1, Write the features and application of python programming language

A: Features in python?

1, Easy to code: Python is a High-level programming language
2, free and open source
3, object-oriented language
4, GUI programming support
5, High-level language
6, Extensible features
7, Python is portable language
8, Python is integrated language

2, List the various operators supported in python?

A: Python language supports the following types of operators

They are:

1. Arithmetic operators
2. comparison operator (or) Relation
3. Assignment operators
4. Logical operators
5. Bitwise operators
6. Membership operators
7. identity operators.

1. Arithmetic operators:

Arithmetic operators are used to perform the mathematical operations like Addition substraction, Multiplication, etc

Ex: a=10
b=2
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
print(a%b)
print(a**b)

output:

12
8
20
5.0
5
0
100

It can also applied to str type which is considered as string concentenation from '+' operator both should be string

```
>>> 'Siva'+'3'
>>> 'Siva3'
```

'*' from one should be string and other value is int

```
>>> 2*Siva
'SivaSiva'.
```

2. Relational operators

comparison operators are used to compare values. It return either True or false according to the condition

Ex: a=10
b=20
print('a<b';a>b)
print('b<b';a<b)

output:

a>b False
a<b True.

3. Assignment operators

Assignment operators are used in python to Assign values in variable.

Ex: a=10
b=20
c=30
d=40
Print(a)
O/P:
10.

4. Logical operators: Logical operators are the and, or, not operators. We can apply these operators for all types.

For Boolen types:

and → If Both Arguments are True then only result is True
or → If atleast one Argument is True then result is True
not → If True the False, If false is True.

Ex:
>>> True and False
False
>>> True or False
True
>>> not True
False.

5. Bitwise operators: Bitwise operator act on operands as if they were strings of binary digits. They operator bit by bit. as name indicates Bitwise. these operator are applicable only for Int and Boolean type.

operators: &, |, ~, ^, >>, <<

Ex: >>> 4&5
4
>>> 4|5
5
>>> 4^5 → 1
>>> 10<<2

40
>>> 10>>2
2

>>> ~4
-011

Special operators?

Python language offers some special type of operators like

i, identity operators

ii, Membership operators.

6. Membership operators?

in and not in are the membership operator in Python they are used to test whether a value or variable is found in a sequence.

Ex: list1 = [10, 20, 30]

Print('10 in list1:', 10 in list1)

Print('30 not in list1:', 30 not in list1)

output:

10 in list1: True

30 not in list1 = False

i, Identity operators?

is and is not are the identity operators in Python. they are checked if two values are located on the same part of the memory or not. if two variable that are equal does is True does not imply they are identical.

Ex: a=10

b=10

c=20

Print('a is b:', a is b)

Print('b is not c:', b is not c)

>>> a is b

True

>>> b is not c

True.

3, # Program to display x^y

Ex: 2^5 = 32

Sol
```
x = int(input('enter x value'))
y = int(input('enter y value'))
Print(pow(x,y))
```

Ex: >>> 2 ^ 5

32

4. # Program to greet the user.

Sol
```
first_name = input('enter your first name:')
last_name = input('enter your last name:')
    Print("Greetings!!!", first_name, last_name)
```

5. Program for conversion of given distance into various units

Sol
```
# cm = centimeters
# m = meters
# km = kilometers
# ft = feet
# inc = inches
km = float(input('enter the distance between cities in kilometers:'))
m = km * 1000
ft = km * 3280
Inc = km * 10.079
cm = km * 100000
Print('distance in meters', m)
Print('distance in feets', ft)
Print('distance in inches', inc)
Print('distance in centimeters', cm)
```

7, Program to convert temperature from Fahrenheit to celsius

Sol
```
# F = temperature in fahrenheit
# c = temperature in celsius
F = int(input('enter the temperature of city:'))
C = (F-32)* 5/9
Print('temperature in Fahrenheit:', F)
Print('temperature in celsius:', C)
```

8, Program to convert temperature from celsius to Fahrenheit

Sol
```
# F = temperature in fahrenheit
# c = temperature in celsius
C = int(input('enter the temperature of city:'))
F = 1.8C+32
Print('Temperature in celpius:', C)
('Temperature in fahrenheit:', F)
```

Explain with example of Membership and identity operators in Python?

Membership operators? in and not in are the membership operators in Python they are used to test whether a value or variable is found in a sequence

In operator? in operator is used to check if value is exists in a sequence or not. Evaluate to true if it find a variable in a specified sequence and false other wise.

not in operator? Evaluates to true if it does not find a variable in a specified sequence and false otherwise.

Ex: list1 = [10, 20, 30, 40, 50]
print('10 in list:', 10 in list1)
print('70 in list1:', 70 in list1)
print('100 not in list1;' '100 not in list1)

output:

10 in list1 : True
70 in list1 : False
100 not in list1 : True.

Identity operators:

is and is not are the identity operators in Python they are used to check if two values (or) variables are located on the same part of the memory or not. Two variable that are equal does not imply that they are identical.

1. 'is' operator: Evaluates to true if the variable on either side of the operator point to the same object and False otherwise.

2. 'is not' operator:

Evaluates to false if the variables on either side of the operator point to the same object and True otherwise

Ex: a=10
b=20
c= 20
print('a is b:', a is b)
print('b is not c:', b is not c)

output

a is b : True
b is not c : True.

9. Explain various data types available in python?

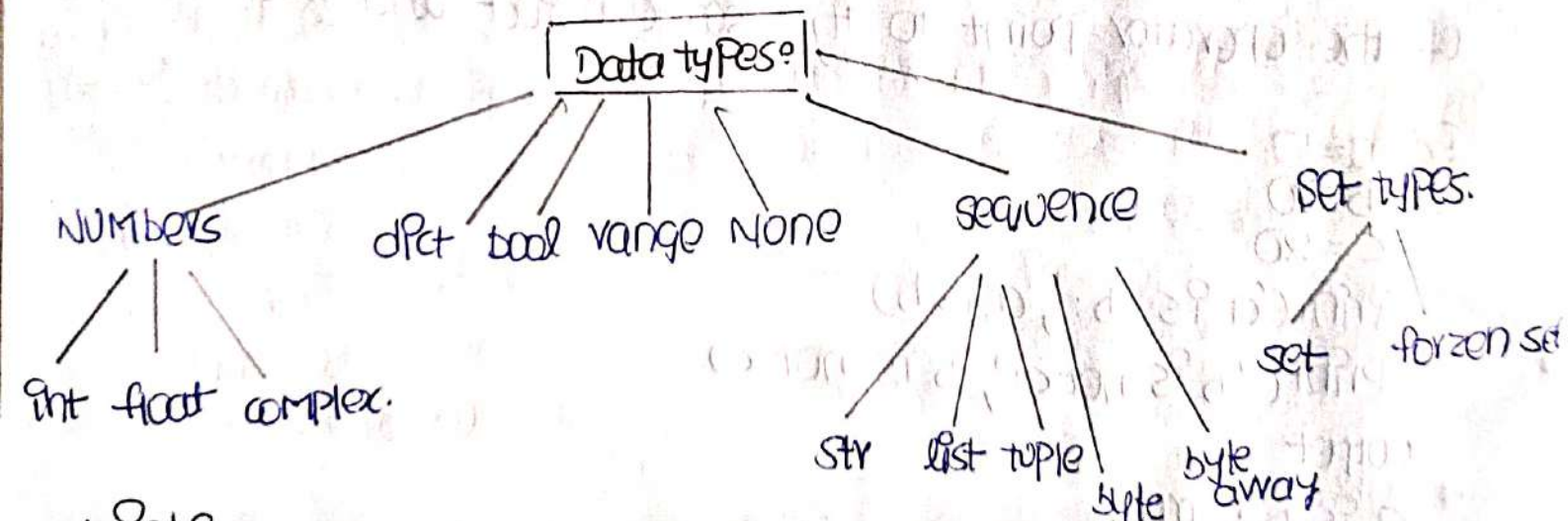
Sol Data types are the classification or categorization of data items.

