Asmita Porwal
Batch-1
Day-7
29/1/2024
Data engineering

Assignment-7

```
#variables
x = 5          # x is a variable with the value 5
name = "John" # name is a variable with the value "John"
pi = 3.14      # pi is a variable with the value 3.14
is_true = True # is_true is a variable with the value True

#arithmetic operators
a = 10
b = 3

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
modulo = a % b           # Modulo (remainder after division)
exponentiation = a ** b # Exponentiation
floor_division = a // b # Floor division (returns the quotient, discarding
any remainder)
print(addition,subtraction,multiplication,division,modulo,exponentiation,f
loor_division)

#comparison operators
x = 5
y = 10

equal = x == y
not_equal = x != y
greater_than = x > y
less_than = x < y
greater_equal = x >= y
less_equal = x <= y
print(equal,not_equal,greater_than,less_than,greater_equal,less_equal)
```

```python
#logical operators
p = True
q = False

logical_and = p and q  # Logical AND
logical_or = p or q    # Logical OR
logical_not = not p    # Logical NOT
print(logical_and,logical_not,logical_or)

#assignment operators
x = 5

x += 3
print(x)
x -= 2
print(x)
x *= 4
print(x)
x /= 2
print(x)
```

Output

```
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEn
13 7 30 3.3333333333333335 1 1000 3
False True False True False True
False False True
8
6
24
12.0
PS D:\DataEngineeringhexa\Python>
```

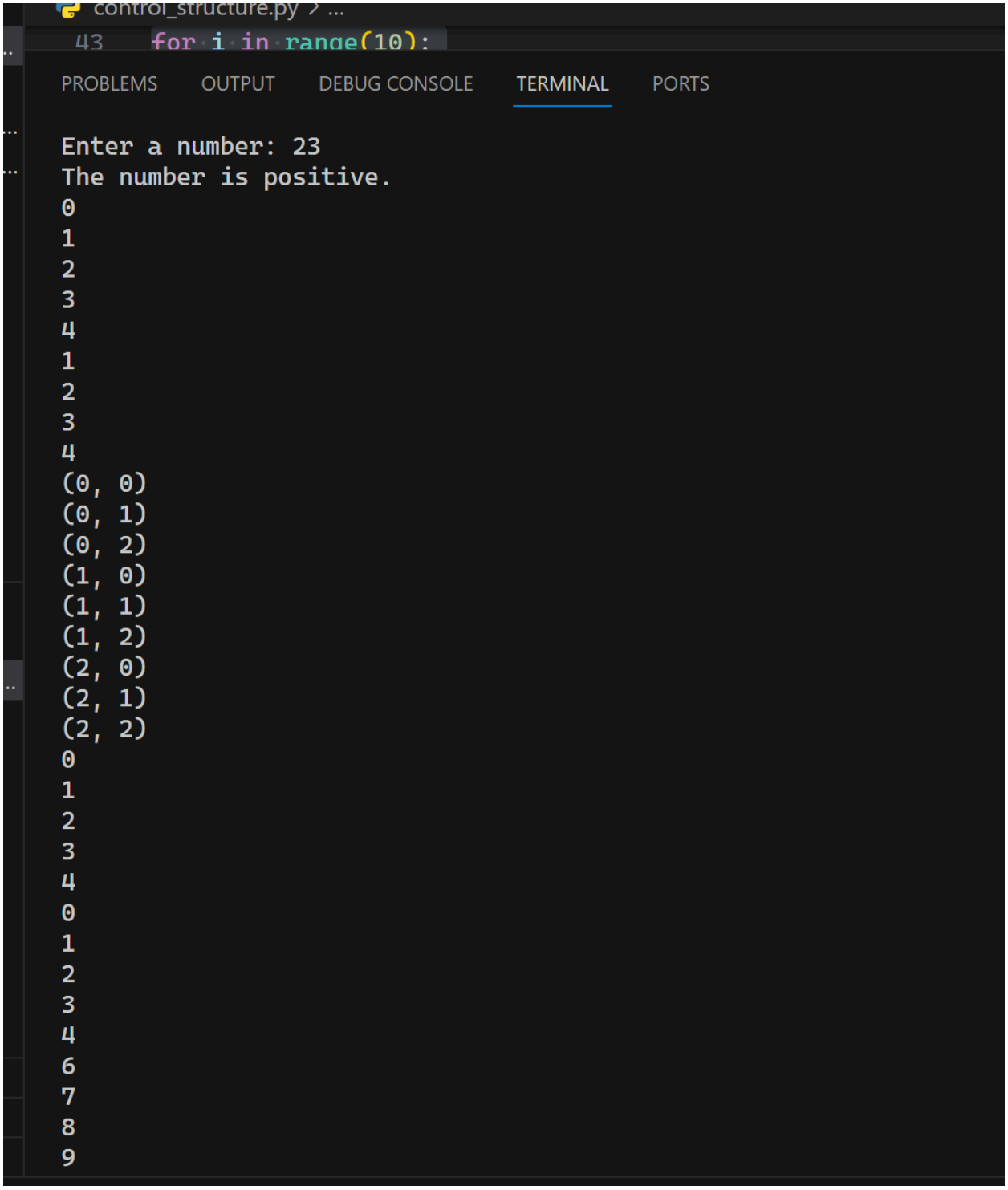
Control structure

```python
# if-else
```

```python
def check_number(number):
    if number > 0:
        print("The number is positive.")
    elif number < 0:
        print("The number is negative.")
    else:
        print("The number is zero.")

if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    check_number(user_input)


# for loop

for i in range(5):
    print(i)

# while
j=1
while j<5:
    print(j)
    j +=1

# Nested loop
for i in range(3):
    for j in range(3):
        print(f"({i}, {j})")

# break,continue and pass

for i in range(10):
    if i == 5:
        break
    print(i)

for i in range(10):
    if i == 5:
        continue
    print(i)
```

```
for i in range(10):
   pass

# input and output

name = input("Enter your name: ")
print("Hello, " + name + "!")
```

