PYTHON INTERVIEW QUESTIONS

Q1. Python is what type of language (procedure or object oriented or scripting)?

Ans: Procedure oriented languages deals with functions.

```
void main()
{   Logics Here
}
```

Object oriented langauges deals with classes.

```
class Test
{     logics  Here
}
```

Scripting languages we can write the logics directly.

```
num1 = 10
num2 = 20
res = num1 + num2
print(res)
```

Python is multi-Programming, supports above three flavours.
Python is procedure-oriented deals with functions.
Python is object-oriented deals with classes.
Python is script-based language write the code directly.

Q2. what is keyword & write a code to get the keywords?

Ans:

python keywords are special reserved words which is having fixed meaning.
It is not possible to use keywords as a identifier.
ex: if,else,try,elif....etc

To print the keyword use "kwlist" variable of "keyword" module.

Code to print keywords

```
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

Q3. Explain any 4-errors in python?

Ans:

TypeError	:	*when we combine different of data.*
IndentationError	:	*if the alignment is not current.*
ValueError	:	*if the type conversion is not possible.*
NameError	:	*if the data is not available*

Q4. What are the comments in python? What are the advantages of comments?

Ans:

Comments are used to write the description about the application to understand the logics easily.

The main objective of comments are the project maintenance will become easy.

The comments are non-executable code.

There are two types of comments in python

Single line comments :

```
# this is filter logics
```

Multiline comments: Below two syntax are valid & same.

```
"""hey ratan sir"""
'''hi durga sir'''
```

Q5. What is the meaning of Dynamically typed? How to check the type of the data?

***Ans**: Other languages every variable must declare the data type.*

```
int num1 = 100;
int num2 = 200;
```

python is dynamically typed language, So no need to declare the data type. At runtime it will take the type automatically.

```
num1 = 100
num2 = 200
```

To check the type of the data use type() function.

```
print(type(10))
print(type(10.5))
```

Q6. what are the different ways to format the data?

***Ans**:*

we can format the data properly in three ways

% { } f

```
eid,ename,esal = 111,'ratan',10000.45
print("emp id=%d emp name=%s Emp sal=%g"%(eid,ename,esal))

print("emp id={} emp name={} Emp sal={}".format(eid,ename,esal))
print("emp id={0} emp name={1} Emp sal={2}".format(eid,ename,esal))

print(f"emp id={eid} emp name={ename} Emp sal={esal}")
```

Q7. In below code how many objects are created?

```
a = True
b = True
c = True
d = False
```

Ans: 2

Q8. what is the difference between the is,is not & ==,!=?

Ans:

Is, is not : To perform memory comparison.
== != : is comparison of data

```
d1 = [10,20,30]
d2 = [10,20,30]
print(d1==d2)
print(d1!=d2)
Print(d1 is d2)
Print(d1 is not d2)
```

Q9. What is the difference bitwise(&) , logical(and) ?

Ans: cond1 & cond2 : it will check the both conditions then only it will give result.
cond1 and cond2 : Here the second condition execution depends on first condition
if the first condition is true then only second condition executed.
if the first condition is false then second condition not executed.

```
if (10>20) & ("anu"+10=="anu"):
        print("Good morning")
else:
        print("Good Night")

if  10>20  and  "anu"+10=="anu":
        print("Good morning")
else:
        print("Good Night")
```

Q10. what is the difference between the concatenation & replication?

Canact : used to combine two list store the result in third variable.
Replication : same data replicated multiple times.

```
data1 = [10,20,30]
data2= [40,50,60]
res1 = data1 + data2
print(res1)

res2 = data1*2 + data2*3
print(res2)
```

Q11. In Application when to use for loop & while loop Give the scenario?

Ans: If we know the number of iteration use for loop.

```
for x in range(10):
        print("hi ratna sir")
```

If we don't no how many times to repeat the loop then use while.

```
while True:
```

```
                    logics
                    if num==25:
                            break
```

Q12. What is the Difference between mutable & immutable data?

Ans: mutable data : The data allows modifications.

ex: list, set,dict....

Immutable data : It does not allows modifications.

ex: int, float, str, tuple....etc

Q13.How to add one list into another list?

Ans: To add one list into another list use extend() function.

```
data1 = [10,20,30]
data2 = [40,50,60]
data1.extend(data2)
print(data1)
```

Q14. What is the diff between getting the data using index base & slicing?

```
Ans:    data = [10,20,30,40,50]
            print(data[0])
            print(data[2])
            #print(data[9])               IndexError: list index out of range

            print(data[2:4])
            print(data[:4])
            print(data[2:])
            print(data[:])

            print(data[1:4:2])
            print(data[1::2])
            print(data[:4:2])
            print(data[::2])
```

The index will return the data in single element format.

```
print(data[2])  #30
```

The slicing will return ther data in list format.

```
print(data[2:3])         # [30]
```

Q15. What is data unpacking?

Ans: Extracting the elements from the list assigning to variables.

```
data = [10,20,30,40]
a,b,c,d = data
print(a,b,c,d)

data = [10,20,30,40]
```

```
a,b,*c = data
print(a,b,c)
```
**c : will get the data in list format.*

Q16. What is the difference between sort() vs. Sorted()?

Ans: *sort() will sort the data in memory.*

sorted() will print the report in sorting order but data in memory not sorted.