Python supports the following built-in data types.

1. Scalar types : int, float, compler, bool, NONE
2. Sequence types : string, list, tuple
3. Mapping types : Dictionary
4. Set types : set, frozenset
5. Mutable and immutable types : Number, string and tuples immutable list, dictionary, set and user-defined classes.

they are

1. int
2. float
3. compex
4. bool
5. str
6. bytes
7. bytearray
8. List
9. tuple
10. range
11. set
12. frozenset
13. dict
14. None.



1, int?

We can use int data type to represent whole number

Ex: a=20
Print(a)
Print(type(a))

op: 20
<class 'int'>

2. We can represent int values in the following forms

1, Binary
2, octal
3, Decimal
4, Hexa-decimal.

Binary: It uses two values 0 and 1. It uses Base 2 (0,1). Binary literal should be prefixed with 0b or 0B

Ex: a=0b1001
print(a)
O/P:
9

octal: It uses eight values 0 to 7. It uses Base-8 (0-7). octal literal should be prefixed with 0o or 0O

Ex: a=0010
print(a)
O/P:
8

Decimal:
It uses ten values 0 to 9. It uses Base-10 (0-9). Decimal literal can be represented without any prefix leading zeros in decimal integer literals are not permitted

a=10
print(a)
O/P
10

Hexadecimal:
It use 16 values they are 0 to 9, A to f (or) a to f.
It uses Base-16 (0-9, A-F). Hexadecimal literal should be prefixed with 0x or 0X

Ex: a= 0x10
print(a)
O/P
16

2. Float: We can float data types to represent fractional decimal or scientific notations. In decimal number, decimal point is used while in scientific notation exponent is used

Ex: a=52.83
Print(a)
Print(type(a))
output:
52.83
<class 'float'>

3. Complex:
Complex numbers are written in the form of x+yj. where x is a real number and y is a imaginary part

Ex: x=10+20j
Print(x)
Print(type(x))
O/P:
10+20j
<class 'complex'>

4. Bool:
We can use this data type to represent boolean values. It represent one of the two values. True or False. Internally python represent true as 1 and false as 0

Ex: x=True
Print(x)
Print(type(x))
O/P:
True
<class 'bool'>

5, Str:
A string is a sequence of characters. string can be created by enclosing characters inside in a single quote(' ') or double quotes (" "). Even triple quotes (''' ''' or """ """)

2, But generally used to represent the Multiline strings and docstrings.
3, In Python there is no data type for character.
4, Python internally uses single quotes representation. so it is recommended to use single quotes only

Ex: S = 'Hello'
Print(S)

o/p:
Hello

Accessing characters in a string?

1. We can Access the individual characters using indexing and a range of character using slicing.

Accessing characters using index?

1. Python Allow +ve and -ve indexing for its sequence.
2. +ve index starts from 0
3. -ve index starts from -1
4. Trying to Access a character out of its index range will raise index Error.
5. the index must be an integer. We can't use floats or other types. this will result into the Error.

Accessing characters using slicing operators?

1. We can Access a range of items in a string by using the slicing operator.
2. S[begin:end] → Return substring from begin index to end-1 index
3. S[begin:] If we are not specifying end, then it will consider begin to last character of string.
4. S[:end] → If we are not specifying begin, we will get from starting character end-1 index of str.
5. [:] If we are not specifying begin and end, then it entire string will be taken.

Ex:

0	1	2	3
A	J	a	y

6. Bytes:

It represent group of bytes numbers. Every value should be from 0 to 256. only bytes type is immutable

Ex: a = [10,20,30,40]
b = bytes(a)
Print('type of b is: ' type(b))
Print(b[1])

o/p:

type of b is: <class 'bytes'>.
20

7. Byte array:

bytes and bytearray are same, the only difference in bytes is immutable and byte array is mutable. Every value should be from 0 to 255 only.

Ex: a = [10,20,30,40]
b = bytearray(a)
Print('type of b is:', type(b))

o/p:

type of b is: <class 'bytearray'>

8, List:

1, list is an ordered sequence of items.
2, It is one of most used data type in Python, It is very flexible
3, It is mutable
4, It is hetergeneous, object are allowed
5, Duplication values are allowed
6, slice(:) operator is applicable in list

Ex: l = [10, 20, 30, 15.2, 'siva', None]
l[1:5:2]

output:

[20, 15.2]

9. tuples:
1. tuple is a immutable type
2. A tuple is created by placing all the items inside parenthesis () are separated by commas.
3. Parenthesis is a optional. However, it is a good practice to use them.
4. Indexing and slicing is possible

Ex: a = (10, 20, 30, 40, 50, 60)
print(a[1])

o/p:
20

10. range: It is used to generate a sequence of numbers

Syntax: range(start, stop, step):

Ex: for i in range(1, 11)
print(i, end = " ")

o/p:
1 2 3 4 5 6 7 8 9 10

11. None:
the None keyword is used to define a null value, or no value at all. None is a datatype of it's own (None type) and only None can be None

Ex: >>> a = None
>>> type(a)
<class 'None type'>

3) Momentum is calculated as $e = mc^2$, where m is the mass of the object and c is its velocity. Write a program that accepts an object's mass and velocity and display s it's Momentum?

Ans Where M is
the mass of the object and c is it's velocity.

Program:

```
m = float(input("Enter Mass:"))
c = float(input("Enter velocity:"))
e = m*c**2
Print("the momentum of the particle is", e)
```

11, Write a program that calculates number of seconds in a day.

Ans Program:

```
a = int(input('enter the number of days'))
# one day has 24 hours, 60 minutes in a hour and 60 seconds in a
minute
Seconds = a*24*60*6
Print('the number of seconds on given number of days is', Seconds
```

12, Write a program the prompt the user to enter first name and the last name. then display following message.

Sol
```
a = input('Enter your name:')
Print("Hello", a)
Print("Welcome to python!")
```

13, Explain with examples of Bitwise operators in python?

Ans Bitwise operators:

In python Bitwise operators act an operands as if they were string of binary digits. they operators bit by bit. as name indicates Bitwise. these operators are applicable only for int and Boolean type.

(or)

In Python, Bitwise operators are used to perform bitwise calculations on integers. the integers are first converted into binary and then bit by bit operators are performed.

Types of Bitwise operators with Example.

operator	Description	syntax
&	Bitwise AND	x&y
|	Bitwise OR	x|y
~	Bitwise NOT	~x
^	Bitwise xOR	x^y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Eg >>> 485
4
>>> 415
5
>>> 4^5
1

Bitwise complement(~) we have apply complement for all bits.

1. the most significant bit act as sign bit 0 value represent +ve no. 1 represent -ve value.
2. +ve number directly represented in the memory where as -ve number will be represents indirectly in 2's complement.

>> ~A = 10
>> ~A
-11

Left shift operator (<<)? After left shift the empty cells we have to fill with 0

Ex? >>> 10<<2
40

Right shift operator (>>)? after right shift the empty cell we have to fill with sign bit (0 for +ve 1 for -ve)

Ex? 10>>2
2

15, To print Area of Triangle using Heron's formula?

A? a=int(input('length of one side of the triangle'))
b=int(input('length of second side of the triangle'))
c=int(input('length of third side of the triangle'))
s=(a+b+c)/2
area=(s*(s-a)*(s-b)*(s-c))**0.5
print('the area of triangle is', area)

17, Write a program to calculate the total amount of money in the piggy bank, given the coins of Rs.100, Rs.50, Rs.20, Rs.10

A? # to find the total amount of money in piggy bank given the coins of rs100, rs50, rs20, & rs10
rs100=int(input("enter no. of rs100 coins:"))
rs50=int(input("enter no. of rs50 coins:"))
rs20=int(input("enter no. of rs20 coins:"))
rs10=int(input("enter no. of rs10 coins:"))
total int
total=100*rs100+50*rs50+20*rs20+10*rs10
print("total money:", total)

18, Write the rules for an identifier? Write a program to print the digit at one's place of a number?