Output


control_structure.py > ...
43 for i in range(10):
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Enter a number: 23
The number is positive.
0
1
2
3
4
1
2
3
4
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)
0
1
2
3
4
0
1
2
3
4
6
7
8
9

```
3
4
6
7
8
9
Enter your name: asmita
Hello, asmita!
```

List

```
l = [1,2,3,4,5,3]
l.append(6)
print(l)
list=[7,8]
l.extend(list)
print(l)
l.insert(2,10) # insert at position 2 value 10
print(l)
l.remove(10)
print(l)
pop_element=l.pop(1) #pop(position of a value to be removed)
print("poped elemnt: ",pop_element)
print(l)
count = l.count(3)
print(count)
l.sort()
print(l)
l.reverse()
print(l)
```

Output

```
PS D:\DataEngineeringhexa\Python> python -u "
[1, 2, 3, 4, 5, 3, 6]
[1, 2, 3, 4, 5, 3, 6, 7, 8]
[1, 2, 10, 3, 4, 5, 3, 6, 7, 8]
[1, 2, 3, 4, 5, 3, 6, 7, 8]
poped elemnt:  2
[1, 3, 4, 5, 3, 6, 7, 8]
2
[1, 3, 3, 4, 5, 6, 7, 8]
[8, 7, 6, 5, 4, 3, 3, 1]
PS D:\DataEngineeringhexa\Python>
```

List Slicing

```python
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# Basic slicing
slice1 = my_list[2:5]  # Elements from index 2 to 4 (5 is exclusive)
print(slice1)

# Omitting start or stop
slice2 = my_list[:5]   # Elements from the beginning to index 4
slice3 = my_list[5:]   # Elements from index 5 to the end
print(slice2)
print(slice3)

# Negative indices
slice4 = my_list[-4:-1]  # Elements from the 4th last to the 2nd last
print(slice4)

# Slicing with step
slice5 = my_list[1:8:2]  # Elements from index 1 to 7 with a step of 2
print(slice5)
```

```
# Omitting start, stop, and using step
slice6 = my_list[::2]    # Every second element from the beginning to the
end
print(slice6)

# Reversing a list
reversed_list = my_list[::-1]
print(reversed_list)
```

Output

```
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEngineeringhexa\P
[2, 3, 4]
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[6, 7, 8]
[1, 3, 5, 7]
[0, 2, 4, 6, 8]
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
PS D:\DataEngineeringhexa\Python>
```

