NADAR HR.SEC.SCHOOL, RAJAPALAYAM.

XII - COMPUTER SCIENCE – ENGLISH MEDIUM

QUARTERLY EXAM 2024 ANSWER KEY

PART - A

1.	Which of the following defines what an object can do?	**Interface**
2.	The data type whose representation is unknown are called	**Abstract datatype**
3.	The members that are accessible from within the class and are also available to its sub- classes is called	**Protected members**
4.	The algorithm tha yields expected output for a valid input is called as	**Algorithmic solution**
5.	Which of the following character is used to give comments in Python Program ?	#
6.	Which amongst this is not a jump statement ?	**for**
7.	A Function which calls itself is called as	**Recursion**
8.	In which arguments the correct positional order is passed to a function?	Required
9.	Strings in python:	**Immutable**
10.	What is stride?	**third argument of slice operation**
11.	Let list1=[2,4,6,8,10], then print(List1[-2]) will result in	**8**
12.	Which of the following Python function can be used to add more than one element within an existing list?	**extend()**
13.	Which of the following method is used as destructor?	**_ _del_ _()**
14.	The process of creating an object is called as:	**Instantiation**
15.	The ____Part of the while loop is optional	**else**

PART - B

16)	**What is a subroutine?** (i) Subroutines are the basic building blocks of computer programs. Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly. (ii) In Programming languages these subroutines are called as Functions.
17)	**Differentiate constructors and selectors.** (i) Constructors are functions that build the abstract data type. (ii) Selectors are functions that retrieve information from the data type.
18)	Python prescribes a convention of prefixing the name of the variable/method with single or double underscore to emulate the behaviour of protected and private access specifiers.
19)	Analysis of algorithms and performance evaluation can be divided into two different phases: **1. A Priori estimates:** This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors. **2. A Posteriori testing:** This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.
20)	**What are the different operators that can be used in Python ?** The operators that can be used in Python (i) Arithmetic operators (ii) Relational or Comparative operator (iii) Logical operators (iv) Assignment operators (v) Conditional operator

21)	**Write note on break statement.** (i) The **break** statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. (ii) When the **break** statement is executed, the control flow of the program comes out of the loop and starts executing the segment of code after the loop structure. (iii) If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.
22)	**What are the main advantages of function?** Main advantages of functions are (i) It avoids repetition and makes high degree of code reusing. (iij It provides better modularity for your application.
23)	**How will you delete a string in Python?** Python will not allow deleting a particular character in a string. Whereas you can remove entire string variable using **del** command.
24)	**Write the syntax of creating a Tuple with n number of elements.** **Syntax:** Tuples_Name=(e1,e2,e3…….en) Tuples_Name=e1,e2,e3…….en Tuples_Name=(e1,)
	PART - C
25)	**Interface** The return value of the pure functions solely depends on its arguments passed. Hence, if you call the pure functions with the same set of arguments, you will always get the same return values. They do not have any side effects. They do not modify the arguments which are passed to them **Implementation** The return value of the impure functions does not solely depend on its arguments passed. Hence, if you call the impure functions with the same set of arguments, you might get the different return values. For example, random(), Date(). They may modify the arguments which are passed to them
26)	**Define Global scope with an example.** (i) A variable which is declared outside of all the functions in a program is known as Global variable. (ii) This means, global variable can be accessed inside or outside of all the functions in a program. Consider the following example
27)	**List the characteristics of an algorithm.** (i) Input (ii) Output (iii) Finiteness iv) Definiteness (v) Effectiveness (vi) Correctness (vii) Simplicity (viii) Unambiguous (ix) Feasibility (x) Portable (xi) Independent
28)	**Explain Ternary operator with examples.** (i) Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. (ii) It simply allows testing a condition in a single line replacing the multiline if-else making the code compact. Variable Name = [on_true] if [Test expression] else [on_false] **(iii) Example:** min = 50 if 49 < 50 else 70

29)	**Program :**
	for i in range(65, 70):
	  for j in range(65, i+1):
	  print(chr(j), end='\t')
	  print('\n')

30) **Write the rules of local variable.**

Rules of local variable :
(i) A variable with local scope can be accessed only within the function/block that it is created in.
ii) When a variable is created inside the function/block, the variable becomes local to it.
(iii) A local variable only exists while the function is executing.
(iv) The format arguments are also local to function.