Ans Rules of an Identifier

1 Identifier is a combination of letters in lowercase(or) uppercase or digits or an underscore '_'

i, the lowercase letters (a to z)

ii, uppercase letters (A to z)

iii, digits (0 to 9)

iv, underscore (_)

v, An identifier cannot start with a digit

vi, It can be an any length.

vii, we can't use special symbol as !, @, #, $, %, etc

viii, keywords can't be used in identifiers

ix, It is a case sensitive

x, classes can be used as identifiers, But it is not recommended to use.

```
n=int(input("give me a number :"))
print(n)
# u= number at unit's place
u=n%10
print("the number at unit's place is:", u)
```

Q, the provided code stud reads two integers from stdin, a and b,
write a python program to print three lines where : i. the first line
contains the sum of two numbers
ii, the second line contain difference of two numbers
iii, the third contains the product of two numbers

Sol
```
a=int(input('enter the number'))
b=int(input('enter the second number'))
c=a+b
print(c)
d=a-b
print(d)
e=a*b
print(e)
```

output?

a=3
b=2

o/p?

5
1
6

1, Explain the decision-control statement in python?

Ans Control flow statements are required to alter the flow of program execution based on the conditions.

they are three types:

1. Condition (or) Selection (or) Decision-Making statements
2. Iterative (or) Repetition (or) Looping statements
3. Transfer (or) Branching statements.

Decision-Making statements:

1, If:

If is the most basic control flow statement in python. It enables the program to execute a certain part of the program if given condition in an if statement is satisfied

Syntax:

```
if expression:
    statement 1
    :
    statement n
```

Ex: a=5
b=10
if a<b:
print("a is smaller than b")

O/P:
a is smaller than b

2, else:

Any expression evaluated to True leads the execution of it's Body statement whereas else body statement is executed once expression evaluates to false. It

Syntax:

```
if exp:
    statement 1 }
    .....
    statement n }
else:
    statement n }  # execution is done
```

Ex:
```
a=10
b=15
if a<b:
  print("a is smaller than b")
else:
  print("a is greater than b")
```
op:
a is smaller than b.

if, elif:
elif statement is used to chain multiple if statements one after another. The program executes the body of the first if expression is TRUE. The program executes the body of else if any one of the if expression is TRUE. If none of the if expression is TRUE, else the body of else is executed.

Syntax:
```
if exp:
    statement1
    ....
elif exp:
    ....
    ....
elif exp: ...
    ....
else:
    statement n
```
Ex:
```
food=input("Enter your favourite food item:")
if food == "biryani":
    print("It will deligous in paradise")
elif food == "pizza":
    print("It will deligous in dominos")
elif food == "burger":
    print("It will be delicious in Mc donalds")
else:
    print("I don't about the food item")
```
Output:
Enter your favourite food items: Pizza
It will be deligous in dominos

2, Write a program to determine whether character entered is a vowel or not?

Ans

```
character = input()
lower_case_character = character.lower()
if len(character)>1 :
    Print('please enter only a single character')
else:
    if lower_case_character=='a' or lower_case_character=='e' or
    lower_case_character=='i' or lower_case_character=='o' or
    lower_case_character=='u':
        Print('vowel')
    else:
        print('constant')
```

3, Write a program to find the greatest number from three numbers

Ans

```
a = int(input())
b = int(input())
c = int(input())
if a>b and a>c:
    print(a)
if b>a and b>c:
    Print(b)
else:
    Print(c)
```

4, Write a program to calculate tax given the [illegible]

i, If income is less than 1,50,000 then no tax
ii, If taxable income is 1,50,001 - 300,000 then charge 10% tax
iii, If taxable income is 3,00,001 - 500,000 then charge 20% tax
iv, If taxable income is above 5,00,000 then charge 30% tax

Sol

```
a = int(input('Enter income:'))
if a < 150000:
  Print ('no tax')
if 150001 < a < 300000:
  Print ('your tax is', 0.1*a)
if 300001 < a < 500000:
  Print ('your tax is', 0.2*a)
if a > 500001:
  Print ('your tax is', 0.3*a)
```

5, Write a program to enter the marks of a student in four subjects then calculate the total and aggregate, and display the grade obtained by student.

i, If the student scores an aggregate greater than 75%, then the grade is distinction
ii, If aggregate is 60>= and <75 → First division
iii, If aggregate is 50>= and <60 → second division
iv, If aggregate is 40>= and <50 → third division
v, else, grade is fail.

Sol

```
lac = int(input())
bee = int(input())
PP = int(input())
che = int(input())
```

```
total= lac+bee+pp+che
agg=(total/400)*100
print(agg)
if agg>=75:
    print('grade is first division')
if 50<=agg<60:
    print('grade is second division')
if 40<=agg<50:
    print('grade is third division')
if agg<40:
    print('fail')
```

6 write a program to calculate the roots of a quadratic eqn?

sol

```
a=int(input("enter an integer"))
b=int(input("enter an integer"))
c=int(input("enter an integer"))
d=(b**2)-(4*a*c)
if d>0:
    x1=-b+(math.sqrt(d))/(2*a)
    x2=-b-(math.sqrt(d))/(2*a)
print("roots are real and distinct",x1,x2)
elif d==0:
    x1=-b/2*a
    x2=-b/2*a
print("roots are equal",x1,x2)
else:
print("roots are imaginary")
```

7) Explain the different types of loops in Python with example?

Ans looping statements:

1) for:

for loop is used to iterate over any iterable objects such as list, set, string etc

2. It visit each item in an iterable serves unless the keyword break or continue is encountered and assign a variable that specify for statement

Syntax:

```
for item in iterable:
    statements
```

Ex: name = "SIVA"

```
for ch in name:
    Print(ch)
```

output:

S
I
V
A

2) while

the body of while loop is executed as long as the expression evaluates to true

Syntax:

```
while expression:
    statements
```

Ex:

```
n=int(input("Enter number:"))
sum=0
i=1
while i<=n:
    sum=sum+i
    i=i+1
Print("the sum of first", n, "numbers:", sum)
```

output:

```
Enter number:10
The sum of first 10 number is:55
```

Nested for loop: putting one loop inside another loop

Ex:

```
for i in range(3):
    for j in range(3):
        print("i= ", i, "j= ",j)
```

o/p:

```
i=0 j=0
i=0 j=1
i=0 j=2
i=1 j=1
i=1 j=2
i=2 j=0
i=2 j=1
i=0 j=2
```

i=1 j=0

8, Write a program to calculate the LCM and GCD of two numbers?

Sol

```
num1=int(input("Enter first Number:"))
num2= int(input("Enter second Number:"))
  x= num1
  Y= num2
   while(y):
       x, y=y, x % y
   print("GCD of", num1, "and", num2, "is",x)
   lcm=(num1*num2)//x
   print("LCM of", num1, "and", num2,"is", lcm)
```

o/p:

9, Explain break and continue statement with example

A: Transfer (or) Branching Statements:

Break:

In Python, the Break statement provides you with the opportunity to exit out of the loop when an external condition is triggered. You will put the break statement within the Block of code under your loop statement, usually after a conditional if statement.

Ex:
```
for i in range(10):
    if i==7:
        print("value is 7 Break the loop")
        break
        print(i)
```

output:
```
0
1
2
3
4
5
6
value is 7 Break the loop
```

Continue:

Whenever a continue statement is encountered during a program execution, the execution flow skips the current iteration and goes to next iteration

Ex:
```
for i in range(10):
    if i%2!=0
        continue
    print(i, end=" ")
```

output:
```
1 3 5 7 9
```

Pass statements:

1. the Pass statement a null statement or) none
2. When the pass statement is executed, nothing happens.
3. the Pass statement is used as a placeholder for future code. It is used to Pass the executed program

Ex: for i in range(10):
    if i%2==0
      Print(i, end = ' ')
    else:
      Pass

O/P:
0 2 4 6 8

10, Write a program to read a number and then calculate the sum of its digits?