Dictionary

```
#dictionary
my_dict = {
    "name": "John",
    "age": 25,
    "city": "New York"
}

print(my_dict["name"])
print(my_dict["age"])
print(my_dict["city"])

#get method
age = my_dict.get("age")
print(age)

salary = my_dict.get("salary", 0)
print(salary)
```

```python
#key and values method
all_keys = my_dict.keys()
print(all_keys)

all_values = my_dict.values()
print(all_values)

#all items
all_items = my_dict.items()
print(all_items)

#update
new_data = {"salary": 50000, "gender": "Male"}
my_dict.update(new_data)
print(my_dict)

#pop specific item
removed_value = my_dict.pop("age")
print(removed_value)

#pop item
last_item = my_dict.popitem()
print(last_item)
```

Output

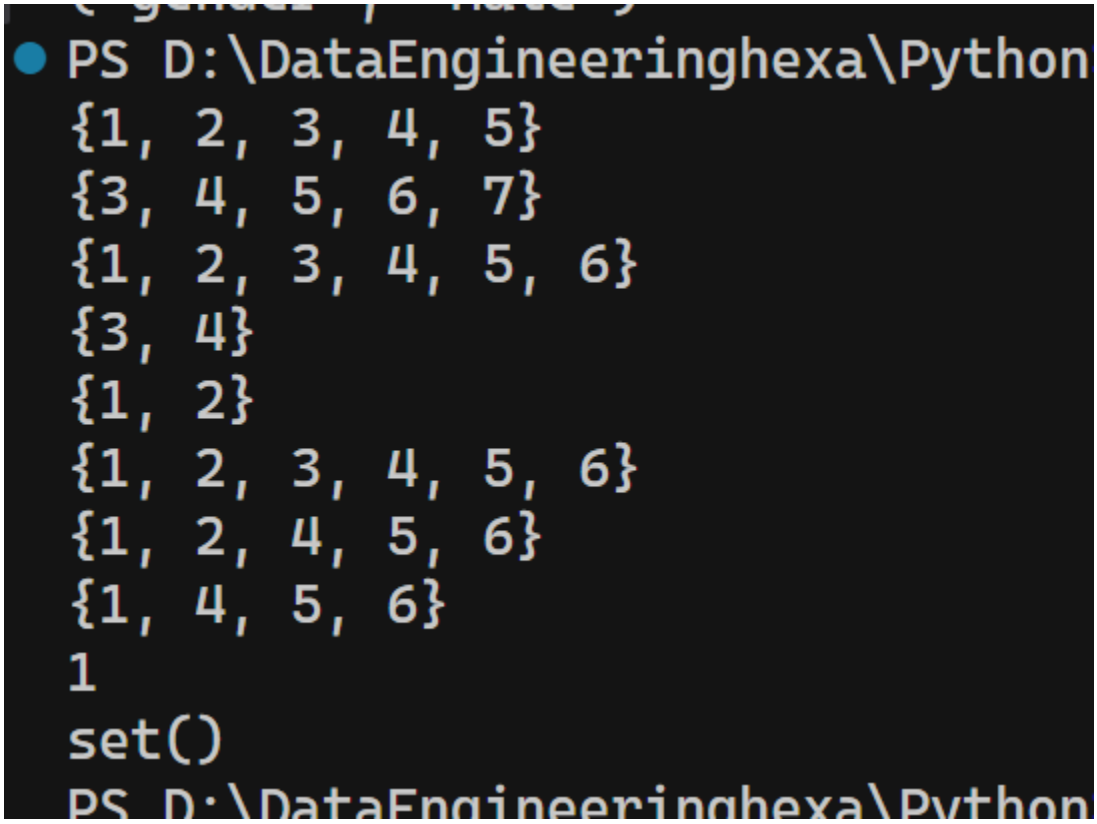
```
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEngineeringhexa\Python\tempCodeRunn
John
25
New York
25
0
dict_keys(['name', 'age', 'city'])
dict_values(['John', 25, 'New York'])
dict_items([('name', 'John'), ('age', 25), ('city', 'New York')])
{'name': 'John', 'age': 25, 'city': 'New York', 'salary': 50000, 'gender': 'Male'}
25
('gender', 'Male')
```

