ASSIGNMENT ON CLASSES

Q1. Classes are needed to solve problems which functions cannot. (T/F)

Ans. False

Q2. Define what is an ‘object’ in few words.

Ans. Instance of a class

Q3. Constructor sets the initial state of an object. (T/F)

Ans. True

Q4. A class variable has same value for all the objects instantiated from the class. (T/F)

Ans. True

Q5. Inside class definition, a method can call the attributes and other methods using \_\_\_\_\_ keyword.

Ans. Self

Q6. ‘global’ keyword is always needed to access outside variables inside class definition. (T/F)

Ans. False

Q7. In Python, super() cannot be used in child class inheriting from multiple parents. (T/F)

Ans. False

Q8. In Python, what is the correct way to develop a class called Person that has parameters in the initialize function called name, age, and sex?

a) class initialize:

def \_\_Person\_\_(self, name, age, sex):

self.name = name

self.age = age

self.sex = sex

b) class Person{

def \_\_initialize\_\_(self, name, age, sex):

self.name = name

self.age = age

self.sex = sex

}

c) class Person:

def \_\_init\_\_(self, name, age, sex):

self.name = name

self.age = age

self.sex = sex

d) class def Person: (self, name, age, sex):

self.name = name

self.age = age

self.sex = sex

Q9. Write a Python class named Rectangle constructed by a length and width and method which can compute area of a rectangle.

class Rectangle():

def \_\_init\_\_(self, l, w):

self.length = l

self.width = w

def calculate\_area(self):

return self.length\*self.width

newRectangle = Rectangle(12, 10)

print(newRectangle. calculate\_area())

Q10. A class named Square inherits from the above class Rectangle. It takes only one input in constructor and over-writes the area method.

class Square(Rectangle):

def \_\_init\_\_(self, s):

super().\_\_init\_\_(s,s)

def calculate\_area(self):

return self.length\*self.length

newSquare = Square(12)

print(newSquare.calculate\_area())

Q11.Write a Python class named Board which is constructed with number of rows and columns. It has methods which can

1. print the board,
2. update a position with a character and
3. return list of values in the row, column, diagonal containing a given point.

Here is the sample behaviour,

board = new Board(4,5) # initializes a board with 4 rows and 5 columns having ‘0’ in all positions. Board has positions from (1,1), (1,2), …, (4,4), (4,5).

board.display() # prints the board

# [[‘0’,’0’,’0’,’0’,’0’],

# [‘0’,’0’,’0’,’0’,’0’],

# [‘0’,’0’,’0’,’0’,’0’],

# [‘0’,’0’,’0’,’0’,’0’]]

board.update((1,4),’X’) # updates the position (1,4) as ‘X’

board.display()

# [[‘0’,’0’,’0’,’X’,’0’],

# [‘0’,’0’,’0’,’0’,’0’],

# [‘0’,’0’,’0’,’0’,’0’],

# [‘0’,’0’,’0’,’0’,’0’]]

rowVals = board.returnVals((2,5),’-’) # returns a list of values in row containing position (2,5)

print(rowVals)

# [‘0’,’0’,’0’,’0’,’0’]

columnVals = board.returnVals((2,5),’|’) # returns a list of values in column containing position (2,5)

print(columnVals)

# [‘0’,’0’,’0’,’0’]

lrDiagVals = board.returnVals((2,5),’\’) # returns a list of values in left-to-right diagonal containing position (2,5)

print(lrDiagVals)

# [‘X’,’0’]

rlDiagVals = board.returnVals((2,5),’/’) # returns a list of values in right-to-left diagonal containing position (2,5)

print(rlDiagVals)

# [‘0’,’0’,’0’]

Thank You. Hare Krishna.

**ARCHIVE:**

def inc(a,b=1):

return(a+b)

a=inc(1)

a=inc(a,a)

print(a)

a) 3

b) 4

c) 2

d) It will error out

Q2.

Which alternative code is logically equivalent to the code below?

max=x if (x>y) else y

a) if (x>=y):

max=x

elif:

max=y

b) max=y if (x>y): max=x

c) if (y>x): y else: x

d) if (x>y):

max=y

elif (x==y):

max=y

else:

max=x

Q3.