Sol
```
n=int(input())
Sum=0
while(n!=0):
    Sum=Sum+(n%10)
    n=n//10
    Print(sum)
```

11, Write a program to calculate the average of first n natural numbers using for loop?

Ans
```
num= int(input('How many numbers: '))
    total_sum=0
        for n in range(num):
            numbers= float(input('Enter number: '))
            total_sum+ = numbers
    avg= total_sum/num
Print('Average of', num, 'number is:', avg)
```

13) Write a program using for loop to print all the numbers from m to n there by classifying them as even or odd?

Sol

```
m= int(input())
n = int(input())
even = []
odd = []
for i in range (m,n+1):
  if i%2==0:
  even.append(i)
  else:
   odd.append(i)
  Print("even numbers are", even)
  Print("odd numbers are", odd)
```

14) Write a program to check whether a number is prime or not?

A) Program

```
num = int(input("Enter the value:"))
flag = False
if num >1:
      for i in range(2, num):
            if(num % i)==0:
                  flag = True
                  break
if flag:
   Print(num,"is not a prime number")
else:
   Print( num," is a prime number")
```

15, Write a program to print the following pattern?

```
0
1 2
3 4 5
6 7 8 9
```

Sol

```
n = int(input())
count = 0
for items in range(0, n):
    for items2 in range(0, items+1):
        print(count, end = ' ')
        count += 1
    print('')
```

16, Write a program to print following pattern:

```
1
2 2
3 3 3
4 4 4 4
```

Sol

```
n = int(input())
for i in range(1, n)
    for j in range(1, i+1)
        print(i, end = ' ')
    print('')
```

17, Write a python program to print inverted right triangle star pattern series using for loop.

Sol

```
n = int(input())
for i in range(n+1, 0, -1):
    print('*' * (i-1))
```

O/P

```
* * * * *
* * * *
* * *
* *
*
```

↓ Write a program that creates a list of numbers from 1 to 20 that are either divisible by 2 or divisible by 4?

Sol
```
div=[]
for i in range(1,21):
    if i%2==0 or i%4==0:
        div.append(i)
print(div)
```

2) Copying the content of one file to another file:

Sol Program:
```
first = input("Enter the name of first file: ")
second = input("Enter the name of second file: ")
f=open(first,'r')
text = f.readlines()
f.close()
f=open(second,'w')
for data in text:
    f.write(data)
f.close()
print("file copied successfully!")
```

Output:

3 write the built-in- function in set data type?

1. Add(): add() Method adds a given element to a set. If the element is already present, it doesn't add any element It return None.

2. update(): update() Method update the set, adding element from iterable. update() Method return None

3. copy(): copy() Method returns a shallow copy of set

4. remove(): this Method remove specified element from set set.remove(element)

5. discard(): discard() Method remove specified element from the set.

6. POP(): POP() Method remove an arbitary element from set and return the element removed.

7. clear(): clear() Method removes all element from the set. It doesn't return any value.

8. Difference_update(): It update the set calling difference_update() Method with difference of set.

9. symmetric_difference(): this Method returns the symmetric difference of two sets.

10. union(): this Method return a new set with distinct elements from all set.

11. issubset(): this Method return True if all element of a set are present in another set. If not, it return false.

12. Intersection_update(): It update the set calling intersection_update() Method with intersection of set

1, Write the built-in- function of tuples?

A: 1. len(): To find the length of tuple

2. count(): The count() Method returns the number of times the specified element appear in tuple.

3. index(): the index Method return the index of specified element in tuple

4. sorted(): To sort the element of tuple in default natural sorting order. It return list

5. sort(): sort() Method is not available for tuple, because tuple is immutable

6. Min(): Min() It is used to find minimum element of tuple

7. Max(): It is used to find the Maximum element of tuple

8. SUM(): It returns the arithMetic SUM of all items in tuple

9. any(): If even one item in tuple has a Boolean value of true then fn return True. other false.

10. all(): all() return true only if all item have Boolean value of true. otherwise, it return false.

3, Write a program to create a list of numbers in range 1 to 10. then delete all even numbers from list and print the final result?

Sol: list = [x for x in range(1,11)]
for numbers in list:
    if numbers % 2 == 0:
        list.remove(numbers)
print(list)

Output:

4, Write a program that creates a list of words by combining the word in two individual list?

Sol
```
a = list(input().split())
b = list(input().split())
c = a+b
Print(c)
```

output:

5, Write a program to remove all duplication elements from a list?

A:
```
list1 = list(input("enter elements"))
set1 = set(list1)
list2 = list(set1)
Print(list2)
```

output:

***7, Explain the following list methods with example?

1. list()
2. append()
3. pop()
4. extend()

A: 1, list(): the list insert() method inserts an elements to the list at specified index. the syntax of insert() method is list.insert()

Ex:
```
>>> list = [1,2,3]
>>> list.insert(4,5)
>>> print(list)
[1,2,3,4,5]
```

2. append(): the append() method in python adds a single item to existing list. It doesn't return a new list of items but will Modifty the original list by adding the item to end of list.

Ex:
```
>>> list = [1,5,7]
>>> list.append(20)
>>> print(list)
[1,5,7,20]
```

iii) pop(): pop() is an built-function in python that removes and returns last value from the list or given index vale.

2. If index is not given, then the last element is popped out and removed

Ex:
```
>>> list = [2.3,4,44,5]
>>> list.pop()
5
```

iv) extend(): extend() is an built function that adds the specified list elements to the end of current list. the extend() method extends the list by adding all items of list to an end.

Ex:
```
>>> list = ['apple', 'orange', 'mango']
>>> list2 = [1,8,0]
>>> list.extend(list2)
>>> print(list)
['apple', 'orange', 'mango', 1, 8, 0]
```

8, write a program that creates a list of 10 random integers. then create two lists - odd list and Even list then has all odd and even list in respectively?

Sol:
```
list = []
even = []
odd = []
for i in range(10):
    list.append(int(input()))
for x in list:
    if x%2==0:
```

```
        even.append(x)
    else:
        odd.append(x)
print('the odd numbers are:', odd)
print('the even numbers are:', even)
```

9, Explain map(), filter() and reduce() functions with example?

A: map()?

for every element present in the given sequence, apply some functionality and generate new element with required modification. For this requirement we should go for map() function.

(or)

the fn is used to apply a fn on all the elements of specified iterable and return map object.

syntax?

map(function, sequence)

Ex:

```
l = [1,2,3,4,5]
l1 = list(map(lambda x: 2*x, l))
print(l1)
```

output?

[2,4,6,8,10]

filter()?

the filter() method filters the given sequence with the help of a function that tests each element in sequence to be true or not

syntax?

filter(function, sequence)

Ex:

```
l = [0,5,10,15,20,25,30]
l1 = list(filter(lambda x: x%2==0, l))
print(l1)
l2 = list(filter(lambda x: x%2!=0, l))
print(l2)
```

output?

[0,10,20,30]

[5,15,25]

reduce()?

reduce() function reduces sequence of elements into a single element by applying the specified function

syntax?

reduce(function, sequence)

Ex? from functools import *

result = reduce(lambda x, y: x+y, range(1,101))

print(result)

output?

5050

11. Explain the following tuple operations?

i. Min()
ii. max()
iii. index()
iv. count()

A: Min()? Returns the element with minimum value from tuple

Ex? $t_1 = (34, 10, 15, 45, 60)$

print("Minimum element of tuple is:", Min(t1))

output?

Minimum element of tuple is: 10

Max()? It is used to find the maximum element of the tuple.

Ex?

$t_1 = (45, 35, 36, 20, 10)$

print("Maximum element of tuple is:", Max(t1))

output?