Sets

```python
# Creating a set
my_set = {1, 2, 3, 4, 5}
print(my_set)

# Using the set() constructor
another_set = set([3, 4, 5, 6, 7])
print(another_set)

# Set operations
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

union_set = set1 | set2  # Union
print(union_set)

intersection_set = set1 & set2  # Intersection
print(intersection_set)

difference_set = set1 - set2  # Difference
print(difference_set)

#add method
my_set.add(6)
print(my_set)

#remove  Removes the specified element from the set. Raises a KeyError if the element is not present.
my_set.remove(3)
print(my_set)

#discard  Removes the specified element from the set if present. Does not raise an error if the element is not found
my_set.discard(2)
print(my_set)

#pop(): Removes and returns an arbitrary element from the set. Raises a KeyError if the set is empty.
popped_element = my_set.pop()
print(popped_element)
```

```
#clear(): Removes all elements from the set, making it empty
my_set.clear()
print(my_set)
```

Output

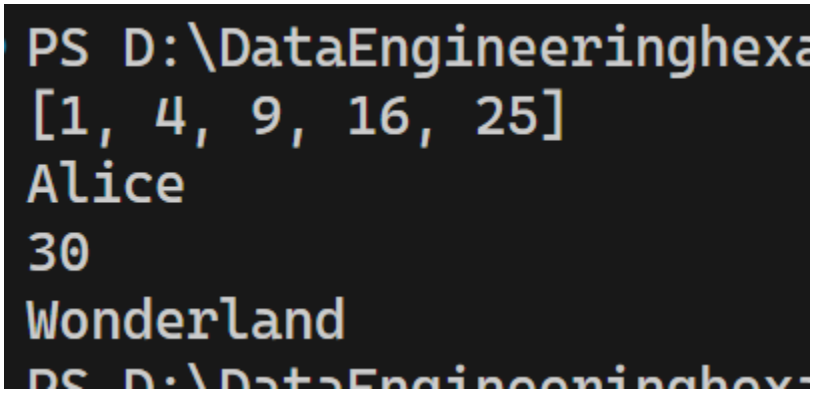


Map

```
#"map" can refer to both a function (map() function) and a data structure
(dict type).

#Map as a Function
numbers = [1, 2, 3, 4, 5]
squared_numbers = map(lambda x: x**2, numbers)
print(list(squared_numbers))

#A dict (dictionary) in Python is often referred to as a map. It is an
unordered collection of key-value pairs.
```

```python
#dictionary (map)
my_map = {
    "name": "Alice",
    "age": 30,
    "city": "Wonderland"
}


print(my_map["name"])
print(my_map["age"])
print(my_map["city"])
```

Output



Functions

```python
#Mapping function
#A mapping function typically refers to a function that transforms or maps elements from one set to another. In Python, the map() function is commonly used for this purpose. It applies a specified function to all items in an input iterable and returns an iterable of the results.


numbers = [1, 2, 3, 4, 5]
squared_numbers = map(lambda x: x**2, numbers)
print(list(squared_numbers))
```

```python
#String Function
#String functions in Python are methods that operate on strings.

my_string = "Hello, World!"
print(my_string.upper())
print(my_string.find("World")) #it will give the position
print(my_string.replace("Hello", "Hi"))

#Number Function


x = 5
print(abs(x))
print(pow(x, 2))      #power
print(round(3.14159))

#Date and Time Function
from datetime import datetime, timedelta

current_date = datetime.now()
print(current_date)
future_date = current_date + timedelta(days=7)
print(future_date.strftime("%Y-%m-%d"))

#Python Functions
#Functions in Python are blocks of reusable code that perform a specific
task. They are defined using the def keyword.

def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))

#Default Argument Values
#Python functions can have default values for their parameters. If a value
is not provided for a parameter, the default value is used.
```

```python
def greet(name="Guest"):
    return f"Hello, {name}!"

print(greet())
print(greet("Alice"))

#Keyword Arguments
#You can pass arguments to a function by specifying the parameter names
along with the values. This is known as using keyword arguments.

def greet(greeting, name):
    return f"{greeting}, {name}!"

print(greet(name="Alice", greeting="Hi"))

#Special parameters
#Special parameters in Python functions include *args (arbitrary argument
lists) and **kwargs (arbitrary keyword argument dictionaries).

def my_function(*args, **kwargs):
    print(args)
    print(kwargs)

my_function(1, 2, 3, name="Alice", age=30)

#Arbitrary Argument Lists
#Arbitrary argument lists allow a function to accept any number of
positional arguments.

def sum_all(*args):
    return sum(args)

result = sum_all(1, 2, 3, 4)
print(result)

#Lambda Expressions
#Lambda expressions are a concise way to create anonymous functions
(functions without a name).

square = lambda x: x**2
```

```
print(square(4))
```

