  'Task': 'Flask Project',
  'Created_date': '2023-09-15 10:00:25.116253',
  'Due_date': '2023-09-16 00:00:00',
  'Status': 'In-Progress'},
 {'TaskNo': 2,
  'Task': 'Meeting at 3',
  'Created_date': '2023-09-15 13:49:46.580811',
  'Due_date': '2023-09-17 00:00:00',
  'Status': 'In-Progress'}]
```

Ex. convert json data to df

In [48]:
```
pd.DataFrame(data2)
```

Out[48]:

	TaskNo	Task	Created_date	Due_date	Status
0	1	Flask Project	2023-09-15 10:00:25.116253	2023-09-16 00:00:00	In-Progress
1	2	Meeting at 3	2023-09-15 13:49:46.580811	2023-09-17 00:00:00	In-Progress

Ex. save data to json file

In [ ]:
```
import json
with open("data.json", "w") as file:
    json.dump(data2, file)
```

Ex. read data from json to pandas df

In [ ]:
```
pd.read_json("data.json")
```

Out[ ]:

	TaskNo	Task	Created_date	Due_date	Status
0	1	Flask Project	2023-09-15 10:00:25.116253	2023-09-16 00:00:00	In-Progress
1	2	Meeting at 3	2023-09-15 13:49:46.580811	2023-09-17 00:00:00	In-Progress

---

DataBase Source

In [ ]:
```
pip install SQLAlchemy
pip install pymysql
```

- Syntax - dialect+driver://username:password@host:port/database
- Mysql - "mysql+pymysql://root:1234@localhost:3306/onlineshopping"
- Oracle - "oracle+cx_oracle://s:t@dsn"

Data Connection

In [80]:
```
from sqlalchemy import create_engine, text
engine = create_engine(r"sqlite:///employee.sqlite3")
conn = engine.connect()
conn
```

Out[80]:
```
<sqlalchemy.engine.base.Connection at 0x26edacbddc0>
```

Select Clause

In [81]:
```
cursor = conn.execute(text("Select * from Employee"))
cursor.fetchall()
```

Out[81]:
```
[(0, 'Claire', 88962, 'Manager', 35),
 (1, 'Darrin', 67659, 'Team Lead', 26),
 (2, 'Sean', 117501, 'Manager', 36),
 (3, 'Brosina', 149957, 'Senior Manager', 44),
 (4, 'Andrew', 32212, 'Team Lead', 33),
 (5, 'Irene', 63391, 'Team Lead', 33),
 (6, 'Harold', 14438, 'Developer', 23),
 (7, 'Pete', 22445, 'Developer', 22),
 (8, 'Alejandro', 72287, 'Team Lead', 35),
 (9, 'Zuschuss', 195588, 'Managing Director', 53),
 (10, 'Ken', 17240, 'Developer', 25),
 (11, 'Sandra', 115116, 'Manager', 41),
 (12, 'Emily', 18027, 'Developer', 24),
 (13, 'Eric', 55891, 'Team Lead', 31),
 (14, 'Tracy', 109132, 'Manager', 34),
 (15, 'Matt', 83327, 'Manager', 43),
 (16, 'Gene', 22125, 'Developer', 22),
 (17, 'Steve', 29324, 'Team Lead', 29),
 (18, 'Linda', 54003, 'Team Lead', 35),
 (19, 'Ruben', 18390, 'Developer', 25),
 (20, 'Erin', 141401, 'Senior Manager', 47),
 (21, 'Odella', 19593, 'Developer', 22),
 (22, 'Patrick', 57093, 'Team Lead', 26),
 (23, 'Lena', 130556, 'Senior Manager', 52),
 (24, 'Darren', 22093, 'Developer', 25),
 (25, 'Janet', 13058, 'Developer', 25),
 (26, 'Ted', 26180, 'Team Lead', 27),
 (27, 'Kunst', 23259, 'Developer', 24),
 (28, 'Paul', 34248, 'Team Lead', 27),
 (29, 'Brendan', 27416, 'Team Lead', 30)]
```

In [ ]:
```
cursor = conn.execute(text("Select * from Employee"))
cursor.fetchone()
```

Insert - Update - Delete

In [ ]:
```
conn.execute(text("Insert into Employee values (30, 'Jack', 112233, 'Manager', 32)"
cursor = conn.execute(text("Select * from Employee"))
cursor.fetchall()
```

In [ ]:
```
conn.execute(text("Update Employee set Designation = 'TL' where Name = 'Jack'"))