Maximum element of tuple is: 45

Index()?

the index() method return the index of specified element in tuple

Syntax? tuple.index(element, start, end)

Ex: t = ('s', 'u', 'r', 'e', 's', 'h')
print("Index of e is: t.index('e'))
output:
Index of e is: 3

count(): the count() method returns the number of times the specified element appears in the tuple

syntax:
tuple.count(element)

Ex: >>> t1 = (12, 34, 57, 7, 6, 12, 5)
>>> t1.count(12)
2

(2) Explain the following set operations with examples?
i) update()
ii) add()
iii) remove()
iv) pop()
v) copy()

A. update(): update() method updates the set, adding elements from other iterable. update() method return None.

syntax:
set.update(iterable)

Ex: >>> s1 = {10, 20, 30}
>>> s1.update({40, 50, 60})
>>> print(s1)
{50, 20, 40, 10, 60, 30}

Add(): Add() method adds a given element to a set. If the element is already present, it doesn't add any element. It returns None.

syntax:
set.add(element)

Ex:
```
>>> S= {"Ak", "Ram", "krishna"}
>>> S.add((10,20))
>>> S
{'Ak', (10,20), 'ram', 'krishna'}
```

remove():
This method removes the specified element from the set.

Syntax:
```
Set.remove(element)
```

Ex:
```
>>> S={10,20,30,40}
>>> S.remove(40)
>>> print(S)
{10,30,20}
```

POP():
POP() Method removes an arbitrary element from set and returns the element removed.

Syntax:
```
Set.POP()
```

Ex:
```
>>> S= {"Sri", "ram", "krishna", "Rayu"}
>>> print(S)
{"ram", "Sri", "krishna", "Rayu"}
>>> S.POP()
'ram'
```

copy(): copy() Method returns a shallow copy of the set.

Syntax:
```
set.copy()
```

Ex:
```
S1 = {10,20,30,40}
S2 = S1.copy()
Print("S1 is:", S1)
```

Output:
```
S1 is: {40,10,20,30}
```

8. Explain the process of adding and modifying an items in dictionary with examples?

A: -Adding an elements in dictionary?

-Adding an item using subscript notation: To add a new entry or a key-value pair in a dictionary, just specify the key value pair without any existing key in dictionary.

Syntax?

```
dict_var[key]=val
```

Example

```
dict1={"name": 'Ajay', 'course': 'MTech'}
Print(dict1)
dict1["year"]="2021"
Print(dict1)
```

output:

```
{"name": 'Ajay', "course": "MTech"}
{"name": "Ajay", "course": "MTech", "year": "2021"}
```

Modifying an items in Dictionary: It updates the values or replace the value with the given data.

Ex?

```
dict_var={"name": "krishna", "age", "27"}
Print(dict_var)
dict_var["age"]=37
Print(dict_var)
```

output?

```
{"name": "krishna", "age": 27}
{"name": "krishna", "age": 37}
```

12) Explain zip() function with suitable example. Write a program to Print the elements and their indices using enumerate() function.

A: zip():

zip() is a built-in function that takes two or more sequences and "zips" them into a list of tuples.

the tuples thus, formed has one element from each sequence

Program to show the use of zip() function:

```
tup=(1,2,3,4,5)
list1 = ['a','b','c','d','e']
print(list(zip(tup, list1)))
```

output:

```
[(1,'a'),(2,'b'),(3,'c'),(4,'d'),(5,'e')]
```

Enumerate():

Enumerate() function adds a counter to an iterable and returns it in a form of enumerate object.

Eg:
```
list = ['dog','cat','rat','donkey']
for x,y in enumerate(list):
    print x
    print y
```

output:

```
0
dog
1
cat
2
rat
3
donkey.
```

8) Explain the following dictionary methods?
   i) copy()
   ii) getckey)
   iii) items()
   iv) update()
   v) has_keyckey)

A: copy() copy() method returns a shallow copy of the dictionary

Ex: >>> d1
{'c':'cat', 'B':'Ball', 'A':'Apple'}
>>> d2 = d1.copy()
>>> d2
{'B':'Ball', 'c':'cat', 'A':'Apple'}

getc() this method is used to access the value by using a key. If the specified key is not available, then it return None.
syntax: d.getckey)

Ex: >>> d1
{'c':'cat', 'B':'Ball', 'A':'Apple'}
>>> d1.get('A')
'Apple'

items() items() method returns a view object that display a list of dictionary's ckey, value) tuple pairs.
(or)
Return a list of key, value pairs from the dictionary.

Syntax:
d.items()

Ex: >>> d1
{'c':'cat', 'B':'Ball', 'A':'Apple'}
>>> d1.items()
dict_items([('c', 'cat'), ('B', 'Ball'), ('A', 'Apple')])

update(): update a dictionary by taking key-values from an existing dictionary.

Syntax:

```
dict.update([other])
```

Ex:

```
d = {1: "one", 2: "three"}
d1 = {2: "two"}
d.update(d1)
Print(d)
```

op:

```
{1: 'one', 2: 'two'}.
```

has_key(): has_key() return true if a given key is available in the dictionary, otherwise it returns a false.

Syntax:

```
dict.has_key(key)
```

Ex:

1, Explain the following built-in- string method?
   i, count()
   ii, find()
   iii, startswith()
   iv, capitalize()

sol count()°
The string count() method returns the number of occurences of substring in given string.

syntax
String.count(substring, start, end)

Ex°>>>S='uma shanker'
>>> s.count('a')
2.

find()° the find() method returns the index of first occurence of the substring. If not found, it returns -1.

Syntax°
Str.find(sub, start, end)

Ex° >>>S='siva rama prasad kollu'
>>> s.find('kollu', 10)
17.

Starts with()° the starts with() method return true if the string starts with specified prefix. If not, it return false

Syntax° Str.startswith(prefix, start, end)

Ex° >>>S_1 = "Hai Bye Bye"
>>> S_2 = "Bye"
>>> If S_1.startswith(S_2):
print("S_1 begin with S_2")
else:
print("S_1 does not begin with S_2")

op° S_1 does not begin with S_2

capitalize()?

The capitalize() method converts first character of a str to uppercase letter and lowercase all other characters. If any.

Syntax

S.capitalize()

Eg:

```
>>> S='ajay ram'
>>> S.capitalize()
o/p 'Ajay ram'
```

2, Explain the string comparison operators with examples?

A: Comparing strings:

python allows to compare strings using relational operators such as >, <, <= etc, some of these operators along with description usage given below table.

operators	Description	Example.
==	If two strings are equal, it returns true.	>>> "ABC"=="Abc" True
!= (or) <>	If two strings are not equal it return true	>>> "abc"<>"ABc" True
>	If the first string is greater than second, it return true	>>> "abc">"Abc" True
<	If the second string is greater than first it return true	>>> "Abc"<"abc" True
>=	If the first string is greater than or equal to the second string, it return true	>>> "aBc">="ABC" True
<=	If the second string is greater than or equal to first string it return as true	>>> "ABC"<="ABc" True.

3) Explain the following built-in string methods with examples
   i) isalnum()
   ii) isalpha()
   iii) islower()
   iv) isdigit()

A: isalnum(): the isalnum() method return True if all characters in the string are alphanumeric. If not, it returns False.

syntax:
```
s.isalnum()
```
Ex:
```
>>> s="surya123"
>>> s.isalnum()
True
```
isalpha(): the isalpha() method returns True if all characters in the string are alphabets, If not, it return false.

syntax:
```
s.isalpha()
```
Ex:
```
>>> s="python"
>>> s.isalpha()
True.
```
islower(): the islower() method return True if all alphabets in a string are lowercase alphabets. If a string atleast one uppercase alphabet, it returns False.

syntax:
```
s.islower()
```
Ex:
```
>>> s="Ajay"
>>> s.islower()
False.
```
isdigit(): the isdigit() method returns True if all characters in a string are digits. If not, it return False.

syntax:
```
s.isdigit()
```
Ex:
```
>>> s="456ak"
>>> s.isdigit()
```

8, Explain the following built-in-string methods with example?
i, lstrip()
ii, rstrip()
iii, strip()

Sol lstrip(): the lstrip() method return a copy of the string with leading characters removed.

Syntax:
String.lstrip([chars])

Ex: >>> S = ' Ajay'
>>> S.lstrip()
'Ajay'

rstrip(): the rstrip() method return a copy of the string with trailing characters removed.

Syntax:
String.rstrip([chars])

Ex: >>> S = 'Ajay '
>>> S.rstrip()
'Ajay'

strip(): the strip() method return a copy of the string by removing both the leading and trailing characters.

Syntax:
String.strip([chars])

Ex: >>> S = ' Ajay '
>>> S.strip()
Ajay

1, Define a function? What are types of function? write a Program to add two number using a function?

A: function?

1. A function is a group or a block of related statements that perform a specific task
2. function is help us to break our program into smaller module.
3. they are two types of function.
   i. In-built functions
   ii, user defined functions.

i. In-built function?

functions that readily come with python are called built-in--functions.

2. the python interpreter has a number of function and types built into it that are allway available.

ii, user-defined functions?

If our requirement is available in built-in-f then we can define our own function called user defined functio

Syntax?

```
def fun_name(parameters):
    statement-1
    statement-2
      :
    statement-n
```

4, keyword def indicates start of function header
5, function_name should be unique and valid identifier
6, parameters through which we pass value to a function
7, colon(:) indicate the end of function header.
8, statements of the function must have the same indential le

9; Calling functions

once we have define a function, we can call it from another function program or even from the python prompt.

3, To call a function we have to type the function name with appropriate parameters.

10, program to subtract two number

2, Explain the local and global variable with an examples?

A: they are two types of variables.

1. global variable
2. Local variable.

Global variable?

A variable declared outside of all the functions. It is available to all the functions is called global variable

Eg?

```
a=10
def func1():
    print('func1:', a)

def func2():
    print('func2:', a)
func1()
func2()
```

output:
func1 : 10
func2 : 10

local variable:
A variable declared inside any function is called local variable. Local variable can be accessed only in that function

Ex:
```
def func1():
    a=10
    print('func1:',a)
def func2():
    Print('func2:',a)
func1()
func2()
```

output:
func1 : 10
NameError : name 'a' is not defined.

3. Explain how a function can return Multiple values with an example?

A: In C, C++, Java, a function can return only single value, But in Python, we can return multiple value.

2. In Multiple arguments are returned, default return type is tuple. If we need a list, we have to convert explicitly.

Ex:
```
def arithmetic_operations(x,y):
    return x+y, x-y, x*y, x/y, x%y
sum,sub,mul,div,mod = arithmetic_operations(20,10)
print('Addition:',sum)
Print('subtractor:',sub)
Print('Multiplication:',mul)
print('Division:',div)
Print('Modulus:',mod)
```

output:
Addition:30

subtractor: 10
Multiplication: 200
Division: 2.0
Modulus: 0

7) Explain the following types of arguments used in functions
1. Required argument or position
2. keyword argument
3. default argument
4. variable-length argument.

A:

Positional arguments

If we define any function with 'n' parameter, then we have to call that function with 'n' arguments only. If we call it with an different arguments, the interpreter will show an error message.

2 When we call a function with some values, the value get assigned to argument according to position.

Ex: def wish(name, msg):
    Print('Hi', name+',', msg)
wish('Siva', 'Good Morning')
wish('ram', 'Good Afternoon')
wish('prasad', 'Good Evening')
wish('Kollu', 'Good Night')

output:
Hi Siva, Good Morning
Hi rama, Good Afternoon
Hi Prasad, Good Evening
Hi Kollu, Good Night

Keyword Arguments

Python allows functions to be called using keyword arguments, when we call function, the order of arguments can be changed.

Here, order is not important but number of arguments must be same.

Ex: def wish(name, msg):
    Print('Hi', name+',', msg)
wish(name='siva', msg='Good Morning')
wish(msg='Good Afternoon', name='rama')

output:
Hi Siva, Good Morning
Hi rama, Good Afternoon

3. Default argument: In python, we can initialize default value for function argument. If we are not passing the argument to msg, then the default value will be taken.

Ex: def wish(name, msg='Good Morning'):
        Print('Hi', name+',', msg)
    wish('siva')
    wish('ram', 'Good Evening')
    wish('prasad')

O/P:
Hi Siva, Good Morning
Hi ram, Good Evening
Hi prasad, Good Morning.

4. Arbitrary arguments: function can take a variable number of argument to be passed to a function is not know beforehand. They are two types of variable length arguments namely variable-length positional arguments and variable-length keyword argument.

Ex: def var_args(*args):
        Print(args)
    var_args("welcome", "to", "python", 30, 29.56)

VLPA:

output:
('welcome', 'to', 'python', 30, 29.56)

variable length keyword argument (**kwargs): It collect the keyword argument in dictionary, where the argument names are keys.

Eg: def var_arg(**kwargs):
    Print(kwargs):
var_args(str1 = "welcome", str2 = "40", str3 = "python")
output
{'str1': 'welcome', 'str2': 40', 'str3': 'python'}

18, Explain the usage of anonymous function lambda with example?

A: 1, An anonymous function is a function that is defined without a name.

2, While normal functions are defined using the def keyword in Python, anonymous functions are defined using lambda keyword

3. Syntax
    lambda arguments: expression.

4. Lambda functions can have any number of arguments but only one expression.

5. the expression is evaluated and returned.

6. Lambda functions can be used wherever function object are require

7. For instant use only.

8. Lambda function internally return expression value and we not required to write return statement explicity.

9. If we want to pass function argument to another function, lambda function are best choice.

10. Lambda function are used along buiet-in-function like filter(), Map(), reduce() etc

Eg: S = lambda n: n*n
Print(S(5))
Print(S(2))
O/P 25
4

8. Write a python program to print fibo series?

```
def fib(n):
    a, b=0,1
    while a<n:
        print(a,end=' ')
        a, b=b, a+b

fib(100)
```

O/P?

0 1 1 2 3 4 8 13 21 34 55 89

9. Explain recursive functions? Write a program to calculate the factorial of a number recursively?

A. Recursive functions?

1. A function calls itself is called recursive function

Advantages?

1. Reduce the length of code
2. Improve readility
3. We can solve complex problem easily

Program?

```
def factorial(n):
    if n==0:
        result=1
    else:
        result=n* factorial(n-1)
    return result
print (factorial(5))
```

O/P?

9, Write a python program to check a number is palindrome or not using in python?

A: Program

```
def reverse(n):
    rev=0
    while n>0:
        r=n%10
        rev=rev*10+r
        n=int(n/10)
    return rev
x=int(input("Enter a number:"))
result=reverse(x)
if result==x:
    Print("Number is Palindrome:",x)
else:
    Print("Number is not Palindrome:",x)
```

O/P

Enter a number:121

Number is Palindrome:121.

9, Explain the following built-in-class attributes?

1. -dict-
2. -doc-
3. -name-
4. -bases-
5. -Module-

A: -dict-: this is a dictionary holding the class namespace

Ex:

```
class Awesome:
    def __init__(self):
        Print("Hello from __init__ method.")
Print(Awesome.__dict__)
```

O/P: {'__Module__': '__main__', '__doc__'}

Syntax:

Class.__dict__

2, __doc__ : This gives us the class documentation if documentation is present. None otherwise.

Syntax:

class.__doc__

Ex: class Awesome:

def __init__(self):

Print("Hello from __init__ method.")

Print(Awesome.__doc__)

o/p: this is a sample class called Awesome.

3, __name__: this gives us the class name.

Syntax: class.__name__

Ex: class Awesome:

def __init__(self):

Print("Hello from __init__ method.")

Print(Awesome.__name__)

output:

Awesome

4. __bases__: A possibly empty tuple containing the base classes in the order of their occurrence.

Syntax:

class.__bases__

Ex: class Awesome:

def __init__(self):

Print("Hello from __init__ method.")