Output

```
PS D:\DataEngineeringhexa\Python> python -u "d:
[1, 4, 9, 16, 25]
HELLO, WORLD!
7
Hi, World!
5
25
3
2024-01-29 23:25:19.411333
2024-02-05
Hello, Alice!
Hello, Guest!
Hello, Alice!
Hi, Alice!
(1, 2, 3)
{'name': 'Alice', 'age': 30}
10
16
```

OOPS

```
#OOPS
#Object-Oriented Programming is a programming paradigm that uses objects
to organize code. Objects can encapsulate data and behavior, providing a
modular and structured approach to software development.

#Class and Object
#Class: A class is a blueprint for creating objects. It defines attributes
and methods that the objects of the class will have.
```

```python
#Object: An object is an instance of a class. It represents a real-world entity and has attributes and behaviors defined by the class.

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

    def bark(self):
        print("Woof!")

my_dog = Dog("Buddy", 3)
print(my_dog.name)
my_dog.bark()


#Access Specifiers
#Access specifiers in Python determine the visibility of attributes and methods within a class.
#Public: Attributes and methods are accessible from outside the class.
#Private: Attributes and methods are accessible only within the class.

class MyClass:
    def __init__(self):
        self.public_variable = "I am public"
        self.__private_variable = "I am private"

    def public_method(self):
        print("Public method")

    def get_private_variable(self):
        return self.__private_variable


new_class=MyClass()
print(new_class.public_variable)
print(new_class.public_method())
p=new_class.get_private_variable()
print(p)
```

```python
#Constructor
#A constructor is a special method that gets executed when an object is created. In Python, the constructor is named __init__.
class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model

my_car = Car("Toyota", "Camry")

#Inheritance
#Inheritance allows a class (subclass or derived class) to inherit properties and behaviors from another class (base class or superclass).
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):
    def bark(self):
        print("Dog barks")

my_dog = Dog()
my_dog.speak()
my_dog.bark()

#example 2
class Bird:
    def __init__(self,name):
        self.name=name

    def print_info(self):
        print("The bird name is : ",self.name)

    def fly(self):
        print("The bird can fly ")

class Parrot(Bird):
    def __init__(self, name,color,character):
        super().__init__(name)
```

```
        self.color=color
        self.character=character

   #override method

    def print_info(self):
        print("The bird is :",self.name)
        print("color of bird is :",self.color)
        print("charater of bird is :",self.character)

obj_parrot=Parrot('Parrot','Green','good')
obj_parrot.fly()
obj_parrot.print_info()

#types of inheritance
#Single inheritance
#In single inheritance you can derive a (child) class from a single parent
class.
class father:
    def quality(self):
        print("inside father class")
        print("father has intelligence and  deep thinking power")
        print("\n")
class son(father):
    def aim(self):
        print("inside son class")
        print("child wants to be software enginner")
        print("\n")
ram=son()
ram.quality()
ram.aim()

#Multilevel inheritance
#Python allows multilevel inheritance. In it a new derived class inherits
the properties of the base class. Actually a class is permitted to inherit
from a base class or child class or derived class. So, the classes are
inherited at multiple individual levels.
class grandfather:
    def gf_quality(self):
        print("inside gf class")
```

```
        print("gf was a honest person")
        print("\n")
wazed=grandfather()
class father(grandfather):
    def father_quality(self):
        print("inside father class")
        print("father has intelligence and  deep thinking power")
        print("\n")

fazul=father()

class son(father):
    def aim(self):
        print("inside son class")
        print("child wants to be software enginner")
        print("\n")
ram=son()
ram.gf_quality()
ram.father_quality()
ram.aim()

#Multiple Inheritance
#In it you can inherit the features of more classes into a single class.
Suppose we have two parents classes (Father and Mother) and one child
class that is derived form two parents classes.
class father():
    def father_quality(self):
        print("inside father class")
        print("father has intelligence and  deep thinking power")

    def father_nature(self):
        print("inside father class")
        print("father is strict in principle")
        print("\n")

fazul=father()
class Mother():
    def mother_quality(self):
        print("inside mother class")
        print("mother is a good cook")
```

```
    def mother_nature(self):
        print("inside mother class")
        print("mother has soft mind")
        print("\n")

fazul=father()

class son(father,Mother):
    def aim(self):
        print("inside son class")
        print("child wants to be software enginner")
        print("\n")
ram=son()
ram.mother_quality()
ram.father_quality()
ram.aim()

#Hierarchical inheritance
#In it you can drive multiple classes from a single base class. In the
following illustration we have one patent class and two derived classes.
class father():
    def father_quality(self):
        print("inside father class")
        print("father has intelligence and  deep thinking power")

    def father_nature(self):
        print("inside father class")
        print("father is strict in principle")
        print("\n")
class son1(father):
    def aim(self):
        print("inside son1 class")
        print("child wants to be software enginner")
        print("\n")

ram=son1()
ram.father_nature()
ram.father_quality()
ram.aim()
```

```python
class son2(father):
    def aim(self):
        print("inside son2 class")
        print("child wants to be a cook ")
        print("\n")
shyam=son2()
shyam.father_quality()

#Polymorphism
#Polymorphism allows objects of different types to be treated as objects
of a common type. It can be achieved through method overloading or method
overriding.
class Cat(Animal):
    def speak(self):
        print("Cat meows")

def animal_sound(animal):
    animal.speak()

my_cat = Cat()
animal_sound(my_dog)
animal_sound(my_cat)

#Method Overriding
#Method overriding occurs when a subclass provides a specific
implementation for a method that is already defined in its superclass.
class Bird(Animal):
    def speak(self):
        print("Bird sings")

my_bird = Bird()
my_bird.speak()

#File handling
#File handling in Python involves reading from and writing to files. The
open() function is commonly used for file operations.
# Writing to a file
with open("example.txt", "w") as file:
    file.write("Hello, Asmita!")
```

