Jamia Millia Islamia

Dept. of Computer Science

Subject: Pattern Matching Using Python

Submitted By: Wasit Shafi
Roll No: 18MCA054
Date: 15-NOV-2020

Submitted To: Prof.Usman Malik

Section A

a) State one difference between variable and constant.

Sol) Variables name can be writtern in combination of lowercase/uppercase/underscores while all constants are written in all capital letters and underscores separating the words.

b) Define Destructors.

Sol) destructor is used to destroy the object and perform the final clean up(free memorry).
It can be defined as :

```
def __del__(self):
    print ("Object destroyed");
```

d) What are user defined data types?

Sol) A user-defined data type (UDT) is a data type that derived from an existing data type. We can use UDTs to extend the built-in types already available and create your own customized data types.

e) Define preprocessor directives.

Sol) Preprocessing directives are lines in your program that start with # (In C/C++). The # is followed by an identifier that is the directive name. For example, #define is the directive that defines a macro. Whitespace is also allowed before and after the #.Since python is an interpreter, there's no preprocessing step to be applied.

f) "Abstract class may be instantiated". This statement is true or false?

Sol) False

g) Define scope resolution operator.

Sol)

h) What are static functions?

Sol) A static function is a member function of a class that can be called even when an object of the class is not initialized. A static function cannot access any variable of its class except for static variables.

i) Define polymorphism

Sol) The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Real life example of polymorphism: A person at the same time can have

different characteristic. Like a man at the same time is a father, a husband, an employee.

j) Can a constructor be overloaded?

Sol) Yes, can overload constructors in a class.

k) Write the use of break statement?

Sol) In Python, break can alter the flow of a normal loop. Loops iterate over a block of code until the test expression is false, but sometimes we wish to terminate the current the whole loop without checking test expression. The break statement is used in these cases.

l) Write one difference between constructor and destructor.

Sol) Constructor helps to initialize the object of a class while as destructor is used to destroy the instances.

Section B

Note: Short answer type questions.

i) Write short notes on:

a) Encapsulation b) Data hiding

Sol)

a) Encapsulation: Encapsulation refers to the bundling of data, along with the methods that operate on that data, into a single unit.Encapsulation may also refer to a mechanism of restricting the direct access to some components of an object, such that users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

b) Data hiding : It technique of hiding internal object details i.e. data members. Data hiding guarantees restricted data access to class members & maintain object integrity.

ii) What are the advantages and disadvantages of object oriented programming approach.

Sol)

Advantages:

1. It models real world well.
2. Programs are easy to understand.
3. It offers classes reusability.
4. It facilitates quick Development as parallel development of classes is possible.

5. Program are easier to test manage and maintain.

Disadvantages:

1. In OOPs, classes tend be overly generalized.
2. The relation among classes become artificial at times.
3. The OOP programs design is tricky.
4. One needs to do proper planning and proper design for OOP Programming.
5. To Program with OOP, Programmer need proper skills such as design skills programming skills, thinking in term of objects etc.

iii) How is static binding different from dynamic binding?

Sol)

1. Static binding happens during compilations while dynamic binding happens at run time.
2. Method overloading is example of static binding while method overriding is example of dynamic binding.
3. Private, static and final methods shows static binding because they cannot be overrinden while other trhan private, static and final methods shows dynamic binding because they can be overrriden.

iv) Explain private inheritance using an example.

Sol)

v) What are inline functions? What are the advantages?

Sol) Python lambda functions, also known as anonymous functions, are inline functions that do not have a name. They are created with the lambda keyword. This is part of the functional paradigm built-in Python.Python lambda functions are restricted to a single expression. They can be used wherever normal functions can be used.

Advantages:

The lambda keyword in Python provides a shortcut for declaring small anonymous functions.

Lambda is a tool for building anonymous functions.

Lambda is a tool for building callback handlers.

vi) Write a program in PYTHON illustrating the concept of for loop.

Sol)

Syntax :

```
for val in sequence:
    Body of for
```

Program to print 0 to 9:

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

vii) Explain virtual base class?

Sol)

viii) Describe parametrised constructor with an example.

Sol) When we declare a constructor in the way that accepts the arguments during the object creation these type of constructors are called the Parameterized Constructors. It is used to initialize the instance members of the object.

```
class Student:
    def __init__(self,name,ids,college):
        self.name=name
        self.ids=ids
        self.college=college

    def Display_Details(self):
        print("Student Details:")
        print("Student Name:",self.name)
        print("Student ids:",self.ids)
        print("Student college:",self.college)

student=Student("Yash",2023,"Kcc")
student.Display_Details()
```

What is the difference between a union and a structure.

