INDEX :

Classes And Objects:

```python
class Railway :
    formType = "RailwayForm"
    def printData(self) :
        print(f"Name of passenger is :  {self.name}")
        print(f"Train is :  {self.train}")

    def disp(self) :
        print("This is disp function.")

    @staticmethod  # when we use this we don't need to use self as argument in function.
    def greet() :
        print("Good morning..!")

myApplication = Railway()
myApplication.name = "It's Me"
myApplication.train = "SuhelDev Express"
Railway.printData(myApplication)  # -> because of this we need to use self
myApplication.printData()  # -> easy way to use above line for same.
myApplication.disp()

myApplication.greet()
```

Output :-

```
PS D:\Tutorial Of Python> python
Name of passenger is :  It's Me
Train is :  SuhelDev Express
Name of passenger is :  It's Me
Train is :  SuhelDev Express
This is disp function.
Good morning..!
PS D:\Tutorial Of Python>
```

Constructor:

```python
class Employee :
    company = "Google"

    def __init__(self, name, salary) :
        self.name = name
        self.salary = salary
        print("Constructor called..!")

    def getDetails(self) :
        print("This coming from a function")
        print(f"Name of the company is :  {self.company}")
        print(f"The name of the employee is :  {self.name}")
        print(f"The salary of the employee is :  {self.salary}")

ItsMe = Employee("Aman", 1500000)
ItsMe.getDetails()
```

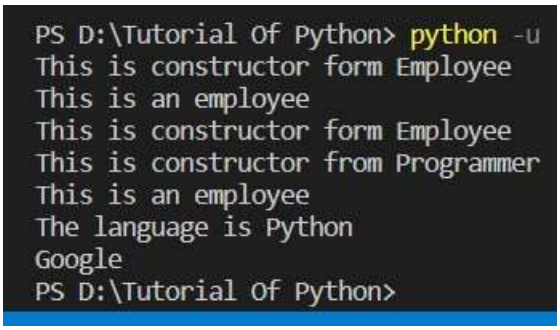
Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial
Constructor called..!
This coming from a function
Name of the company is :  Google
The name of the employee is :  Aman
The salary of the employee is :  1500000
PS D:\Tutorial Of Python>
```

Inheritance :

```
class Employee :
    company = "Google"

    def __init__(self) :
        print("This is constructor form Employee")

    def showDetails(self) :
        print("This is an employee")

class Programmer(Employee) :
    language = "Python"
    # company = "Micro Soft" We can over-ride the variables of the base class form
child class.

    def __init__(self) :
        super().__init__()  # 'super' keyword use to call base constructor or
function.
        print("This is constructor from Programmer")

    def getLanguage(self) :
        print(f"The language is {self.language}")

e = Employee()
e.showDetails()

p = Programmer()
p.showDetails()
p.getLanguage()
print(p.company)
```

Output :-


PS D:\Tutorial Of Python> python -u
This is constructor form Employee
This is an employee
This is constructor form Employee
This is constructor from Programmer
This is an employee
The language is Python
Google
PS D:\Tutorial Of Python>

ClassMethod :

```
class Employee :
    company = "Camel"
    salary = 10000
    location = "Delhi"

    # def changeSal(self, sal) :
        # self.__class__.salary = sal  # -> going to change class attribute, this
is alternate method

    @classmethod
    def changeSal(cls, sal) :
        cls.salary = sal
    # classmethod -> is a method which is bound to the class not to the object of
the class.

e = Employee()
print("Salary of the Employee is :  ", Employee.salary)

e.changeSal(40000)
print("Now the salay becomes change", e.salary)
print("This is from class attribute :  ", Employee.salary)

e1 = Employee()
print(Employee.salary)
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial
Salary of the Employee is :   10000
Now the salay becomes change 40000
This is from class attribute :   40000
PS D:\Tutorial Of Python>
```

Property Dicorator :

```
class Employee :
    company = "Hindustan Petroleum"
    salary = 5600
    salBonus = 500

    @property
    def totalSal(self) :  # -> acts as a property of the class.
        return self.salary + self.salBonus

    @totalSal.setter  # -> to set the totalSal, and it will be called
automatically whenever it needs
    def totalSal(self, val) :  # -> name of function should be same as of the
property name.
        self.salBonus = val - self.salary

e = Employee()
print("Printing total salary by the use of property :  ", e.totalSal)
# print(e.totalSal())  # -> cann't write as like this as this method acts as a
property.

print("As 'totalSal' acts as a property of the given class so we can make changes
into it.")
e.totalSal = 6000
print(e.salary)
print(e.salBonus)
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\Q_PropertyDecorator.py"
Printing total salary by the use of property :   6100
As 'totalSal' acts as a property of the given class so we can make changes into it.
5600
400
PS D:\Tutorial Of Python>
```

Operator Overloding :

```
class NumOperation :
    def __init__(self, num) :
        self.num = num

    def __add__(self, obj) :  # -> this is dunder method
        print("Lets add")
        return self.num + obj.num

    def __mul__(self, obj) :
        print("Lets multiply")
        return self.num * obj.num

    def __str__(self) : # if we don't use this method print(obj) prints address of
the object.
        return f"Decimal Number :  {self.num}"

n1 = NumOperation(6)
n2 = NumOperation(5)

print(n1)

sum = n1 + n2
print("Sum of the given number is :  ", sum)

print("Multiply of the given number is :  ", n1 * n2)
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial
Decimal Number :  6
Lets add
Sum of the given number is :   11
Lets multiply
Multiply of the given number is :   30
PS D:\Tutorial Of Python>
```

Try And Catch Clause :

```
while(True) :
    print("Enter 'q' to exit ")
    a = input("Enter a number :  ")

    if a == 'q' :
        break

    try :
        print("Before any exception this line(within try block) alway going to
execute")
        a = int(a)

        if(a > 10) :
            print("You entered a number greater than 10")
        else :
            print("You entered a number smaller than 10")
    except Exception as e :
        print(f"Error occur :  {e}")
        print("Re-enter a valid input")


try :
    a = int(input("Enter a number :  "))
    b = 100/a
    print(b)
except ValueError as e :  # -> for incorrect input
    print(f"Exception 1 occured :  {e}")
    print("Re-enter the correct value")

except ZeroDivisionError as e :  # -> for arithmetical error
    print(f"Exception 2 occured :  {e}")
    print("Cann't divided by the 0")
```

```python
# Raising custom error -> We can raise custom exceptions using the raise keyword
in python.

def increment(num) :
    try :
        return int(num) + 1
    except :
        raise ValueError("This is not good, give number carefully..!")

c = increment(364)
print("The value after performing the operation :  ", c)

try :
    c = increment('add364')
    print(c)
except Exception as e :
    print(e)


print("This is end of this program.\nThank you..!")
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\T_TryAndExcept
Enter 'q' to exit
Enter a number :  23
Before any exception this line(within try block) alway going to execute
You entered a number greater than 10
Enter 'q' to exit
Enter a number :  0
Before any exception this line(within try block) alway going to execute
Enter 'q' to exit
Enter a number :  q
Enter a number :  123
0.8130081300813008
This is end of this program.
Thank you..!
PS D:\Tutorial Of Python>
```

TryElse And Finally :

```
try :
    i = int(input("Enter a number :  "))
    print("After operation value will be :  ", 100/i)

except Exception as e :
    print(f"Exception occur :  {e}")
    exit()

else :
    print("Exception doesn't occur that why I'm printing this else clause.")

# Python offers a finally clause which ensures execution of a piece of code while
exception occurs.
finally :
    print("finally clause run in any condition however the execution of code also
exited.")

print("This will not going to run when any exception occurs.")
```

Output :-

```
PS D:\Tutorial Of Python> python -u "d:\Tutorial Of Python\U_TryElseFinallyClause.py"
Enter a number :  23
After operation value will be :   4.3478260869565215
Exception doesn't occur that why I'm printing this else clause.
finally clause run in any condition however the execution of code also exited.
This will not going to run when any exception occurs.
PS D:\Tutorial Of Python>
```