cursor = conn.execute(text("Select * from Employee where Name = 'Jack'"))
cursor.fetchall()
```

In [ ]:
```
conn.execute(text("delete from Employee where Name = 'Jack'"))
cursor = conn.execute(text("Select * from Employee"))
cursor.fetchall()
```

Connecting to Database using pandas library

In [1]:
```
from sqlalchemy import create_engine
import pandas as pd
```

```
conn = create_engine(r"sqlite:///employee.sqlite3")
conn
```

Out[1]:
```
Engine(sqlite:///employee.sqlite3)
```

Ex. Read data from database as df

In [2]:
```
df = pd.read_sql("Employee", conn)
df.drop(columns=["index"], inplace=True)
df.head()
```

Out[2]:

	Name	Salary	Designation	Age
0	Claire	88962	Manager	35
1	Darrin	67659	Team Lead	26
2	Sean	117501	Manager	36
3	Brosina	149957	Senior Manager	44
4	Andrew	32212	Team Lead	33

Ex. Increase salary of all employees by 10%

In [84]:
```
df.Salary = df.Salary * 1.10
df.head()
```

Out[84]:

	Name	Salary	Designation	Age
0	Claire	97858.2	Manager	35
1	Darrin	74424.9	Team Lead	26
2	Sean	129251.1	Manager	36
3	Brosina	164952.7	Senior Manager	44
4	Andrew	35433.2	Team Lead	33

Ex. Filter the data from df

Ex. Extract all employee in age group of 25-35

In [86]:
```
df[df.Age.between(25, 35)]
```

Out[86]:

	Name	Salary	Designation	Age
0	Claire	97858.2	Manager	35
1	Darrin	74424.9	Team Lead	26
4	Andrew	35433.2	Team Lead	33
5	Irene	69730.1	Team Lead	33
8	Alejandro	79515.7	Team Lead	35
10	Ken	18964.0	Developer	25
13	Eric	61480.1	Team Lead	31
14	Tracy	120045.2	Manager	34
17	Steve	32256.4	Team Lead	29
18	Linda	59403.3	Team Lead	35
19	Ruben	20229.0	Developer	25
22	Patrick	62802.3	Team Lead	26
24	Darren	24302.3	Developer	25
25	Janet	14363.8	Developer	25
26	Ted	28798.0	Team Lead	27
28	Paul	37672.8	Team Lead	27
29	Brendan	30157.6	Team Lead	30

Ex. Extract all managers from the df

In [27]:
```
df_managers = df[df.Designation == "Manager"].reset_index(drop = True)
df_managers
```

Out[27]:

	Name	Salary	Designation	Age
0	Claire	88962	Manager	35
1	Sean	117501	Manager	36
2	Sandra	115116	Manager	41
3	Tracy	109132	Manager	34
4	Matt	83327	Manager	43

Ex. Adding new table if exists replace

In [28]:
```
df_managers.to_sql("Managers", conn, if_exists="replace", index=False)
```

Out[28]: 5

In [29]:
```
pd.read_sql("Managers", conn)
```

Out[29]:

	Name	Salary	Designation	Age
0	Claire	88962	Manager	35
1	Sean	117501	Manager	36
2	Sandra	115116	Manager	41
3	Tracy	109132	Manager	34
4	Matt	83327	Manager	43

In [30]:
```
df_managers.loc[:, "Age"]  = df_managers.loc[:, "Age"] + 2
df_managers
```

Out[30]:

	Name	Salary	Designation	Age
0	Claire	88962	Manager	37
1	Sean	117501	Manager	38
2	Sandra	115116	Manager	43
3	Tracy	109132	Manager	36
4	Matt	83327	Manager	45

Ex. Adding new table if exists append - add new rows to table

In [31]:
```
df_managers.to_sql("Managers", conn, if_exists="append", index=False)
```

Out[31]:
```
5
```

In [32]:
```
pd.read_sql("Managers", conn)
```

Out[32]:

	Name	Salary	Designation	Age
0	Claire	88962	Manager	35
1	Sean	117501	Manager	36
2	Sandra	115116	Manager	41
3	Tracy	109132	Manager	34
4	Matt	83327	Manager	43
5	Claire	88962	Manager	37
6	Sean	117501	Manager	38
7	Sandra	115116	Manager	43
8	Tracy	109132	Manager	36
9	Matt	83327	Manager	45

In [ ]:

In [ ]:

In [ ]: