PADASALAI'S - COMMON QUARTERLY EXAMINATION – SEP 2019

STD: 12 **COMPUTER SCIENCE** **Marks: 70**

Time : 2Hrs 30Min

PART - I

Choose the correct answer:

1. **D) Parameters**
2. **C) Concrete Data Type**
3. **A) Name spaces**
4. **B) Half interval**
5. **A) 2**
6. **C) White spaces**
7. **B) Step**
8. **A) 1**
9. **C) clear()**
10. **D) B**
11. **C) Dictionary**
12. **B) __del__()**
13. **D) ER Database**
14. **A)1970**
15. **C) Commit**

PART - II

16. Pure Function:

Pure functions are functions which will give exact result when the same arguments are passed. For example the mathematical function sin (0) always results **0**.

Example:

```
let square x
    return: x * x
```

17. List:

List is constructed by placing expressions within square brackets separated by commas.

Example for List is [10, 20].

18. LEGB Rule:

Scope also defines the order in which variables have to be mapped to the object in order to obtain the value. Let us take a simple example as shown below:

```
1. x:= 'outer x variable'
2. display():
3. x:= 'inner x variable'
```

Output

```
outer x variable
inner x variable
```

4. print x
5. display()

19. Floor Division Operator:

- Floor Division Operator is //

- It only returns the integer part of quotient

Let a=10 >>>a//3 Ans: 3

20. Program to display the sum of n natural numbers:

```
n=int(input("Enter a number:"))
sum=0
for i in range(1,n+1)
    sum+=1
print("Sum of natural numbers upto",n," is : ",sum)
```

21. abs() and chr()

abs ( )	Returns an absolute value of a number. The argument may be an integer or a floating point numb e r.	abs (x)	x=20 y=-23.2 print('x = ', abs(x)) print('y = ', abs(y)) **Output:** x = 20 y = 23.2

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 +

22. Set:

A Set is a mutable and an unordered collection of elements without duplicates. That means the elements within a set cannot be repeated. This feature used to include membership testing and eliminating duplicate elements.

Example:

```
>>> S1={1,2,3,'A',3.14}
>>> print(S1)
        {1, 2, 3, 3.14, 'A'}

>>> S2={1,2,2,'A',3.14}
>>> print(S2)
        {1, 2, 'A', 3.14}
```

23. Types of Database Model:

Hierarchical Model

Relational Model

Network Database Model

Entity Relationship Model

Object Model

24. Primary Key Constraint:

This constraint declares a field as a Primary key which helps to uniquely identify a record. It is similar to unique constraint except that only one field of a table can be set as primary key. The primary key does not allow **NULL** values and therefore a field declared as primary key must have the **NOT NULL** constraint.

PART – III

25. Algorithmic function definition to find minimum of 3 numbers:

```
let min 3 x y z:=
        if x<y then
                if x<z then x else z
        else
                if y<z then y else z
```

26. Difference b/w Constructor and Selector:

Constructors are functions that build the abstract data type.

Selectors are functions that retrieve information from the data type.

27. Asymptotic Notation:

Asymptotic Notations are languages that uses meaningful statements about time and space complexity. The following three asymptotic notations are mostly used to represent time complexity of algorithms:

(i) Big O

Big O is often used to describe the worst-case of an algorithm.

(ii) Big Ω

Big Omega is the reverse Big O, if Bi O is used to describe the upper bound (worst - case) of a asymptotic function, Big Omega is used to describe the lower bound (best-case).

(iii) Big Θ

When an algorithm has a complexity with lower bound = upper bound, say that an algorithm has a complexity O (n log n) and Ω (n log n), it's actually has the complexity Θ (n log n), which means the running time of that algorithm always falls

28. Syntax of While loop:

```
while <condition>:
    statements block 1
[else:
    statements block2]
```

29. a) math.ceil(3.5) ➔ 4
b) abs(-3.2) ➔ 3.2
c)pow(2,0) ➔ 1

30. format( ):

format ( )	Returns the output based on the given format 1. Binary format. Outputs the number in base 2. 2. Octal format. Outputs the number in base 8. 3. Fixed-point notation. Displays the number as a fixed-point number. The default precision is 6.	format (value [, format_ spec])	x= 14 y= 25 print ('x value in binary :',format(x,'b')) print ('y value in octal :',format(y,'o')) print('y value in Fixed-point no ',format(y,'f ')) **Output:** x value in binary : 1110 y value in octal : 31 y value in Fixed-point no : 25.000000

31. Reverse Indexing:

Python enables reverse or negative indexing for the list elements. Thus, python lists index in opposite order. The python sets -1 as the index value for the last element in list and -2 for the preceding element and so on. This is called as **Reverse Indexing.**

Marks = [10, 23, 41, 75]

Marks	**10**	**23**	**41**	**75**
Index (Positive)	0	1	2	3
Index (Negative)	**-4**	**-3**	**-2**	**-1**

32. **SELECT (symbol : σ)**

- General form σ_c (R) with a relation R and a condition C on the attributes of R.
- The SELECT operation is used for selecting a subset with tuples according to a given condition.
- Select filters out all tuples that do not satisfy C.

PROJECT (symbol : Π)

- The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

33. Constructor:

Constructor is the special function that is automatically executed when an object of a class is created. In Python, there is a special function called **"init"** which act as a Constructor. It must begin and end with double underscore. This function will act as an ordinary function; but only difference is, it is executed automatically when the object is created. This constructor function can be defined with or without arguments. This method is used to initialize the class variables.

General format:

def __init__(self, [args ……..]):

<statements>

Example:

```
class Sample:
    def __init__(self, num):
        print("Constructor of class Sample...")
        self.num=num
        print("The value is :", num)
S=Sample(10)
```

Output: Constructor of class Sample...
The value is : 10

PART –IV

34. a) Selection Sort Algorithm:

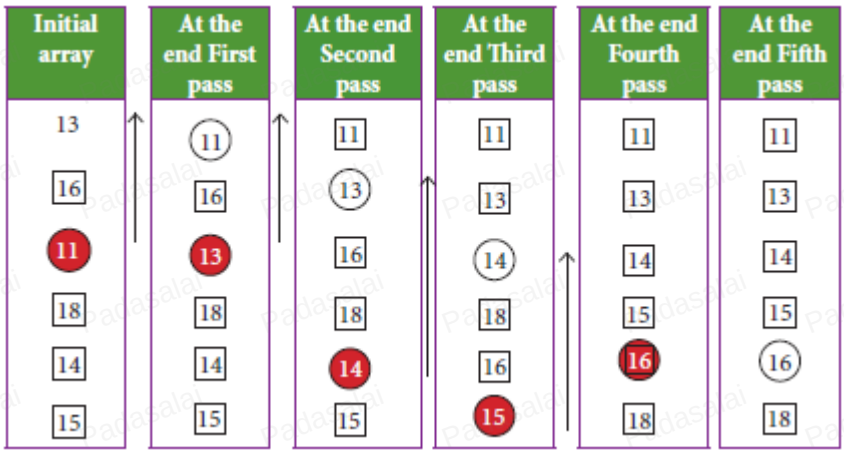
The selection sort is a simple sorting algorithm that improves on the performance of bubble sort by making only one exchange for every pass through the list. This algorithm will first find the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue until the entire array is sorted in respective order.

This algorithm repeatedly selects the next-smallest element and swaps in into the right place for every pass. Hence it is called selection sort.

Pseudo code

1. Start from the first element i.e., index-0, we search the smallest element in the array, and replace it with the element in the first position.
2. Now we move on to the second element position, and look for smallest element present in the sub-array, from starting index to till the last index of sub - array.
3. Now replace the second smallest identified in step-2 at the second position in the or original array, or also called first position in the sub array.
4. This is repeated, until the array is completely sorted.

Let's consider an array with values {13, 16, 11, 18, 14, 15}



In the first pass, the smallest element will be 11, so it will be placed at the first position.

After that, next smallest element will be searched from an array. Now we will get 13 as the smallest, so it will be then placed at the second position.

Then leaving the first element, next smallest element will be searched, from the remaining elements. We will get 13 as the smallest, so it will be then placed at the second position.

Then leaving 11 and 13 because they are at the correct position, we will search for the next smallest element from the rest of the elements and put it at third position and keep doing this until array is sorted.

Finally we will get the sorted array end of the pass as shown above diagram.

34) b) Token in python:

Python breaks each logical line into a sequence of elementary lexical components known as **Tokens**. The normal token types are

1) Identifiers,

2) Keywords,

3) Operators,

4) Delimiters and

5) Literals.

1. Identifiers

An Identifier is a name used to identify a variable, function, class, module or object.

Valid: Sum, total_marks, regno, num1

Invalid : 2Name, name$, total-mark, continue

2. Keywords :

Keywords are special words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

Example: if , for, while, false , true

3 . Operators:

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called **operands**. Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as **operands**.

(i) Arithmetic operators

An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them. They are used for simple arithmetic. Most computer languages contain a set of such operators that can be used within equations to perform different types of sequential calculations.

Operator - Operation	Examples	Result
Assume a=100 and b=10. Evaluate the following expressions		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30 (Integer Division)	3

(ii) Relational or Comparative operators

A Relational operator is also called as **Comparative** operator which checks the relationship between two operands. If the relation is true, it returns **True**; otherwise it returns **False**.

Operator - Operation	Examples	Result
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

(iii) Logical operators

In python, Logical operators are used to perform logical operations on the given relational expressions. There are three logical operators they are **and, or** and **not**.

Operator	Example	Result
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

(iv) Assignment operators

In Python, = is a simple assignment operator to assign values to variable. Let **a** = 5 and **b** = 10 assigns the value 5 to **a** and 10 to **b** these two assignment statement can also be given as **a,b=5,10** that assigns the value 5 and 10 on the right to the variables a and b respectively. There are various compound operators in Python like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	Description	Example
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b=”Computer”
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5
*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2
%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
*=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3

(v) Conditional operator

Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.

Example:

min= 50 if 49<50 else 70 # min = 50

min= 50 if 49>50 else 70 # min = 70

4 . Delimiters

Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings. (,) , { , }, [,], **,** ,= and so on.

5 . Literals

Literal is a raw data given in a variable or constant. In Python, there are various types of literals.

1) Numeric 2) String 3) Boolean

(i) Numeric Literals

Numeric Literals consists of digits and are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.

(ii) String Literals

In Python a string literal is a sequence of characters surrounded by quotes. Python supports single, double and triple quotes for a string. A character literal is a single character surrounded by single or double quotes. The value with triple-quote ''' ''' is used to give multi-line string literal.

(iii) Boolean Literals

A Boolean literal can have any of the two values: True or False.

(iv) Escape Sequences

In Python strings, the backslash "\" is a special character, also called the **"escape"** character. It is used in representing certain whitespace characters: **"\t"** is a tab, **"\n"** is a newline, and **"\r"** is a carriage return. For example to print the message "It's raining", the Python command is

>>> print ("It\'s rainning")

It's raining

35) a) Program to Check the given character is a vowel or not

```
ch=input("Enter a character: ")
if ch in ( 'a',' A',' e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')
        print(ch, "is a vowel")
else:
        print(ch, "is not a vowel")
```

Output:

```
Enter a character: L
L is not a vowel
```

35. B) Function Arguments:

Arguments are used to call a function and there are primarily 4 types of functions that one can use: *Required arguments, Keyword arguments, Default arguments and Variable-length arguments.*

1. Required Arguments

"Required Arguments" are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function defi nition. You need atleast one parameter to prevent syntax errors to get the required output.

```
def printstring(str):
        print ("Example - Required arguments ")
        print (str)
        return
printstring("Welcome")
```

Output:

```
Example - Required arguments
Welcome
```

2. Keyword Arguments

Keyword arguments will invoke the function after the parameters are recognized by their parameter names. The value of the keyword argument is matched with the parameter name and so, one can also put arguments in improper order (not in order).

Example:

```
def printdata (name):
    print ("Example-1 Keyword arguments")
    print ("Name :",name)
    return
printdata(name = "Gshan")
```

Output:

```
Example-1 Keyword arguments
Name :Gshan
```

3. Default Arguments

In Python the default argument is an argument that takes a default value if no value is provided in the function call. The following example uses default arguments, that prints default salary when no argument is passed.

Example:

```
def printinfo( name, salary = 3500):
        print ("Name: ", name)
        print ("Salary: ", salary)
        return
printinfo("Mani")
```

Output:

```
Name: Mani
Salary: 3500
```

4. Variable-Length Arguments

In some instances you might need to pass more arguments than have already been specified. Going back to the function to redefine it can be a tedious process. Variable-Length arguments can be used instead. These are not specified in the function's definition and an asterisk (*) is used to define such arguments.

Lets see what happens when we pass more than 3 arguments in the sum() function.

```
def sum(x,y,z):
    print("sum of three nos :",x+y+z)
sum(5,10,15)
```

Output:

Sum of three nos: 30

36. A) 1. id() 2.type() 3.lower() 4. max() 5. min()

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' s= True print (type (x)) print (type (y)) print (type (s)) **Output:** <class 'float'> <class 'str'> <class 'bool'>
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
min ( )	Returns the minimum value in a list.	min (list)	MyList = [21,76,98,23] print ('Minimum of MyList :', min(MyList)) **Output:** Minimum of MyList : 21

max ( )	Returns the maximum value in a list.	max (list)	MyList = [21,76,98,23] print ('Maximum of MyList :', max(MyList)) **Output:** Maximum of MyList : 98
sum ( )	Returns the sum of values in a list.	sum (list)	MyList = [21,76,98,23] print ('Sum of MyList :', sum(MyList)) **Output:** Sum of MyList : 218

b) Program to check whether the given string is palindrome nor not.

```
str1 = input ("Enter a string: ")
str2 = ' '
index=-1
for i in str1:
  str2 += str1[index]
  index -= 1
print ("The given string = { } \n The Reversed string = { }".format(str1, str2))
if (str1==str2):
  print ("Hence, the given string is Palindrome")
else:
  print ("Hence, the given is not a palindrome")
```

Output : 1

Enter a string: malayalam
The given string = malayalam
The Reversed string = malayalam
Hence, the given string is Palindrome

Output : 2

Enter a string: welcome
The given string = welcome
The Reversed string = emoclew
Hence, the given string is not a palindrome

37. A) SET OPERATIONS:

As you learnt in mathematics, the python is also supports the set operations such as Union, Intersection, difference and Symmetric difference.

I) Union: It includes all elements from two or more sets

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

```
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'}

II) Intersection: It includes the common elements in two sets

The operator & is used to intersect two sets in python.

The function **intersection( )** is also used to intersect two sets in python.

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

Output: {'A', 'D'}

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

The minus (-) operator is used to diff erence set operation in python. Th e function **difference( )** is also used to diff erence operation.

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

Output: {2, 4}

IV) Symmetric difference

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.

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

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

Output:

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

37 . b) Class and Object:

Defining classes

In Python, a class is defined by using the keyword class. Every class has a unique name followed by a colon (:).

Syntax: class class_name:
statement_1
statement_2
……..
……..
statement_n

Example:

class Sample:

x, y = 10, 20

Creating Objects

Once a class is created, next you should create an object or instance of that class. Theprocess of creating object is called as "Class Instantiation".

Syntax:

Object_name = class_name( )

Note that the class instantiation uses function notation ie. class_name with

Example:

```
class Student:
    mark1, mark2, mark3 = 45, 91, 71 #class variable
def process(self): #class method
    sum = Student.mark1 + Student.mark2 + Student.mark3
    avg = sum/3
    print("Total Marks = ", sum)
    print("Average Marks = ", avg)
    return

S=Student()
    S.process()
```

Output

Total Marks = 207

Average Marks = 69.0

38) Output

```
C
C O
C O M
C O M P
C O M P U
C O M P U T
C O M P U T E
C O M P U T E R
```

38. b)

a) CREATE

You can create a table by using the **CREATE TABLE** command. When a table is created, its columns are named, data types and sizes are to be specified. Each table must have at least one column. The syntax of **CREATE TABLE** command is :

Syntax:

CREATE TABLE *<table-name>*
(<column name><data type>[<size>]
(<column name><data type>[<size>]……);

Example:

CREATE TABLE *Student*

(Admno integer,
Name char(20),
Gender char(1),
Age integer,
Place char(10),);

b) SELECT

One of the most important tasks when working with SQL is to generate Queries and retrieve data. A Query is a command given to get a desired result from the database table. The **SELECT** command is used to query or retrieve data from a table in the database. It is used to retrieve a subset of records from one or more tables. The **SELECT** command can be used in various forms:

Syntax:

SELECT <column-list>FROM<table-name>;

Example:

SELECT * FROM STUDENT;

c) DELETE

The **DELETE** command permanently removes one or more records from the table. It removes the entire row, not individual fields of the row, so no field argument is needed.

Syntax:

DELETE FROM table-name WHERE condition;

Example:

DELETE * FROM Student;

d) ALTER

The **ALTER** command is used to alter the table structure like adding a column, renaming the existing column, change the data type of any column or size of the column or delete the column from the table.

Syntax:

ALTER TABLE <table-name> ADD <column-name><data type><size>;

Example:

ALTER TABLE Student ADD Address char;

e) DROP

The **DROP TABLE** command is used to remove a table from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using **DROP TABLE** command. But there is a condition for dropping a table; it must be an empty table.

Syntax:

DROP TABLE table-name;

Example:

DROP TABLE Student;