Print(Awesome.__bases__)

output:

(<class 'object'>,)

5, __Module__ : this gives us the name of the module in which the class is defined in an interactive mode it will give us __main__

Syntax:

```
Awesome.__Module__
```

Eg:

```
class Awesome:
    def __Module__(self):
        Print("Hello from __init__ method.")

Print(Awesome.__Module__)
```

o/p:

```
__main__
```

10, Explain the following built-in-functions?

Syntax: 1. hasattr(obj, name)
2. getattr(obj, name, default)
3. setattr(obj, name, value)
4. delattr(obj, name).

1, __getattr__(self, name): Returns the attributes self.name if not found through normal attribute lookup.

2, __setattr__(self, name, value): sets the attribute self.name = value. overrides the default machine.

3, __delattr__(self, name): Delete the attribute self.name.

4, __hasattr__(self, name):

hasattr() is called by getattr() to chek to see if AttributeError is to be raised or not

2, hasattr() Method takes two parameters:

object: object whose named attribut is to be checked

name: name of attribute to be searched, otherwise False.

5, getattribute(self, name) - Returns the attribute self.name.

11, What is a Module? Write a program by using Module?

A 1, Modules refer to a file containing python statements and definitions

2, A python containing python code, Ex: Ajaypy is called a Module, and it's Module name would be Siva.

3, We use Modules to break down largest programs into small manageable and organized files.

4, Modules provide reusability of code

5, Every python file (.py) as a Module.

6, If we want to use members of Module in our program then we should import that Module.

7, We can access members of the Module by using Module name.

Modulename.variable
Modulename.function()

8, Advantages of Module
i, Code Reusability
ii, Readability improved
iii, Maintainability is easy.

9, We can also use alias name for the Modules

10, We can import particular members of Module by using from with import.

11, the main advantage of this is we can access member directly without using Module name.

Ex: from Ajayop import *
add(10,20)
sub(20,10)
O/P:
sum is : 30
sub is : 10

12, What is mean by packages?

A. Package

A package is basically a directory with Python files and file with the name __init__.py

1, package is a group of related entities.

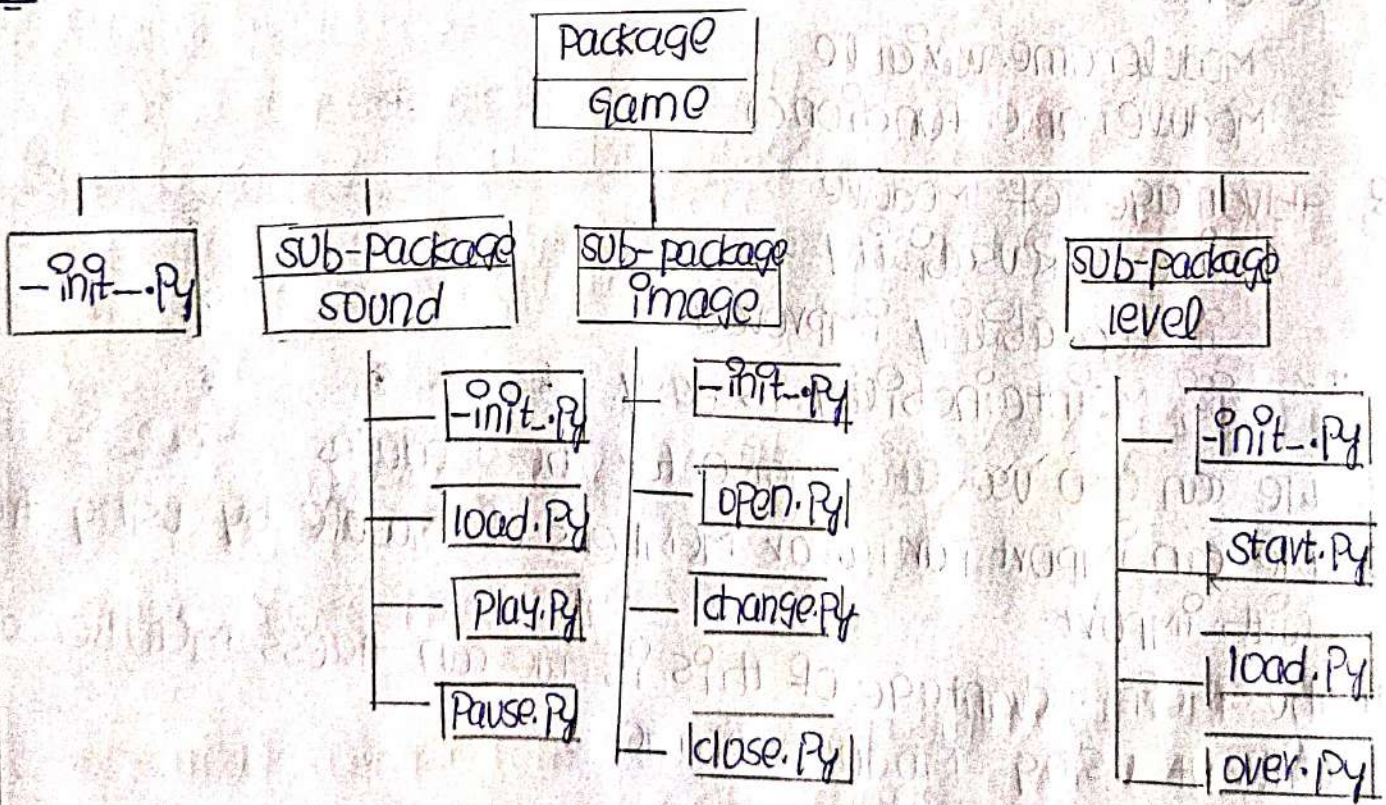
2, As a directory can contain subdirectories and files, a python package can have sub-package and modules

3, It is encapsulation mechanism to group related module in single unit.

4, A directory must contain file named __init__.py in order for python to consider package.

5, the file can be left empty but we generally place initialization code for package in file

Exe



6, Advantages:

i, We can resolve naming conflicts

ii, Modularity is improved

iii, Readability and maintainability is improved

7. package names should be all lower case when multiple words are needed. an underscore should separate them. It is usually preferable to stick to 1 word names.

8. Modules:

i. Module name should be all lowercase
ii. when multiple words are needed an underscore should separate them
iii. It is usually preferable to stick to 1 word names.

13. What is the file? Explain the built-in-functions of file?

1. files are named locations on disk to store related information
2. files are used to store data permanently in a non-volatile memory
Ex: Hard disk.
3. Random Access Memory is volatile we can use file for future use of data by permanently storing them.
4. there are two types of files that can be handle in python
   1. Text files
   2. Binary files.

Text files: In this type of file, each line of text is terminated with a special character called EOL, which is new line character ('\n') in python by default.

Binary files: In this type of file, there is no terminator for line and data is stored in form of machine understandable binary language

5. In python, file operation take place in following order
   1. open a file
   2. Read or write
   3. close the file.

Various properties of file object:

Once we opened a file and we got file object, we can get various detail related to file by using properties.

name: Name of opened file
Mode: Mode in which the file is opened
closed: Return boolean value indicate that file is closed or not
readable(): Return boolean value indicate that whether file readable or not
writable(): Return boolean value indicate that whether file is writable or not.

writing to files:

write(s) : write the string s to file and returns the number of characters written
writelines(lines): writes a list of lines to file

Reading from files:

read(): To read total data from the file
read(n): To read 'n' characters from the file
readline(): To read only one line
readlines(): To read all lines from the file into a list

seek():

We can change our current file cursor (position) using seek() Method

Syntax: seek(offset, from)

tell():

tell() is used to tell the current position of file pointer from file character. Index of starting character is 0

1) What is meant by pickling and unpickling of object?

Ans Sometimes we have to write total state of object to the file and we have to read total object from the file.

1) the process of writing state of object to the file is called "pickling" and the process of reading state of an object from the file is called "unpickling".

2 we can implement pickling and unpickling by using Pickle Module in Python.