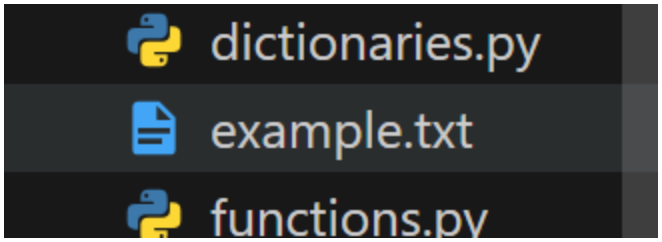
```python
# Reading from a file
with open("example.txt", "r") as file:
    content = file.read()
    print(content)

#Exception Handling
#Exception handling in Python involves using try, except, else, and
finally blocks to handle errors and exceptions gracefully.
#Exceptions: Exceptions are raised when the program is syntactically
correct, but the code results in an error. This error does not stop the
execution of the program, however, it changes the normal flow of the
program.
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
else:
    print(result)
finally:
    print("Execution completed")


#Encapsulation
class Base:
    def __init__(self):
        self.a = "HexaforHexa"
        self.__c = "HexaforHexa"

class Derived(Base):
    def __init__(self):

        # Calling constructor of
        # Base class
        Base.__init__(self)
        print("Calling private member of base class: ")
        # print(self.__c)
obj1 = Base()
obj2 = Derived()
print(obj1.a)
```