```
names = ["ratan","anu","sravya","naresh"]
names.sort()
print(names)

names = ["ratan","anu","sravya","naresh"]
names.sort(reverse=True)
print(names)

fruits = ['apple',"mango","banana","orange"]
print(sorted(fruits))
print(fruits)

fruits = ['apple',"mango","banana","orange"]
print(sorted(fruits,reverse=True))
print(fruits)
```

Q17. What is the output of below code?

Ans:
```
t1 = (1,2,4,3)
t2 = (1,2,3,4)
print(t1<t2)
print(t1>t2)

t1 = ("anu","ratan","anu")
t2 = ("anu","ratan","ani")
print(t1>t2)
print(t1<t2)

t1 = (10,10,"ratan")
t2 = (10,"ratan","anu")
print(t1>t2)
print(t1<t2)
```

Q18. What is the Difference between List & Tuple?

Ans:

List data :

store the multiple elements : [10,20,30,40] : ordered collection : index base: mutable data

Tuple data :

store the multiple elements : (10,20,30) : ordered collection : index base : immutable data

Q19. What are the ways to remove the data from list?

Ans:

a. remove()
b. pop()
c. del
d. clear

```
fruits = ['mango','banana','apple']
fruits.remove("banana")
#fruits.remove("anu")  ValueError: list.remove(x): x not in list
print(fruits)

names = ["ratan","anu","sravya","radha"]
names.pop(2)
names.pop()
#fruits.pop(10)  IndexError: pop index out of range
print(names)

#remove the data using sliceing
numbers = [1,2,3,4]
del numbers[2:]
del numbers[:1]
print(numbers)

#clear remove the complete data
data = [10,"ratan",True,10.5]
data.clear()
print(data)
```

Q20: How to get the predefined functions are present in python like list,tuple,set,dict...etc?

Ans: *To get the predefined support use dir() function.*

```
print(dir(list))
print(dir(tuple))
print(dir(set))
print(dir(dict))
```

To get the module information user dir()

```
import keyword
print(dir(keyword))
```

Q21. Define the module in python? What is the Difference between normal import & from import?

Ans: python file is called module

When we are accessing other module data we have to import that module
There are two ways to import the data,

Normal import :
We have to access the data using module name.
```
import operations
```
From import :
we can access the module data directly without module name.
```
from operations import add,mul
```

Q22. what is the purpose of the nonlocal keyword in python?

Ans: *inside the inner function to represent (or change) outer function variable use nonlocal keyword.*

```
def outer():
    name = "ratan"
    def inner():
        nonlocal name
        name = "sravya"
        print(name)
    #inner function calling
    inner()
    print(name)
#calling outer function
outer()
```

Q23. What is the difference between local import & global import?

Ans: There are two types of import

1. global import : Entire module can access the imported data.
2. local import : Only specific function can access the import data.

```
#operations.py
add() function
mul() function
#client.py  : 1. global import:Entire module can use this import
from operations import add,mul
class Myclass:
    def my_func1(self):
        add(10,20)
        mul(3,4)
    def my_func2(self):
        add(10,20)
        mul(3,4)
if __name__ == "__main__":
        c = Myclass()
        c.my_func1()
        c.my_func2()
```

Q24. What is the class & Object?

Ans:

class vs. object :

a. class is a logical entity it contains logics.
where as object is physical entity it is representing memory.

b. class is blue print it decides object creation
without class we are unable to create object.

c. Based on single class it is possible to create multiple objects but every object occupies memory.

Q25. What the methods present in python class?

Ans: Python is object oriented programming language.
The object oriented programming languages deals with classes & objects.
Declare the class using class keyword.

There are three types of methods we can declare inside the class.
1. instance methods : first argument is self : access using obj-name
2. static methods : use @staticmethod qualifier: access using class-name
3. cls methods : use @classmethod qualifier & cls argumnet: access using object name & class-name

```
class MyClass:
        def wish(self):
                print("Good morning....")
        @staticmethod
        def disp():
                print("hi Students")
        @classmethod
        def info(cls):
                print("Welcome to Durgasoft...")
if __name__ == "__main__":
        c = MyClass()
        c.wish()
        MyClass.disp()
        c.info()
        MyClass.info()
```

Q26. How to convert python data to JSON format?

Ans: Declare the json data using : { } The json contains the data in key:value pair format.

In json the keys must be strings, values can be any type.

To convert python data to json format use dumps() function.

```
import json
py_data =
{"name":"ratan","hobbies":("cricket","eating"),"id":111,"email":None,"status":True}
json_data = json.dumps(py_data)
print(json_data)
```

Q27. What is the meaning of polymorphic Function in python?

Ans: The function can take different types of input data it will give proper result.

predefined polymorphic functions : length() type() sum() min() max()

```
print(len("ratan"))
print(len([10,20,30,40]))
print(len((10,20,30)))

print(type(10))
print(type(10.5))
print(type("ratan"))
print(type([10,20,30]))
```

Q28. What is the Advantage of Constructor?

Ans: Constructor used to initialize the data during object creation.

```
class Emp:
        def __init__(self,eid,ename,esal):
                self.eid = eid
                self.ename = ename
                self.esal = esal
        def __str__(self):
                return f"Emp id:{self.eid} Emp name:{self.ename} Emp
sal:{self.esal}"

if __name__ == "__main__":
        e1 = Emp(111,"ratan",10000.45)
        print(e1)
```

Q29. Define inheritance? how many types inheritance supported by python?

Ans: The process of creating new class by taking the properties from existing class is called inheritance. There are 5-types of inheritance in python,

i.	*Single inheritance*	*: one parent with one child.*
ii.	*Multi level inheritance*	*: generation by generation.*
iii.	*Hierarchical inheritance*	*: one parent with multiple child classes.*
iv.	*Multiple inheritance*	*: multiple parent classes with one child.*
v.	*Hydrid inheritance*	*: hierarchical + multiple*

Q30. How to convert JSON object to python format?

***A**ns: To convert the json object to python format use loads() function.*
When we convert JSON object to python format, in python we will get Dictionary object.

```
        import json
json_data = '{"name": "ratan", "hobbies": ["cricket", "eating"], "id": 111, "email": null, "status": true}'
        py_data = json.loads(json_data)
        print(py_data)
```

Q31. Explain about __init__, __str__, __del__ ?

Ans:
__init__ : constructor : executed when we create the object.
__str__ :
it will be executed when we print the object, it will return only string data.
if we return other than stirng data we will get TypeError
__del__ : destructor
it will be executed when we destroy the object.Destroy the object using del keyword.

```
class MyClass:
        def __init__(self):
                print("This is constructor")
        def __str__(self):
                return "ratan"
        def __del__(self):
                print("object destroyed...")
if __name__ == "__main__":
        c = MyClass()
        print(c)
        del c
```

Q32. When to use instance & static variables give the scenario?

Ans:
a. instance variables every object separate memory created.
ex: eid,ename, esal
static varaibles for all object single memory created.
ex: country,company_name,School_name.
b. static variables data common for all objects,if one object change the data then all objects are reflected.

instance variable is specific to object, here one object change the data only that object is reflected.

Q33. What are the Different ways to call the parent class Constructors?

Ans: There are two ways to call the parent class constructor.