3. the Pickle Module contains dump() function to perform pickling

```
Pickle.dump(object, file)
```

4 pickle Modue contain load() function to perform unpickling

```
obj = Pickle.load(file)
```

Pick.Py:

```
import emp, Pickle
f = open("emp.dat", "wb")
n = int(input("Enter the number of EMPloyees:"))
for i in range(n):
    eno = int(input("Enter Employee Number:"))
    ename = input("Enter Employee Name:")
    esal = float(input("Enter Employee salary:"))
    eaddr = input("Enter Employee address:")
    e = emp.Employee(eno, ename, esal, addr)
    Pickle.dump(e,f)
print("Employee object pickled successfully")
```

unpick.py:

```
import emp, pickle
f = open ("emp.dat", "rb")
Print ("Employee details:")
While True:
        try:
                obj = pickle.load(f)
                obj. display()
        except EOF Error:
                Print ("All employee completed")
                break
f.close().
```

2, Explain about zipping and unzipping files?

A:

zipping

zipped files take up less storage space and can be transfered to other computers more quickly than other files.

2, A zip file is a single file contain one or more compressed files way. To make large to smaller files keep related together

unzipping

A unzip file is a multiple file contain one or more decompress file way. To make smaller to larger files keep related to separated.

1. It is very common requirement to zip and unzip format.

-Advantages?

1. To improve memory utilization
2. We can reduce Transport time
3. We can improve performance

3, Multiple decorators can chained in python
4, A fn can be decorated multiple times with different decorator

Ex°
```
def square(fun):
    def inner(x):
        y = x ** 2
        fun(y)
    return inner
@square
def num(x):
    Print(x)
num(10)
```
o/p°
100

9, Explain about the generator?

A 1, A generator is a function that return a object. which we can iterate over

2, Generator function is same as normal fn but it uses "yield" keyword.

3, We can iterate the items of a generator using next() function.

4, Methods like __iter__() and __next__() are implemented automatically.

5, the generator object can be iterated only once.

6, To restart the process we need to create another generator object like a = my_gen().

7, Normally, generally fn are implement with a loop having suitable terminating condition.

8, Multiple generator can be used to pipeline series operation

9. Uses°

i, Easy to Implement

ii, Memory utilization in high

```
def myGenerator():
    Print ('FirstItem')
    Yield 10
    Print ('Second item')
    Yield 20
```

op: >> gen = myGenerator()
>> next(gen)
first item
10

3 Explain about the Exception?

1, An Exception is an event, which occurs during the Execution of Program that disrupts the normal flow of program's instructions
2-An exception is a python object that represents an error
3, An unwanted and unexpected event that disturb normal flow of program is called exception.
4, Ex: zeroDivisionError
TypeError, IndexError
ValueError etc.,
5, It is highly recommended to handle exception
6, Every exception in Python is an object
7, for Every Exception type corresponding classes are available.
8, whenever an exception occurs PVM will check the corresponding exception object will check for handle code
9 If handle code is not available then python interpreter terminates the program abnormally and print corresponding except information console.
10 the rest of program won't executed
11 they are two types of Exception
i, predefined Exception
ii, user defined Exception
12, predefined Exception: the exception are by default available in Python
13 user defined Exception: we can define our own Exception in python user responsible to define exception

14, the code which may raise Exception have to take inside a try block.

15, the corresponding handling code we have to take inside except block.

Syntax:

```
try:
    code which may raise exceptions
except ExceptionType:
        Exception Handling code
```

Program:

```
try:
    a=int(input("enter any number of a:"))
    b=int(input("enter any number of b:"))
    c=a/b
    Print("Division of two numbers:"c)
except zeroDivisionError:
    Print("zeroDivision Error division by zero is not possible:")
```

2, What is raise keyword?

A. raise keyword: It is used to raise an exception.

Program:

```
try:
    n1=int(input("enter any value of n1:"))
    n2= int(input("enter any value of n2:"))
    if(n2==0):
        raise zeroDivisionError
except:
    Print("zeroDivisionError is not possible:")
```

6, Write the Different between kernel-level thread and user-level thread

Ans There are TWO types of thread

1, kernel-level thread
2, user-level thread

kernel-level Thread	User level Thread.
1. Recognized by the operating system	1. Does not recognized by the operating system.
2. Implemented by the operating system.	2. Implemented by a user of the system
3. Implementation is complex	3. Implementation is simple and easy.
4. Solaris is an example	4. Posix is an example
5. Require Hardware support	5. Requires no Hardware support

Application of Multi-threading.

1. To implement multimedia graphics
2. To develop animations
3. To develop video games
4. To develop web and application serves
5. It can concurrently run on multiple cpus

9. What is Meant by Synchronization?

A 1. If Multiple threads are executing simultaneously then there may be chance of data inconsistency problems.

2. In synchronization the threads will be executed one by one so that we can overcome data inconsistency problems.

3. Synchronization means at a time only one thread.

4. Application?

1. Online Reservation system
2. funds transfer from joint account

5. In Python, we can implement synchronization by using the following.

1. Lock
2. Rlock
3. Semaphore

6. Lock are the most fundamental synchronization Mechanism Provide threading Module

7. we can create Lock object

```
l=Lock()
```

8. A thread can acquire lock by using acquire() method

```
l.acquire()
```

9. A thread can release lock by using release() Method

```
l.release()
```

10. To call release() Method compulsory thread should has the lock already, otherwise we will Runtime Exception saying

11. Rlock object can be acquired by one thread at time, but owner thread can acquire some lock object Multiple times

12. In this case Rlock object will take care where locked or Unlocked and owner thread information, recursive level

```
l=Rlock()
```

9, In the case of Lock and Rlock, at time only one thread is allowed to execute.

10, Sometime our requirement is at a time particular no of thread are access.

11, To handle requirement we cannot Lock and Rlock concept go to semaphore concept.

14, We can create semaphore object

S = Semaphore (counter)

Ex: S = Semaphore()

In case counter value is 1 and at a time only one thread allowed to access. It is exactly same Lock concept

10, Explain about Inter-thread communication?

1, Some times as the part of programming requirement, thread required to communicate to each other.

2, this concept is nothing but interthread communication.

3, In Python, we can implement interthread communication the following ways.

1. Event
2. condition
3. queue

4. Event object is simplest communication mechanism between threads. one thread signal an event and other thread wait

event = threading.Event()

5. Event manage an internal flag set() or clear()

6. condition is always associated with a Lock

7. A condition has acquire() and release() methods that call corresponding methods

8. We can create condition object as follows

Condition = threading.condition()

9. Queue internally has condition and condition has lock. Hence whenever we are using queue we are not required to worry about Synchronization

10. If we want to use queue first we should import queue Module

import queue

11. We can create queue object as follows

q = queue.Queue()

12. They are two types

1. FIFO Queue: This is default Behaviour. In which order we put item in the queue the same order item come at

q = queue.Queue()

13. LIFO Queue: The removal will happend in reverse order of insertion.

q = queue.LifoQueue().

11. Write the built-in function of threading module?

A 1. activeCount(): Returns the count of thread object which are still live

factory fn 2. current thread(): Returns the current object of thread class

3. enumerate(): List all active thread objects

4. isDaemon(): Returns True if the thread is daemon

5. is-Alive(): Returns true if thread is still live

6. enumerate(): fn return a list of all active thread currnig running

10, What is mean by GIL?

A 1. A Global Interpreter Lock (GIL) is a Mechanism to apply a global lock on an interpreter

2. It is used in a computer language interpreters to synchronize and manage the execution of threads. so, only one native thread can execute at a time

3. In a scenario, where you have multiple threads, what can happen is that both the thread might try to acquire the memory at the time

4. And the result of which they overwritten the data in memory.

5. Hence, arises a need to have a Mechanism that could help Prevent phenomenon.

6. Some popular interpreter that have GIL are cpython and Ruby MRI

7. Even if your process has multiple cores, a Global Interpreter will allow only one thread to executed at time.

8. Because, when a thread start running. It acquire the GIL.

9. When it waits for any I/O operation or cpu bound operation

10. It's release the lock so that other threads of that process can run.

11. Hence, it prevents you from running other threads at same time.