Output

```
Buddy
Woof!
I am public
Public method
None
I am private
Animal speaks
Dog barks
The bird can fly
The bird is : Parrot
color of bird is : Green
charater of bird is : good
inside father class
father has intelligence and  deep thinking power


inside son class
child wants to be software enginner


inside gf class
gf was a honest person


inside father class
father has intelligence and  deep thinking power
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
inside mother class
mother is a good cook
inside father class
father has intelligence and  deep thinking power
inside son class
child wants to be software enginner


inside father class
father is strict in principle


inside father class
father has intelligence and  deep thinking power
inside son1 class
child wants to be software enginner


inside father class
father has intelligence and  deep thinking power
Animal speaks
Cat meows
Bird sings
Hello, Asmita!
Cannot divide by zero
Execution completed
Calling private member of base class:
HexaforHexa
PS D:\DataEngineeringhexa\Python>
```



Modules

My_module.py

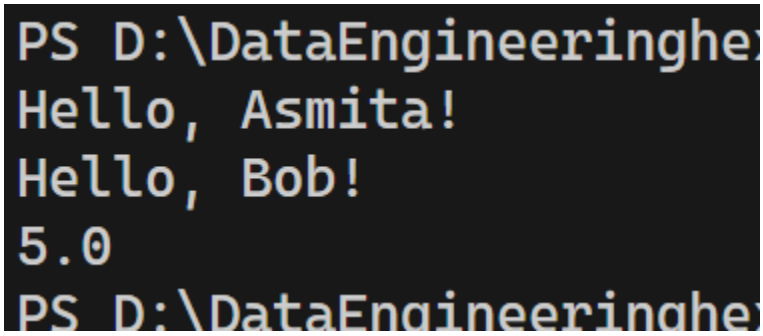
```
def greet(name):
    print(f"Hello, {name}!")
```

Module.py

```
#Python Modules

#User-Defined Modules:
#we can create our own modules by saving Python code in a separate file
and then importing it into our script.
import my_module
my_module.greet("Asmita")

#Executing modules as scripts
#You can use the if __name__ == "__main__": block to check whether the
Python script is being run as the main program or if it is being imported
as a module.
def greet(name):
    print(f"Hello, {name}!")

if __name__ == "__main__":
    # Code to run if the module is executed as a script
    greet("Bob")

#Standard Modules
#Python comes with a set of standard modules that provide additional
functionality. You can use them by importing them into your script.
```

```python
import math

result = math.sqrt(25)
print(result)

#Packages
#A package is a way of organizing related modules into a single directory hierarchy. Packages are created by placing multiple module files in a directory that includes a special file called __init__.py.

#Importing * From a Package
#When importing all modules from a package using *, the __init__.py file can specify what gets imported when the package is imported.


#Intra-package References
#You can reference other modules within the same package using relative imports.

#Packages in Multiple Directories
#Python allows you to organize packages across multiple directories. Each directory containing packages must also include an __init__.py file.
```

Output



CSV

```python
# import csv
# rows = []
# with open("new_data1.csv", 'r') as file:
#     csvreader = csv.reader(file)
```

```
#     header = next(csvreader)
#     for row in csvreader:
#         rows.append(row)
# print(header)
# print(rows)

with open('new_data1.csv') as file:
    content = file.readlines()
header = content[:1]
rows = content[3:]
print(header)
print(rows)
```