```
a. using super()
        super().__init__()
b. using class-name
        A.__init__(self)
class A:
        def __init__(self,num):
                print("A class constructor..",num)
class B(A):
        def __init__(self):
                super().__init__(10)
                print("B class constructor")
if __name__ == "__main__":
        b = B()
```

Q34. Define the variable? what are the different types of variables in python?

Ans: Variables used to store the data & it is fixed memory location storing the value.
Variable is a named memory location.

```
ex :    eid = 111
        ename = "ratan"
        esal= 10000.45
```

Python is procedure oriented language we have 2-types of variables:

1. *local variables*
2. *global variables*

Python is object oriented programming language There are 3-types of variables:

1. *Local variables*
2. *instance variables*
3. *static variables*

Q35. What is the difference between variable Argument(*) & keyword Argument(**)?

Ans: * : *Variable argument : used to take the multiple values(0,1,2,3...etc)*
it will take the data by default tuple format.
** : *keyword argument : can take the data in key=value format. keys are by default string*

```
def disp(*arg,**kwarg):
print(arg)
print(kwarg)
```

```
#function call
disp(10,20,30,i=11,j=22,k=33)
```

Q36. Give some information about functions in python?

Ans: Functions used to write the block of python code.
We can call the function in application wherever we want in different modules.
Declare the function using def keyword.
main objective of the function is code reuse.
The function can return the value, if we are not return anything by default return value is None.
The function can return multiple values.
Python supports inner functions.

```
def wish():
  print("Good morning...")
def disp():
  return "ratan"
#function call
wish()
disp()
```

Q37. What is the purpose of global keyword?

Ans: used to represent global variables.
case 1: It is possible to declare the global variable inside the function by using global keyword.

```
def wish():
        global name
        name  = 'ratan'
        print(name)
#function calling
wish()
print(name)
```

case 2: Inside the function to represent or change the global variables: use global keyword

```
name = "sravya"
def wish():
        global name
        name  = 'ratan'        # it is global data changing
        print(name)
#function calling
wish()
print(name)
```

Q38. What are the different arguments of the functions?

Ans: The function can take the arguments,

There are four types of arguments in python,

a. default arguments

```
def emp_details(eid=111,ename="ratan",esal=10000.45):
        print(eid,ename,esal)
emp_details()
emp_details(444,"sravya",30000.45)
```

b. required arguments

```
def emp_details(eid,ename,esal):
        print(eid,ename,esal)
emp_details(444,"sravya",30000.45)
```

c. keyword arguments

```
def emp_details(eid,ename,esal):
        print(eid,ename,esal)
#function call
emp_details(eid=111,ename="ratan",esal=10000.45)
```

d. variable arguments

```
def disp(*arg):
        print(arg)
disp(10,20,30)
```

Q39. What is the difference between List & Set?

Ans: *List Data*

```
  [10,20,30]   : ordered collection : list duplicates are allowed
               : supports indexing  : support slicing
```

Set Data

```
     {1020,30}   : un-ordered collection  : set duplicates are not allowed
                        : no indexing  : no slicing
```

If the application requirement to store the data with duplicate values use list.

If the application requirement to store the data without duplicate use set.

Q40. How to add the Set inside the set & dict inside the dict?

Ans: To Add the Set inside the Set use : update() function.

```
s = {10,20,30}
s.update({1,2,3})
print(s)
```

To add the Dict inside the Dict use update() function.

```
d1={1:"ratan",2:"anu"}
d2={3: "ratan",4:"anu"}
d1.update(d2)
```

```
print(d1)
```

Q41. How take the list,tuple,set,dict inputs from end user?

Ans: To take the multiple inputs from end-user use eval() Function.
eval() function used to evalute the expressions also.

```
data = eval(input("Enter data:"))
print(data)
n1,n2,n3 = eval(input("Enter data:"))
print(n1,n2,n3)

n1,n2,*n3 = eval(input("Enter data:"))
print(n1,n2,n3)

res = eval(input("Enter a expresson:"))
print(res)
```

Q42. Define dict? When we will use dict give the scenario?

Ans: a. dict is a data type to store the group of key=value pairs
the key=value pairs also known as item. The dict contains group of items.
b. declare the data using : {}
{key:value,key:value,key:value}
c. keys must be unique but values can be duplicates.

case 1:
ratan ---- 5 anu ------ 2 is ----- 5
{"ratan":5,"anu":2,"is":5}

case 2:
10.5.6.4 ----- 5
10.5.4.2 ----- 10
78.90.45.4 ----- 5
{"10.5.6.4":5,"10.5.4.2":10,"78.90.45.4":5}

case 3:
e1 ---> 111 ratan 10000.45 e2 ---> 222 anu 20000.45
The references are keys,The object becomes values

Q43. How to merge two lists or tuples to make the dict?

Ans: Just zip the two elements convert into dict format.

```
l1 = [10,20,30]
l2 = ["ratan","anu","sravya"]
res = dict(zip(l1,l2))
print(res)

t1 = (1,2,3)
t2 = ("aaa","bbb","ccc")
res = dict(zip(t1,t2))
print(res)
```

Q44. what is the purpose of lambda expression?

Ans:

A lambda function is a anonymous function(A function without name).

The lambda function can take any number of arguments but only one expression, which is evaluated and returned.

lambda keyword is used to define the lambda expression.

```
syntax:
            lambda arguments : expression

res = lambda num : num*num
print(res(10))

res = lambda num1,num2 : num1*num2
print(res(10,20))

print((lambda n1,n2,n3 : n1+n2+n3)(10,20,30))
```

Q45. What is the purpose of the Filter?

Ans: Filter: filter used to filter the some data from complete data.