31) **Write a note about count( ) function in python.**

Syntax	Description	Example

count (str, beg, end)	Returns the number of substrings occurs within the given range. Remember that substring may be a single character. Range (beg and end) arguments are optional. If it is not given, python searched in whole string. Search is case sensitive.	>>> str1="Raja Raja Chozhan"
		>>> print(str1.count('Raja'))
		2
		>>> print(str1.count('r'))
		0
		>>> print(str1.count('R'))
		2
		>>> print(str1.count('a'))
		5
		>>> print(str1.count('a',0,5))
		2
		>>> print(str1.count('a',11))
		1

32) **How do define constructor and destructor in Python?**

def __init__(self, [args ……..]):
<statements>

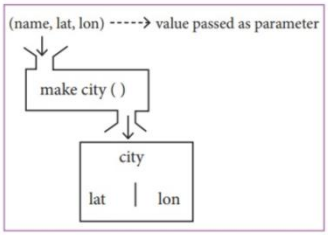
def __del__(self):
<statements>

33) **What will be the output of the following code?**
list = [2x for x in range(5)]**
print(list)
Output: [1, 2,4, 8,16]

PART - D

34) A) **1. What are called Parameters and write a note on**
(i) Parameter without Type (ii) Parameter with Type

Parameters (and arguments): Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.
(i) Parameter without Type: Let us see an example of a function, definition:
(requires: b>=0)
(returns: a to the power of b)
let rec pow a b:=
if b=0 then 1
*else'a * pow a (b -1)*
* In the above function definition variable 'b' is the parameter and the value which is passed to the variable 'b' is the argument. The precondition **(requires)** and postcondition **(returns)** of the function is given.
* Note we have not mentioned any types: **(data types).** Some language compiler solves this type **(data type)** inference problem algorithmically, but some require the type to be mentioned.
* In the above function definition if expression can return 1 in the then branch, by the **typing** rule the entire if expression has type **int**.
* Since the if expression has type **'int',** the function's return type also be 'int'**. 'b'** is compared to 0 with the equality operator, so **'b'** is also a type of 'int'.* Since 'a' is multiplied with another expression using the * operator, **'a'** must be an int.
(ii) Parameter with Type : Now let us write the same function definition with types for some reason:
(requires: b> 0)
(returns: a to the power of b)

	let rec pow (a: int) (b: int): int := *if b=0 then 1* *else a * pow b (a-1)* * When we write the type annotations for **'a'** and **'b'** the parentheses are mandatory. Generally we can leave out these annotations, because it's simpler to let the compiler infer them. * There are times we may want to explicitly write down types. This is useful on times when you get a type error from the compiler that doesn't make sense. Explicitly annotating the types can help with debugging such an error message.
34) B)	**How will you facilitate data abstraction. Explain it with suitable example** To facilitate data abstraction, you will need to create two types of functions: constructors and selectors. **Constructors and Selectors:** (i) Constructors are functions that build the abstract data type. Selectors are functions that retrieve information from the data type. (ii) For example, say you have an abstract data type called city. This city object will hold the city's name, and its latitude and longitude. To create a city object, you'd use a function like *city = makecity (name, lat, Ion)* (iii) To extract the information of a city object, you would use functions like *getname(city)* *getlat(eity)* *getlon(city)* (iv) The following pseudo code will compute the distance between two city objects: *distance(cityl, city2):* *ltl, lgl := getlat(cityl), getlon(cityl)* *lt2, lg2 := getlat(city2), getlon(city2)* *return ((ltl - lt2)**2 +* *(lgl -lg2)**2)t/2* (v) In the above code read distance(), getlat() and getlon() as functions and read It as latitude and lg longitude. Read := as "assigned as" or "becomes" (vi) ltl, lgl := getlat(cityl), getlon(cityl) is read as ltl becomes the value of getlat(cityl) and lgl becomes the value of getlont(cityl). (vii) Notice that you don't need to know how these functions were implemented. You are assuming that someone else has defined them for us. (viii) It's okay if the end user doesn't know how functions were implemented. However, the functions still have to be defined by someone. (ix) Let us identify the constructors and selectors in the above code. As you already know that Constructors are functions that build the abstract data type. In the above pseudo code the function which creates the object of the city is the constructor. city = makecity (name, lat, Ion) (x) Here makecity (name, lat, Ion) is the constructor which creates the object city. *Constructor* Selectors are nothing but the functions that retrieve information from the data type. Therefore in the above code *getname(city) getlat(city) getlon(city)* (xi) are the selectors because these functions extract the information of the city object. 

35) A) **Explain the types of scopes for variable or LEGB rule with example.**

There are 4 types of Variable Scope, let's discuss them one by one:

Local Scope:

Local scope refers to variables defined in current function. Always, a function will first look up for a variable name in its local scope. Only if it does not find it there, the outer

Global Scope:

(i) A variable which is declared outside of all the functions in a program is known as global variable.

(ii) This means, global variable can be accessed inside or outside of all the functions in a program.

(iii) On execution of the above code the variable 'a' which is defined inside the function displays the value 7 for the function call DispO and then it displays 10, because a is defined in global scope.

Enclosed Scope:

(i) All programming languages permit functions to be nested. A function (method) with in another function is called nested function.

(ii) A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.

(iii) When a compiler or interpreter search for a variable in a program, it first search Local, and then search Enclosing scopes.

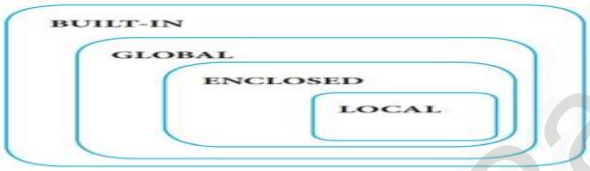
(iv) In the above example Displ() is defined with in Disp(). The variable a' defined in Disp() can be even used by Displ () because it is also a member of Disp().

Built-in Scope:

(i) Finally, we discuss about the widest scope. The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter.

(ii) Any variable or module which is defined in the library functions of a programming language has Built-in or module scope. They are loaded as soon as the library files are imported to the program.

Local(L)	Defined inside function/class
Enclosed(E)	Defined inside enclosing functions (Nested function concept)
Global(G)	Defined at the uppermost level
Built-in (B)	Reserved names in built-in functions (modules)



35) B) **What is Binary search? Discuss with example.**

Binary search: Binary search also called half¬interval search algorithm. It finds the position of a search element within a sorted array. The binary search algorithm can be done as divide- and-conquer search algorithm and executes in logarithmic time.

Pseudo code for Binary search :

Start with the middle element:

(i) If the search element is equal to the middle element of the array i.e., the middle value = number of elements in array/2, then return the index of the middle element.

(ii) If not, then compare the middle element with the search value,

(iii) If the search element is greater than the number in the middle index, then select the elements to the right side of the middle index, and go to Step-1.

(iv) If the search element is less than the number in the middle index, then select the elements to the left side of the middle index, and start with Step-1.

(v) When a match is found, display success message with the index of the element matched.

(vi) If no match is found Tor all comparisons, then display unsuccessful message.

Binary Search Working principles:

(i) List of elements in an array must be sorted first for Binary search. The following example describes the step by step operation of binary search.

(ii) Consider the following array of elements, the array is being sorted so it enables to do the binary search algorithm. Let us assume that the search element is 60 and we need to search the location or index of search element 60 using binary search.

36) A) **Explain input() and print() functions with examples.**

Input and Output Functions: A program needs to interact with the user to accomplish the desired task; this can be achieved using Input- Output functions. The input() function helps to enter data at run time by the user and the output function print() is used to display the result of the program on the screen after execution.

The input() function :

(i) In Python, input( ) function is used to accept data as input at run time. The syntax for input() function is,

Variable = input ("prompt string")

(ii) Where, prompt string in the syntax is a statement or message to the user, to know what input can be given.

(ii) If a prompt string is used, it is displayed on the monitor; the user can provide expected data from the input device. The input( ) takes whatever is typed from the keyboard and stores the entered data in the given variable.

(iv) If prompt string is not given in input() no message is displayed on the screen, thus, the user will not know what is to be typed as input.

>>> city=input ("Enter Your City:")
Enter Your City: Madurai

The print() function :

- The print() function is used to display result on the screen.
- The **print** () evaluates the expression before printing it on the monitor.
- The print () displays an entire statement which is specified within print ().
- **Comma** (,) is used as a separator in **print ()** to print more than one item.

The syntax for print() is as follows :

print ("string to be displayed as output ")
print (variable)
print ("String to be displayed as output ", variable)
print ("Stringl ", variable, "String 2", variable, "String 3" .....)

Example :

>>> print ("Welcome to Python Programming")
Welcome to Python Programming

36) B) **Write a detail note on if..else..elif statement with suitable example.**

(i) When we need to construct a chain of if statement(s) then **'elif** clause can be used instead of **'else'.**

(ii) Syntax :

```
if<condition-1>:
        statements-block 1
elif<condition-2>:
        statements-block 2
else:
        statements-block n
```

(iii) In the syntax of **if..elif..else** mentioned above, condition-1 is tested if it is true then statements-block 1 is executed, otherwise the control checks condition-2, if it is true statements-block2 is executed and even if it fails statements-block n mentioned in **else** part is executed.

(iv) **'elif** clause combines **if..else-if..else** statements to one **if..elif...else, 'elif** can be considered to be abbreviation of'**else if**. In an '**if** statement there is no limit of'elif clause that can be used, but an **'else'** clause if used should be placed at the end.

```
If a>b and a>c:
        Print("a is big")
elif b>c:
            Print("b is big")
else:
        Print("c is big")
```

37) A)

3. Explain the following built-in functions.

(a) id() (b) chr() (c) round() (d) type() (e) pow()

Function	Description	Syntax	Example
id ( )	id( ) Return the "identity" of an object. i.e. the address of the object in memory. **Note:** the address of x and y may differ in your system.	id (object)	x=15 y='a' print ('address of x is :',id (x)) print ('address of y is :',id (y)) **Output:** address of x is : 1357486752 address of y is : 13480736
chr ( )	Returns the Unicode character for the given ASCII value. This function is inverse of ord() function.	chr (i)	c=65 d=43 print (chr (c)) prin t(chr (d)) **Output:** A +
round ( )	Returns the nearest integer to its input. 1. First argument (number) is used to specify the value to be rounded.	round (number [,ndigits])	x= 17.9 y= 22.2 z= -18.3 print ('x value is rounded to', round (x)) print ('y value is rounded to', round (y)) print ('z value is rounded to', round (z))
type ( )	Returns the type of object for the given single object. **Note:** This function used with single object parameter.	type (object)	x= 15.2 y= 'a' z= True print (type (x)) print (type (y)) print (type (z)) **Output:** <class 'float'> <class 'str'> <class 'bool'>
pow ( )	Returns the computation of a**b i.e. (a**b) a raised to the power of b.	pow (a,b)	a= 5 b= 2 c= 3.0 print (pow (a,b)) print (pow (a,c)) print (pow (a+b,3)) **Output:** 25 125.0 343

37) B)

Explain about string operators in python with suitable example.

String Operators: Python provides the following operators for string operations. These operators are useful to manipulate string.

(i) Concatenation (+): Joining of two or more strings is called as Concatenation. The plus (+) operator is used to concatenate strings in python.

Example:

```
>>> "welcome" + "Python"
'welcomePython'
```

(ii) Append (+ =) : Adding more strings at the end of an existing string is known as append. The operator += is used to append a new string with an existing string.

Example :

```
>>> strl="Welcome to"
>>> strl+="Learn Python"
>>> print (strl)
Welcome to Learn Python
```

(iii) Repeating (*): The multiplication operator (*) is used to display a string in multiple number of times.

Example :

```
>>> strl="Welcome"
>>> print (strl *4)
Welcome Welcome Welcome Welcome
```

38) A) **Explain the different set operations supported by python with suitable example.**

The python supports the set operations such as Union, Intersection, difference and Symmetric difference.

Union:

It includes all elements from two or more sets

(i) In python, the operator | is used to union of two sets. The function union() is also used to join two sets in python.

(ii) Example : Program to Join (Union) two sets using union operator

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
U_set=set_A | set_B
 print(U_set)
```

Output:

{2,4,6, 8, 'A', 'D', 'C', 'B'}

Intersection :

(i) It includes the common elements in two sets

(ii) The operator & is used to intersect two sets in python. The function intersection ( ) is also used to intersect two sets in python.

(iii) Example : Program to insect two sets using intersection operator

```
set_A={'A', 2,4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A & set_B)
```

Output:

{'A','D'}

Difference:

(i) It includes all elements that are in fi rst set (say set A) but not in the second set (say set B)

(ii) The minus (-) operator is used to difference set operation in python. The function difference( ) is also used to difference operation.

(iii) Example : Program to difference of two sets using minus operator

```
set_A={'A', 2,4, 'D']
 set_B={'A', 'B', 'C', 'D'J]
print(set_A - set_B)
```

Output: {2,4}

Symmetric difference:

(i) It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.

(ii) The caret (^) operator is used to symmetric difference set operation in python. The function **symmetric_difference( )** is also used to do the same operation.

(iii) **Example:** Program to symmetric difference of two sets using caret operator set_A={'A', 2,4, 'D'}

```
set_B={'A', 'B', 'C', 'D'}
print (set_A ^ A set_B)
```

Output:

{2,4,'B', 'C'}

38) B)

```
a = float(input('Enter first side: '))
b = float(input('Enter second side: '))
c = float(input('Enter third side: '))
s = (a + b + c) / 2
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.2f' %area)
```