Output

```
PS D:\DataEngineeringhexa\Python> python -u "d:\DataEngineeringhexa\Python\tempCodeRunnerFile.py"
[',TransactionId,Payer,Payee,Amount,original_txn_amount,loopids,loop_desc\n']
['2,3,E2,E4,9158140,9158140,1,E1>E6>E2>E4>E3>E1\n', '3,4,E4,E3,6186113,6186113,1,E1>E6>E2>E4>E3>E1\n', '4,5,E3,E
1,15812515,15812515,1,E1>E6>E2>E4>E3>E1\n', '5,6,E7,E6,24798610,24798610,2,E7>E6>E7\n', '6,7,E6,E7,15963196,1596
3196,2,E7>E6>E7\n', '7,8,E2,E1,9242500,9242500,3,E2>E1>E5>E2\n', '8,9,E1,E5,20206869,20206869,3,E2>E1>E5>E2\n',
'9,10,E5,E2,11517817,11517817,3,E2>E1>E5>E2\n', '10,11,NLE4,NLE6,14073349,14073349,-1,\n', '11,12,NLE3,NLE3,1142
3200,11423200,-1,\n', '12,13,NLE3,NLE5,3524889,3524889,-1,\n', '13,14,NLE4,NLE3,11683597,11683597,-1,\n', '14,15
,NLE2,NLE5,5501292,5501292,-1,\n', '15,16,NLE7,NLE1,13697935,13697935,-1,\n', '16,17,NLE3,NLE5,22489733,22489733
,-1,\n', '17,18,NLE6,NLE7,7060216,7060216,-1,\n', '18,19,NLE4,NLE1,3504225,3504225,-1,\n', '19,20,NLE4,NLE7,2039
9154,20399154,-1,\n', '20,21,NLE5,NLE7,18727613,18727613,-1,\n', '21,22,NLE3,NLE1,18108278,18108278,-1,\n', '22,
23,NLE2,NLE2,1867012,1867012,-1,\n', '23,24,NLE3,NLE7,16123724,16123724,-1,\n', '24,25,NLE5,NLE4,2814296,2814296
,-1,\n', '25,26,NLE6,NLE3,10311884,10311884,-1,\n', '26,27,NLE5,NLE5,16189197,16189197,-1,\n', '27,28,NLE1,NLE6,
14700007,14700007,-1,\n', '28,29,NLE7,NLE1,22424835,22424835,-1,\n', '29,30,NLE6,NLE7,6944264,6944264,-1,\n', '3
0,31,NLE5,NLE7,7431724,7431724,-1,\n', '31,32,NLE6,NLE5,1772112,1772112,-1,\n', '32,33,NLE6,NLE5,16494787,164947
87,-1,\n', '33,34,NLE6,NLE1,6182666,6182666,-1,\n', '34,35,NLE1,NLE7,5270681,5270681,-1,\n', '35,36,NLE1,NLE5,20
904296,20904296,-1,\n', '36,37,NLE5,NLE7,17534515,17534515,-1,\n', '37,38,NLE4,NLE2,11459873,11459873,-1,\n', '3
8,39,NLE7,NLE1,22224062,22224062,-1,\n', '39,40,NLE1,NLE1,21613079,21613079,-1,\n', '40,41,NLE7,NLE1,7420149,742
0149,-1,\n', '41,42,NLE6,NLE7,11811239,11811239,-1,\n', '42,43,NLE2,NLE3,9883096,9883096,-1,\n', '43,44,NLE5,NLE
6,23403845,23403845,-1,\n', '44,45,NLE1,NLE6,21348063,21348063,-1,\n', '45,46,NLE5,NLE4,13901929,13901929,-1,\n'
, '46,47,NLE1,NLE3,24731104,24731104,-1,\n', '47,48,NLE1,NLE3,20620617,20620617,-1,\n', '48,49,NLE2,NLE7,1864085
5,18640855,-1,\n', '49,50,NLE2,NLE4,7824290,7824290,-1,\n', '50,51,NLE2,NLE4,23318860,23318860,-1,\n', '51,52,NL
E4,NLE2,6300149,6300149,-1,\n', '52,53,NLE2,NLE5,6450438,6450438,-1,\n', '53,54,NLE4,NLE5,9007433,9007433,-1,\n'
, '54,55,NLE4,NLE7,11536390,11536390,-1,\n', '55,56,NLE6,NLE7,4925029,4925029,-1,\n', '56,57,NLE5,NLE1,1741840,1
741840,-1,\n', '57,58,NLE5,NLE4,13431148,13431148,-1,\n', '58,59,NLE7,NLE7,24321291,24321291,-1,\n', '59,60,NLE5
,NLE7,16948523,16948523,-1,\n', '60,61,NLE3,NLE1,14564881,14564881,-1,\n', '61,62,NLE4,NLE6,12018167,12018167,-1
,\n', '62,63,NLE2,NLE1,23194701,23194701,-1,\n', '63,64,NLE1,NLE4,13714630,13714630,-1,\n', '64,65,NLE6,NLE6,208
72043,20872043,-1,\n', '65,66,NLE4,NLE3,2251143,2251143,-1,\n', '66,67,NLE4,NLE6,19754744,19754744,-1,\n', '67,6
8,NLE1,NLE3,9980477,9980477,-1,\n', '68,69,NLE7,NLE4,20647876,20647876,-1,\n', '69,70,NLE5,NLE1,2808610,2808610,
-1,\n']
```