```
            filter(filter_logics,input_data)

data = [4,5,6,22,3,9,8]
res = list(filter(lambda num : num%2==0,data))
print(res)


data = ["ratan","raju","rani","durgasoft","anu"]
res = list(filter(lambda name : name.startswith('r') and len(name)<=4 ,data))
print(res)
```

Q46. what is the purpose of Map() function?

Ans: Maper :The map function is used to apply a particular operation to every element in a sequence.

```
            map(logics,input)

marks = [34,56,34,23,67,33]
print(list(map(lambda x : x+2 ,marks)))

names = ("ratan","durga","sravya")
print(list(map(lambda x:x+"good" ,names)))
```

Q47. What is the purpose of reduce() function?

Ans: *reduce will perform computition on all elements it returns only one element as result.*

The reduce will take always 2 inputs to perfrom the operations.

reduce is a function present in functools module so import the module using import statement.

```
    [1,2,3,4]
     3  3 4
       6  4
          10
```

```
from functools import reduce
data = [1,2,3,4]
res = reduce(lambda x,y : x*y,data)
print(res)

names = ["ratan","durga","raju"]
res = reduce(lambda x,y:x+y,names)
print(res)
```

Q48. Define the Exception? What is the main objective of exception handing?

Ans: *The dictionary meaning of the exception is abnormal termination.*

exception is is a object raised during the execution of the application to disturb the normal flow of execution is called exception.

Whenever the exception raised in application,

a. Program terminated abnormally.
b. Rest of the application not executed.

we can handle the exception using try-except blocks.
once we handle the exeption

a. Program terminated normally.
b. Rest of the application executed.

The main objective of exception handling is to get the normal termination of application, to execute the remaining code.

Q49.What is the purpose of try-except blocks?

Ans:

try-except blocks used to handle the exception.

try	*: logics it may or may not raise an exception.*
except	*: these logics are executed when exception raised in try.*

```
try:
    n = int(input("Enter a num:"))
    print(10/n)
except ZeroDivisionError as e:
    print("Exception info....",e)
print("rest of the app....")
```

Q50. What is the purpose of default except in python?

Ans: *The except without any exception is called default except*
The default except can handle all types of exception.
Along with the default except, it is possible to write the normal except blocks. But the default except block must be last statement.

```
try:
    n = int(input("enter a num:"))
    print(10/n)
    print(10+"ratan")
except:
    print("durgasoft....")
print("Rest of the app")
```

Q51. What are the possible ways to handle the multiple exception in python?

Ans:

Below all cases can handle multiple exceptions

```
except BaseException as a:
except Exception as a:
except (ZeroDivisionError,ValueError) as a:
except (IndexError,KeyError,TypeError) as a:
except:                  (this is default except can handle all the exceptions)
```

Q52. What is the purpose of finally block?

Ans:

finally block is always executed irrespective of try-except blocks.
finally block executed both normal & abnormal termination of application.
finally blocks is used to release the resource like,
database closeing,file closeing in both normal cases & abnormal cases.

```
try:
    connection open
    tx1
    tx2
    tx3
except ZeroDivisionError as e:
    sdflskdfjl
```

close the connection

1. all operations are success : normal termination : Connection closed
2. in tx2 exception raised: except is matched : normal terminaton : Connection closed
3. in tx3 exception raised : except is not matched : abnormal termination : connection not closed

Q53. If the try-except-finally blocks contains exception so Virtual machine will display which exception?
Ans:

```
try:
  print(10/0)
except ArithmeticError as e:
  print(10+"ratan")
finally:
  print([1,2,3][8])
```

Note : if the try-except- finally blocks contains exceptions it will shows all exceptions.

```
ZeroDivisionError: division by zero
During handling of the above exception, another exception occurred:
TypeError: unsupported operand type(s) for +: 'int' and 'str'
During handling of the above exception, another exception occurred:
IndexError: list index out of range
```

Q54. What is the purpose of raise keyword?
Ans: *raise keyword is used to raise the exceptions.*
using raise keyword we can raise predefined & user defined exceptions.

1. raising predefined error valid but not recommanded because predeinfed error having some fixed meaning.

```
age = int(input("Enter u r age:"))
if age>22:
        print("eligible for mrg")
else:
        raise ValueError("u r not eligible")
```

2. raising userdefined exceptions : without description

```
#step 1: create the userdefined exception
class InvalidAgeError(Exception):
        pass
#step2: use the excetion in project
```

```
age = int(input("Enter u r age:"))
if age>22:
        print("eligible for mrg")
else:
        raise InvalidAgeError("your are not eligible.....")
```

Q55. What are the different modes of the file?

Ans: *To work with the files we have three steps*

step 1: open the file	:	*open() function*
step 2: read or write operations	:	*read() write() function*
step 3: close the file	:	*close() function*

Modes of the file

r	:	*read*
w	:	*write*
a	:	*append*
x	:	*exclusive creation*
r+	:	*read & write*
w+	:	*write & read*
a+	:	*append & read*

```
f = open("ratan.txt","w")
f = open("ratan.txt","r")
```

To read the data file is mandatory. But to write & append the data file is optional.

Note : By default the file is open in read mode.

Q56. What are the different ways to read the data from the file?

Ans: *There are three ways to read the data from file.*

1. using read()	*: char by char*
2. using readline()	*: reads single line*
3. using readlines()	*: reads all lines in list format.*

```
f = open("ratan.txt",'r')
res = f.read(5)
print(res)

f = open("anu.txt",'r')
res = f.readline()
print(res)

f = open("durga.txt",'r')
res = f.readlines()
print(res)
```

```
f.close()
```

Note: The readlines() read the data in list format, so we can apply indexing & slicing to the get the specefic lines.

Q57. What is the purpose of seek() & tell() function in python?
Ans:

After writing the data the control is pointing at last but to read the data move the cursor first location using seek.

After reading 5-chars the data the control is pointing at 5th char but to move the cursor first location using seek.

tell() function will give the cursor position.

```
f = open("ratan.txt",'r')
res = f.read(3)
print(res)
print(f.tell())
f.seek(0)
res = f.read(4)
print(res)
print(f.tell())
f.close()
```

Q58. What are the different ways to close the file?
Ans: *There are three different ways to close the file.*

1. we are closing file explicitly : use close() function.

```
f = open("sample.txt","w")
f.write("hi sir how are you")
f.close()
```

2.when we open the file using with statement once the with statement is completed file is automatically closed.

```
with open("sample.txt") as f:
    data = f.read()
    print(data)
print(f.closed)          #True
```

3.when we reassing existing file reference to new file then existing file will be automatically closed.

```
f1 = open("sample.txt")
f1 = open("ratan.txt")
```

here sample.txt file is closed automatically.

Q59. How to work with the directories/folders in python?

***Ans**:To work with the directory use os module.*

To create the directory use mkdir().

To change the control inside the directory use chdir()

To get the current working directory use getcwd()

To remove the directory use rmdir().

if the directory already available, But we are trying to create the new directory with same name we will get FileExistsError

To delete the directory that directory should be empty.

if the directory contains files we can not delete the directory. So first delete the files then delete the directory.

```
import os
os.mkdir("ratan")
os.chdir("ratan")
print(os.getcwd())
os.rmdir("ratan")
print("operatios are completed")
```

Q60. When we will instance & static variables in Application?

***Ans:** a. instance variables every object separate memory created.*

ex: ename, eid,esal

static variables for all object single memory created.

ex: country,company_name,School_name

b. instance variable is specific to object, here one object change the data only that object is reflected. Instance variable modifications reflect on specific object.

static variables data common for all objects, if one object change the data then all objects are reflected. Static variables modifications reflect on multiple objects.

Q61. What are local variables and global variables in Python?

Ans: Global Variables:

Variables declared outside a function or in a global space are called global variables. These variables can be accessed by any function in the program.

Local Variables:

Any variable declared inside a function is known as a local variable. This variable is present in the local space and not in the global space.

Q62. When to use a tuple vs. list vs. set vs. dictionary in Python?

Ans:

Use a list to store a sequence of items with duplicates that may change(mutable).

Use a tuple to store a sequence of items with duplicates that will not change(immutable).

Use a set to store a unique elements that will change(mutable).

Use a dictionary when you want to associate pairs of two items "key:value".

Q63. What is PEP 8 and why is it important?

Ans:PEP stands for Python Enhancement Proposal.

A PEP is an official design document providing information to the Python community, or describing a new feature for Python or its processes.

PEP 8 is especially important since it documents the style guidelines for Python Code. Python open-source community requires you to follow these style guidelines sincerely and strictly.

```
Assignment operator:  pep8 standards
        num=10        # pep8 : missing whitespaces around operator
        num = 20  # valid and recomanded
Comma (,): pep8 standards
        num1,num2,num3 = 10,20,30 # pep8:missing whitespaces after ,
        num1, num2, num3 = 10, 20, 30
import PEP-8 standards:
        # Valid but Not Recommended
        import keyword, time
        # valid and Recommended
        import keyword
        import time
```

Q64. What is pass in Python?

The pass keyword represents a null operation in Python. It is generally used for the purpose of filling up empty blocks of code which may execute during runtime but has yet to be written. Without the pass statement in the following code, we may run into some errors during code execution.

```
1. if(True):
         pass

2. def myEmptyFunc():
              pass
        myEmptyFunc()
```

```
3. class MyClass(ABC):
        @abstractmethod
        def add(self):
                pass
```

Q65. What is the use of self in Python?

Ans: Self is used to represent the instance of the class.

With this keyword, you can access the attributes and methods of the class in python.

```
self.wish()
self.num1 + self.num2
```

self is not a keyword in Python.

multiple functions can take the self like,

a. instance methods takes the self
b. constructor argument take the self.
c. __str__ takes the self....etc

Q66. What are the PEP8 standard we need to fallow while writing the comments?

Ans: Comments PEP-8:

a. Limit the line length of comments and docstrings to 72 characters.
b. Use complete sentences, starting with a capital letter.
c. Make sure to update comments if you change your code.
d. after # give one white space

Q67. What is the difference between break & continue?

Ans: The break statement terminates the loop immediately. It will execute the remaining code of the application.

The continue used to skip the particular iteration, remaining iteration will execute normally.

```
for x in range(1,11):
        if x==5:
                break
        print(x)

print("**********")

for x in range(1,11):
        if x==5:
                continue
        print(x)
```

Q68. Define the Aggregation & Composition?
Ans:

The process using one class reference in another class is called Aggregation.
It represents "has-a" relationship.
It is a unidirectional association i.e. a one-way relationship.
For example, a department can have students but vice versa is not possible so it is unidirectional in nature.
In Aggregation, both the entries can survive individually which means ending one entity will not affect the other entity. Aggregation is a week relation.
ex: The person class uses Address class
The college class used dept class.

Strong aggregation is called composition.
ex: The emp class contains salary, once we delete emp object salary also deleted.

Q69. Define polymorphism? Explain Method Overriding in python?

Ans: polymorphism : "many forms"
one functionality with multiple behaviors.
The ability to appears more than one form is called polymorphism.

```
class Parent:
    def eat(self):                          #Overridden method
        print("idly...")
class Child(Parent):
    def eat(self):                          #Overridden method
        print("poori...")
if __name__ == "__main__":
    c = Child()
    c.eat()
```

Note: Method implementation already present in parent class, Rewriting the parent method implementation in child class is called overriding.

Q70. What is the difference between normal methods & abstract method in python?

Ans:

There are two types of methods

1. *Normal methods*
2. *Abstract methods*

Normal methods:

Normal methods contains method declaration & implementation.

```
def disp(self):
        print("Good Morning")
```

abstract methods:

The abstract method contains only method declaration but not implementation.

To represent your method is a abstarct use @abstractmethod.

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

Q71. What is the purpose of in & not in?

Ans: *in , not in to check the data available or not & return boolean value.*

```
msg = "hi ratan sir"
print("ratan" in msg)
print("anu" in msg)

print("ratan" not in msg)
print("anu" not in msg)

print(10 in [10,20,30])
print(100 not in (10,20,30))
print(5 in {10,20,30})
print(111 in {111:"ratan",222:"durga"})
```

Q72. What are unit tests in Python?
Ans:
Unit Testing is the first level of software testing where the smallest testable parts of a software are tested.
Unit test is performed using module "unittest".

Unit testing means testing different components of software separately.The unit test is important. Imagine a scenario, you are building software that uses three components namely A, B, and C. Now, suppose your software breaks at a point time. How will you find which component was responsible for breaking the software? Maybe it was component A that failed, which in turn failed component B, and this actually failed the software. There can be many such combinations.

This is why it is necessary to test each and every component properly so that we know which component might be highly responsible for the failure of the software.

```
import unittest
class SimpleTest(unittest.TestCase):
    def test(self):
        self.assertTrue(True)
if __name__ == '__main__':
    unittest.main()
```

Q73. Define abstraction? Difference between normal class & abstract class in python?
Ans:
Abstraction is a process highlighting service details but not implementation details.
There are two types of classes,
1. Normal classes : The class contains only normal methods.

```
class MyClass:
    def m1(self):
        print("....")
```

2. Abstract class : The class contains some abstract methods.
case 1: The class contains at least one abstract method is called abstract class.

```
class MyClass(ABC):
    @abstractmethod
    def wish1(self):
        pass
```

case 2: The below abstract class contains zero abstract methods.

```
class MyClass(ABC):
    def m1(self):
        print("....")
```

Note : abstract class may contains abstract methods or may not contains abstract methods, but for the abstract classes object creations are not possible.

Q74. What is the difference between Python Arrays and lists?

Ans: *Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. Arrays will consumes far less memory than lists.*

Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

```
import array
homo = array.array('i', [1, 2, 3])
for i in homo:
        print(i)

#TypeError: an integer is required (got type str)
hetro = array.array('i', [1, "ratan", 3])
for i in hetro:
        print(i)
print("*******")
data = [1, 2, 'ratan']
for i in data:
  print(i)
```

Q75. What is docstring in Python?

Ans : *Documentation string or docstring is a multiline string used to document a specific code segment. The docstring should describe what the function or method does.*

Declaring Docstrings:

The docstrings are declared using "'triple single quotes"' or """triple double quotes""" just below the class, method or function declaration.All functions should have a docstring.

Accessing Docstrings:

The docstrings can be accessed using the __doc__ method of the object or using the help function.

```
def my_function():
        '''
                List is mutable can change
                tuple is immutable can not change
                set is mutable can change
                dict is mutable can change
        '''
if __name__ == "__main__":
        print("Using __doc__:")
        print(my_function.__doc__)
        print("Using help:")
```

```
help(my_function)
```

Q76. How to print the most common occurred character in string/list/tuple?

Ans: To get the most common data use Counter class in collections module.

```
from collections import Counter
print(Counter("durgaduu").most_common())
print(Counter("durgaduu").most_common(1))
print(Counter("durgaduu").most_common(2))

print(Counter([10,20,10,10,30,10]).most_common())
print(Counter([3,4,6,7,6,7,4,4,4]).most_common(1))
print(Counter([3,4,6,7,6,7,4,4,4]).most_common(2))
```

Q77. How to take the command line arguments in python?

Ans: *The arguments taking through command prompt at runtime is called command line arguments.*

The command line argument separator is space.All the Command line arguments are taken as list.

All Command line arguments taken by default String format.

```
import sys
print(sys.argv)
print(sys.argv[1]+sys.argv[2])
print(int(sys.argv[1])+int(sys.argv[2]))

D:\>python first.py 10 20 30 40
['first.py', '10', '20', '30', '40']
1020
30
```

Q78. How to get some random names in python?

Ans:

To get the some random names in python use names module.

To install the names module use pip command.

```
D:\>pip install names

import names
for x in range(5):
        print(names.get_first_name())

for x in range(5):
        print(names.get_last_name())

for x in range(5):
```

```
print(names.get_full_name())
```

Q79. What is the purpose of assertion in python?

Ans: *Assertion is used to debug the application.*

Assert statements are a convenient way to insert debugging assertions into a program.

bebug : identifying the error & rectify the error.

```
statuscode = int(input("Enter a status code:"))
assert statuscode==200,"we got fail response code"
print("Application working fine")
```

Q80. Explain about logging in python?

Ans:

The log files will track the error messages
The log file contains the information with time stamp.
Generally errors are 5 types

1. CRITICAL	*:: 50*	*::Represents a very serious problem that needs high attention*
2. ERROR	*:: 40*	*:: Represents a serious error*
3. WARNING	*:: 30*	*:: Represents a warning message ,some caution needed. it is alert to the programmer*
4. INFO	*:: 20*	*:: Represents a message with some important information*
5. DEBUG	*:: 10*	*:: Represents a message with debugging*

```
logging.basicConfig(filename='log.txt',level=logging.ERROR)
        this file contains ERROR & CRITICAL
logging.basicConfig(filename='log.txt',level=logging.INFO)
        this file contains ERROR & CRITICAL & WARNING & INFO
```

To add the specific message use below functions in loggining module.

```
logging.debug("Debug message description here")
logging.info("info message description here")
logging.warning("warning message description here")
logging.error("error message description here")
logging.critical("critical message description here")
```

Q81. Explain about the encapsulation?

Ans: *The process of binding the data is called encapsulation.*

The process of wrapping methods & data as single unit.

class is a encapsulation mechanism because class binding methods & variables as a single unit.

module is a encapsulation mechanism because the module is binding classes, variables, functions.

Q82. What are the modifiers/scopes in python?

Ans: *public modifier*

our data is by default public.
public can be used for classes , functions , variables
public Permission all packages & modules can access the public data.

private modifier:
To represent private use two underscores(__) at starting :: __num = 10
private is only for variables , functions
private Permission only with in the class even child can not access the parent private data.

protected modifier:
To represent protected use one underscores :: _num = 10
It is only for variables , functions
with in the package all modules can access + outside pkg also we can access but only in child classes.

Q83. How to perform operations on image data?

Ans: *To perform operations on image data us pillow module(PIL)*
Install the pillow using pip command.

```
D:\>pip install pillow
```

```
from PIL import Image
try:
    original = Image.open("Desert.JPG")
    print(original.format, original.size, original.mode)
except:
    print ("Unable to load image")
```

Q84. What are negative indexes and why are they used?

Ans: *Negative indexes are the indexes from the end of the list or tuple or string.*
data[-1] means the last element of array.

To work with starting value use positive indexing,
To work with ending value use negative indexing.

```
data = [1, 2, 3, 4, 5, 6]
#       -6 -5 -4 -3    -2 -1
print(data[-1])        #get the last element
print(data[-2]) #get the second last element
print(data[-4:-1])
print(data[-4:])
print(data[ :-1])
print(data[ : ])

print(data[ : :-1])

name = "ratan"
print(name[ : :-1])
```

Q85. How to get the ascii characters & numbers in python?

Ans:

To get the ascii characters & numbers use string module.

```
import string
print(string.ascii_lowercase)
print(string.ascii_uppercase)
print(string.ascii_letters)
print(string.digits)
print(string.ascii_letters+string.digits)

print(string.octdigits)
print(string.hexdigits)
```

Q86. How can you generate random numbers?

Ans: *Python provides a module called random using which we can generate random numbers.*

We have to import a random module and call the random() method as shown below:

The random() method generates float values lying between 0 and 1 randomly.

```
import random
#both functions will give int data
print(random.randint(1,50))
print(random.randrange(5,25,5))

#both functions will give decimal values
print(random.random())                    # 0.0 to 1.0
print(random.uniform(2,8))                # specefic range
```

Q87. How to get the mean, median, mode, stdev in python?

Ans: *To work with above functions python given the module is called statistics.*

```
import statistics
print(statistics.mean([2,5,6,9]))
```

```
            print(statistics.median([1,2,3,8,9]))
            print(statistics.median([1,2,3,7,8,9]))

            print(statistics.mode([2,5,3,2,8,3,9,4,2,5,6]))
            print(statistics.stdev([1,1.5,2,2.5,3,3.5,4,4.5,5]))
```

Q88. How to creat & extract the zip file in python?

Ans: *To work with the zip file use zipfile module.*

zipfile is compressed file contains the data.

While creating & reading the data from zip file, the data should present.

case 1: write the files to zip.

```
            from zipfile import ZipFile
            zf = ZipFile('ratan.zip', mode='w')
            zf.write("ramu.txt")
            zf.write("anu.txt")
            zf.write("abc.txt")
            print("Zip file is creates")
```

case-2: Read the data from zip file.

```
            from zipfile import ZipFile
            zf = ZipFile('ratan.zip', mode = 'r')
            res = zf.read("abc.txt")
            print(res)
            res1 = zf.read("ramu.txt")
            print(res1)
```

Q89. How to perform permutations & combinations in python?

Ans: *To perform permutations & combinations in python use itertools module.*

```
    import itertools
    # we will get factorial value 2! = 2   3!=6  4!=24
    for p in itertools.permutations('AB'):
            print(p)
    for p in itertools.permutations('ABC'):
            print(p)
    for p in itertools.permutations('ABCD'):
            print(p)

    for p in itertools.permutations('12'):
            print(p)
    for p in itertools.permutations('123'):
            print(p)

    print(list(itertools.combinations([1,2,3],2)))
```

Q90. What is the difference between thread & process?

Ans: *#Thread vs. process:*

process is heavy weight created by operating system.
thread is light weight created by python virtual machine.
one process contains multiple threads every thread in independent.
ex: browser is a process
it contains multiple tabs these all are threads. every tab is independent.

Multi tasking :There are two types of multi tasking

1. process based multi tasking :
a. Every request one process is created.
b. if number of request increased the processes are increased it effects on performance.

2. Thread based multi tasking:
a. Every request one thread is created.
b. if number of request increased the threads are increased it improves performance because thread is light weigh.

Q91. What is Multithreading? What are the ways to create a Thread?

Ans: *Multithreading : create the multiple threads run the threads parall.*
There are two modules support multithreading concept.
1. thread (Deprecated) 2. threading
We can build the threading concept using functions & classes.

case-1: Function based threading.

```
import threading
def print_square(num):
  print("Square:",num * num)
def print_add(num1,num2):
  print("Addition: ",num1+num2)
if __name__ == "__main__":
        t1 = threading.Thread(target=print_square,args=(10,))
        t2 = threading.Thread(target=print_cube,args=(5,10))
        t1.start()
        t2.start()
```

case 2: class based threading.

```
        from threading import Thread
        class MyThread(Thread):
                def run(self):
                        print("Good Morning Threading")
        if __name__ == "__main__":
```

```
t = MyThread()
t.start()
```

Q92. What are the Thread states/life cycle in Python?

Ans: *Thread states : Every thread contains three states*

a. initial state
b. started state
c. stopped state

```
from threading import Thread
import time
def add(num1,num2):
        print(num1 + num2)
if __name__ == "__main__":
        t = Thread(target=add,args=(10,20),name="ratan")
        print(t)          # initial
        t.start()
        print(t)          # started
        time.sleep(2)
        print(t)          # stopped
```

Q93. What is thread pool in python? what is the advantage of thread pooling?

Ans:

case 1 : application without thread pooling : threads recreated

```
import threading
import time
def disp():
                print(f"running:{threading.currentThread().getName()}")
                time.sleep(1)
if __name__ == "__main__":
        for x in range(8):
                threading.Thread(target=disp).start()
```

in normal approach every request thread is created if number of requests are increased number of threads are increased it effects on performance.

in this is approach threads are recreating it effects on performance.

To overcome above problem, use thread pool.

thread pool approach contains the pool of threads these same threads are reused by multiple request.

Thread pool size : 10 client is requetsed thread: 28

first time : 10 pool threads used after that released back to pool.

second time : 10 same pool threads used after that released back to pool.

third time : 8 same pool threads used after that released back to pool.

if no threading pooling it will create & process 28 threads.it reduce the performance.

case 2: application with thread pooling : threads reused