Sol)

x) Differentiate between constructor and destructor.

Sol)

	constructor	**destructor**
1	A constructor is used to initialize objects of a class.	Destructor destroys the objects when they are no longer needed.
2	A constructor is called when object is created.	Destructor is called when instance of a class is deleted or released.
3	Constructor may or may not have any arguments.	Destructor can not have any arguments.
4	A constructor allocates memory.	Destructor releases the memory.

5	A constructor cannot be declared virtual.	A destructor can be virtual
6	Overloading of constructor is possible.	Overloading of Destructor is not possible.

xi) Write a program in PYTHON that opens and closes a file.

Sol)

```
file_name = 'test.txt'
file = open(file_name, 'r') # Opening a file
print(file.readlines())
file.close()                # Closing a file
```

Section C

Q.1 Write a class named box with data member width of the box and member function setwidth to set the width of the box and a non member function printwidth to print thewidth of box . printwidth function is the friend of setwidth function. Use necessaryobjects and access them.

Sol)

Q.2 Write a python program to find area of circle,triangle and rectangle.Use functionoverloading concept.

Sol)

Method 1 :

```
import math

def area(x = None, y = None, z = None):
  if x != None and y != None and z != None:
    s = (x + y + z) / 2
    return math.sqrt(s * (s - x) * (s - y) * (s - z))
  elif x != None and y != None:
    return x * y
  elif x != None:
    return math.pi * x**2

print("Area of circle with radius 5 is : ", area(5))
print("Area of rectangle with length 5 & width 4 is : ",
area(5, 4))
print("Area of triangle with sides 2, 3 & 4 is : ",
area(2, 3 ,4))
```

Method 2 :

```
import math
from multipledispatch import dispatch

@dispatch(int)
def area(radius):
  return math.pi * radius ** 2

@dispatch(int, int)
def area(length, width):
  return length * width

@dispatch(int, int, int)
def area(s1, s2, s3):
  s = (s1 + s2 + s3) / 2
  return math.sqrt(s * (s - s1) * (s - s2) * (s - s3))

print("Area of circle with radius 5 is : ", area(5))
print("Area of rectangle with length 5 & width 4 is : ",
area(5, 4))
print("Area of triangle with sides 2, 3 & 4 is : ",
area(2, 3 ,4))
```

Q.3 What do you understand by object oriented programming? Compare procedure oriented programming and object oriented programming.

Sol) Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects. It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects. There are many object-oriented programming languages including JavaScript, C++, Java, and Python. Object-oriented programming is based on several principles such as Encapsulation, Abstraction, Polymorphism, Inheritance etc.

	Procedure Oriented	Object Oriented
1	In procedural programming, program is divided into small parts called functions.	In object oriented programming, program is divided into small parts called objects.
2	Procedural programming follows top down approach.	Object oriented programming follows bottom up approach.
3	There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
4	Adding new data and function is not easy.	Adding new data and function is easy.

5	Procedural programming does not have any proper way for hiding data so it is less secure.	Object oriented programming provides data hiding so it is more secure.
6	In procedural programming, overloading is not possible.	Overloading is possible in object oriented programming.
7	In procedural programming, function is more important than data.	In object oriented programming, data is more important than function.
8	Procedural programming is based on unreal world.	Object oriented programming is based on real world.
9	Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

Q.4 Write a program in PYTHON to multiply two matrices.

Sol)

```
# Program to multiply 2 matrices
# 3x3 matrix
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
    [6,7,3,0],
    [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]

# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
       # iterate through rows of Y
       for k in range(len(Y)):
           result[i][j] += X[i][k] * Y[k][j]

for r in result:
   print(r)
```

Q.5 What are the differences between formal and actual parameters. Explain with example.

Sol)

Actual Parameters are the values that are passed to the function when it is invoked while Formal Parameters are the variables defined by the function that receives

values when the function is called. Actual parameters are values that are passed to a function when it is invoked.

The formal parameter is enclosed in parenthesis. The list consists variable names and data types of all the necessary values for the method. Each formal parameter is separated by a comma. When method is not accepting any input values, then the method should have an empty set of parenthesis after method name.Formal parameters are the variables defined by the function that receives values when the function is called. The variable x and y are not the actual parameters. They are copies of the actual parameters. They are known as formal parameters. These variables are only accessible within the method. After printing the addition of two numbers, the control is returned back to the main program.Refer the below program.

```
def addition(x, y)
    addition = x+y
    print(addition)

addition(2, 3)
addition(4, 5)
```

Here value (2 , 3) and (4,5) are the actual parameters which are passed to the method addition, and the sum of two numbers will display on the screen and (x, y) are formal parameters.

Q.6 What are the various types of control structures in PYTHON? Explain any 3 with suitable examples.

Sol)

1. Sequential: It is the default mode where sequential execution of code statements (one line after another) occurs.

```
print ("hello")
print("world")
```

2. Selection: It is used for decisions, branching eg. choosing between 2 or more alternative paths. In Python, these are the types of selection statements:

```
if
if-else
if-elif-else(nested if-else)
```

3. Repetition: It is used for looping, i.e. repeating a piece of code multiple times in a row. In Python, there are 2 types of loops:

```
1. while
2. for
```

Q.7 What is abstract function? Explain the need of abstract function with an example.

Sol)

A class is called an Abstract class if it contains one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and its abstract methods must be implemented by its subclasses.

By defining an abstract functions(abstract base class), we can define a common Application Program Interface(API) for a set of subclasses. This capability is especially useful in situations where a third-party is going to provide implementations, such as with plugins, but can also help you when working in a large team or with a large code-base where keeping all classes in your mind is difficult or not possible.

Section D - OUTPUT Based Questions:

Q1.

```
# Python program showing abstract base class work
from abc import ABC, abstractmethod

class Polygon(ABC):
  # abstract method
  def noofsides(self):
    pass

class Triangle(Polygon):
  # overriding abstract method
  def noofsides(self):
    print("I have 3 sides")

class Pentagon(Polygon):
  # overriding abstract method
  def noofsides(self):
    print("I have 5 sides")

class Hexagon(Polygon):
  # overriding abstract method
  def noofsides(self):
    print("I have 6 sides")
```

```
class Quadrilateral(Polygon):
  # overriding abstract method
  def noofsides(self):
    print("I have 4 sides")

# Driver code
R = Triangle()
R.noofsides()
K = Quadrilateral()
K.noofsides()
R = Pentagon()
R.noofsides()
K = Hexagon()
K.noofsides()
```

Sol 1)
Output :
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides

Q2.

```
# Python program showing abstract base class work
from abc import ABC, abstractmethod

class Animal(ABC):
  def move(self):
    pass

class Human(Animal):
  def move(self):
    print("I can walk and run")

class Snake(Animal):
  def move(self):
    print("I can crawl")
```

```
class Dog(Animal):
  def move(self):
    print("I can bark")

class Lion(Animal):
  def move(self):
    print("I can roar")

# Driver code
R = Human()
R.move()
K = Snake()
K.move()
R = Dog()
R.move()
K = Lion()
K.move()
```

Sol 2)
Output :
I can walk and run
I can crawl
I can bark
I can roar

Q3.

```
# Python program showing implementation of
abstract class through subclassing
import abc

class parent:
  def geeks(self):
    pass

class child(parent):
  def geeks(self):
    print("child class")

# Driver code
```

```
print(issubclass(child, parent))
print(isinstance(child(), parent))
```

Sol 3)
Output :
I can walk and run
I can crawl
I can bark
I can roar

Q4.

```
# Python program invoking a method using super()
import abc
from abc import ABC

class R(ABC):
  def rk(self):
    print("Abstract Base Class")

class K(R):
  def rk(self):
    super().rk()
    print("subclass ")

# Driver code
r = K()
r.rk()
```

Sol 4)
Output :
Abstract Base Class
subclass

Q5.

```
# Python program showing abstract class cannot be
an instantiation
from abc import ABC,abstractmethod

class Animal(ABC):
```

```
  @abstractmethod
  def move(self):
    pass

class Human(Animal):
  def move(self):
    print("I can walk and run")

class Snake(Animal):
  def move(self):
    print("I can crawl")

class Dog(Animal):
  def move(self):
    print("I can bark")

class Lion(Animal):
  def move(self):
    print("I can roar")

c=Human()
c.move()
print(dir(ABC))
```

Sol 5)
Output :
I can walk and run
['__abstractmethods__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__slots__', '__str__', '__subclasshook__', '_abc_impl']

Q6.

```
# Python program showing abstract class cannot be
an instantiation
from abc import ABC, abstractmethod

class Animal(ABC):
```

```
  @abstractmethod
  def move(self):
    pass

class Human(Animal):
  def move(self):
    print("I can walk and run")

class Snake(Animal):
  def move(self):
    print("I can crawl")

class Dog(Animal):
  def move(self):
    print("I can bark")

class Lion(Animal):
  def move(self):
    print("I can roar")

c=Animal()
```

Sol 6)
Output :
Error, Since Animal class is a abstract class so it can't instantiated.