```
import threading
import time
import concurrent.futures
def disp():
                print(f"running:{threading.currentThread().getName()}")
                time.sleep(5)
if __name__ == "__main__":
        executor = concurrent.futures.ThreadPoolExecutor(max_workers=3)
        for x in range(8):
                executor.submit(disp)
```

Q94.What is the purpose of pandas? What Are The Different Types Of Data Structures In Pandas?

Ans:

Pandas is the most popular python library that is used for data analysis & manipulate.

Pandas is a high-level data manipulation tool developed by Wes McKinney.

Panda library supports two major types of data structures,

a. Series
b. DataFrames

Both these data structures are built on the top of NumPy.

install the pandas: pip install pandas

In python pandas we will write the dataFrame to CSV files we will analyze the data.

We will run the pandas code using Jupiter note book.

https://jupyter.org/try
goto the above link : click on try classsic notebook
https://www.anaconda.com/products/individual

The jupyter file extension is :.ipynb file
open the ipynb file using jupyter.

Q95. What is Series in pandas? What are the Different ways create the Series pandas?

Ans: *A series object can contain any type of data.*

Series is a one-dimensional array that can hold data values of any type (string, float, integer, python objects, etc.). It is the simplest type of data structure in Pandas here, the data's axis labels are called the index.

Creating Series:

```
import pandas as pd
```

```
#Creating Series with predefined index
data = [3,2,5,7,9,3,2]
res = pd.Series(data)
print(res)
#Creating Series with user-defined index
s= pd.Series(["ramu","anu","raju","rani"],['i','ii','iii','iv'])
print(s)
#Creating Series using dict data type.
if we are passing dict to Series
The dict keys becomes indexes, The dict values becomes data series
        s = pd.Series({1:"a",2:'b',3:'c',4:'d'})
        print(s)
#Combineing two series
s = pd.Series((1,2,3)) + pd.Series([10,20,30])
print(s)
```

Q96. What is DataFrame in pandas? What are the different ways to create DataFrame in pandas?

Ans: *A DataFrame is a 2-dimensional array in which data is aligned in a tabular form with rows and columns. With this structure, you can perform an arithmetic operation on rows and columns.*

A DataFrame is a collection of Series.
when we make the dict object as a DataFrame
The keys becomes column names[headers].
The values becomes the data.

```
import pandas as pd
names = ['ratan', 'anu','durga','sravya']
df = pd.DataFrame(names)
print(df)

#DataFrame with predefined index
data = {'names':['ratan','ramu','raju'],
                'numbers':[1,2,3],
                'fruits':['apple','mango',"orange"]}
df = pd.DataFrame(data)
print(df)

#DataFrame with Userdefined index
data = {
        'A': [1,4,7],
        'B': [2,5,8],
        'C': [3,6,9]
}
df = pd.DataFrame(data,index=['i','ii','iii'])
print(df)
```

Q97. How to Append the one data frame to another?How to write the dataframe to CSV file?

Ans: To add one data frame to another use append() function it is deprecated.
To add one data frame to another use pandas.concat() function.
To write the DataFrame to csv file use to_csv() function.

```
        import pandas as pd
        df1 = pd.DataFrame({
    "Company Name":["Google", "Microsoft", "SpaceX","Amazon","Samsung"],
    "Founders":['Larry Page','Bill Gates','Elon Musk','Jeff Bezos', 'Lee Byungchul'],
    "Founded": [1998, 1975, 2002, 1994, 1938],
    "Number of Employees": [103459, 144106, 6500647,50320,67145]})
        print(df1)

        df2 = pd.DataFrame({
                'Company Name':['WhatsApp'],
                'Founders':['Jan Koum, Brian Acton'],
                'Founded': [2009],
                'Number of Employees': [5000]}, index=['i'])

        frames = [df1, df2]

        result = pd.concat(frames)
        print(result)
        result.to_csv("company.csv",index=False)

        res = df1.append(df2,ignore_index = True)
        res.to_csv("company.csv",index=False)
        print(res)
```

Q98. Define numpy in python? How to convert list to arrays? How to create Different array using numpy functions?

Ans: *NumPy is a python library used for working with arrays.*

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

install the numpy using pip command :: `pip install numpy`

```
#conversion of list into array. one dimensional
import numpy as np
a = np.array([24, 12, 57])
print(a)
# working with two dimensional
b = np.array([[1,2,3],         [4,5,6]])
print(b)
print(b.shape)
print(b[0, 0], b[0, 2], b[1, 1])
b[0, 0] = 10
b[1, 1] = 20
print(b)

a = np.zeros((2,2))            print(a)
b = np.ones((2,3))     print(b)
c = np.full((2,2), 7)   print(c)
d = np.eye(2)          print(d)
```

Q99. what is the purpose of matplotlip? How to create the pie chart?

Ans: *matplotlip is python module used plot the the graphs.*

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations.

```
from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.plot(x,y)
plt.show()

import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
institutes =[ "durgasoft","Satech","Gana Tech.","Pee Tech.","Dreams
Solu."]
values =[3803,638,150,374,296]
explode = [0.05,0,0,0,0]    #explode 1st i.e slice is separated by 0.05
distance
colors =["c","b","g","r","y"]
plt.pie(values, labels=institutes, explode=explode, colors=colors,
autopct="%0.1f%%")
```

```
plt.show()
```

Q100. How to create the GUI Applications in python?
Ans:

tkinter module used to design GUI application.
tkinter module used to design GUI in standalone application.
In standalne application GUI is developed using tkinter module.
install the tkinter using pip command

```
pip install tkinter
```

We can do the alignment on the frame in three ways.

a. Pack() : Auto adjustment.
b. grid() : Rows & column
c. place() : x & y axis location.

case 1:

```
import tkinter
window = tkinter.Tk()
window.title("GUI Application")
window.geometry("400x200")
window.mainloop()
```

case 2:

```
import tkinter as tk
window = tk.Tk()
window.title("GUI")
window.geometry("400x200")

tk.Label(window,text="Login Application....").grid(row=0,column=1)

tk.Label(window,text="User Name:").grid(row=1,column=0)
tk.Entry(window).grid(row=1,column=1)

tk.Label(window,text="User Password:").grid(row=2,column=0)
tk.Entry(window,show='*').grid(row=2,column=1)

tk.Button(window,text="Login", fg="red", bg="yellow").grid(row=3,column=1)
window.mainloop()
```

************Thank you :: MR. Ratan Sir **************