Question
Explain Applications of Python?

A) Applications of Python:-

Python is used in many application domains.

(1) Web and Internet Development:-

Python offers many choices for web development:

→ Frameworks such as Django, & Pyramid

→ micro-frameworks such as Flask and Bottle

Python's standard library supports many Internet protocols:

→ HTML & XML

→ JSON

→ Email processing

→ supports FTP, IMAP

→ Easy-to-use socket interface

(2) Scientific and Numeric:-

Python is widely used in Scientific and Numeric computing:

→ SciPy → Collection of packages for mathematics, Science and engineering

→ Pandas → Data analysis and modelling Library

→ IPython → supports visualization and parallel computing

(3) Education:-

Python is a superb language for teaching programming both at introductory level and in more advanced courses.

(4) Desktop GUIs :-

The Tk library is included in with most binary distributions of Python

(5) Software Development

Python is given as a support language for software developers, for build control and management, testing and in many other ways.

SCons → for build control

Buildbot & Apache Gump → automated continous compilation and testing

(6) Business Applications

Python is also used to build ERP and e-commerce systems

Ex: Odoo

Tryton

Question

→ List python features?

Ans (1) simple

(2) Easy to learn

(3) open source

(4) High level Language

(5) Dynamically Typed

(6) Platform Independent

(7) portable

(8) procedure and object oriented

(9) Interpreted

(10) Extensible

(11) Embeddable

(12) Huge Library

(13) Scripting Language

(14) Database Connectivity

(15) Scalable

✓

Question

→ State the usage of Comments

Ans; There are 2 types of Comments in Python: Single Line Comments and multi line Comments

→ Single line Comments:

These ~~line~~ Comments starts with a hash Symbol (#) and are useful to mention that the entire line till the end should be treated as Comment. For Example,

```
# To find sum of two numbers
a=10  #store 10 in variable a
```

Here the first line starting with a # Symbol is and hence the entire line is treated as a Comment. In the second line, a=10 is a Statement and after that Statement, # Symbol starts the Comment describing that value 10 is stored in variable 'a'.

→ Comments are non executable Statements.

It means neither the Python Compiler nor the PVM will execute them.

multiline comments

When we want to make several lines as comment, then writing # Symbol in the beginning of every line will be a tedious job.

For ex:

```
#This is a program
# to find factorial of a
# number using Recursion
```

Instead of starting every line with # Symbol, we can write previous block of code inside """ (triple double quotes) (or) ''' (triple single quotes) in the beginning and ending of block as:

```
""" this is a program
to find factorial of a
number using Recursion """
```

(or)

```
''' this is a program to
find factorial of a
number using Recursion '''
```

Question

⇒ Explain declaration and initialization of a variable?

Ans: A variable is a name given to a memory location to store data in the memory. we can think of a variable as a container that holds data that can be changed later in the program.

For Ex:

```
a=10
name='KVK'
city='Tirupati'
print(a)      # 10
print(name)   # KVK
print(city)   # Tirupati
```

Here a, name & city are names of the variables and 10, KVK & Tirupati are variable values

Creating (or) declaring variables

Python has no command for declaring a variable. A variable is created the moment when we assign a value to it.

In Python a variable is created only when we assign a value to it.

Example:-

```
a = 20.5  # declaration and initialization of variable
x = 'Amaravati'
print(a)
print(x)
print(type(a))  # class <float>
print(type(x))  # class <str>
```

---

① Define function?

A function is similar to a program that consists of a group of statements that are intended to perform a specific task. The main purpose of a function is to perform a specific task or work.

→ List Types of functions?

In Python, functions are classified into 2 types:-

(*) Built-in functions

(*) users defined functions

(*) Built-in functions: Python has several built in functions ~~foo~~ to perform various tasks. For Example

print() → to display output

sqrt() → to calculate square root value

powers() → to calculate powers value

(*) users defined functions: Functions are created by users or programmers to solve a specific problem. To create a function, we use def keyword

Ex: def sum(a, b)

c = a+b

return c

Here sum() is the name of the function, a & b are formal parameter

Question

(*) Explain Built in functions?

Ans:

The python interpreter has a number of functions and types built into it. that are always available. Some of the built in functions are listed below

(1) abs() : Returns the absolute value of a number. The argument may be integer, float or (or) an object implementing —abs— () function

(2) all() : Returns true if all the elements of the iterable are true (or if the iterable is empty).

(3) any() : Returns true if any element of the iterable is true.

(4) bin() : Converts an integer number to a binary string prefixed with "0b"

(5) oct() : Converts an integer number to an octal string prefixed with "0o"

(6) hex() : Converts an integer to lower case hexadecimal string prefixed with "0x".

(7) id(): Returns the identity of an object

(8) pow(base, exp, mod=None) Returns base to the power exp

(9) int(): Returns an integer object constructed from a number or string

(10) float(): Returns a floating number constructed from a number or string

(11) bool(): Returns a boolean value i.e True or False

(12) complex(): Returns a complex number with the value real + imag*1j

(13) setc():

(14) locals(): update and returns a dictionary representing local symbol table

(15) globals(): returns the dictionary implementing the current module namespace

(16) hash(object): returns the hash value of the object

(17) help(): invokes the built-in help system

(18) input()
input(prompt) } : used for reading data form inputs device

(19) print(): to display message onto standard output device

(20) range(): used to generate sequence

(21) map(): returns an iterator that applies function to every item of iterable, yielding the results.

(22) filter(function, iterable): returns an iterator that constructs an iterable for which function is true

(23)

(1) Explain Identifiers, Keywords, Indentation, variables

Ans:

(1) An Identifier is a name, that is given to a variable or function or class etc-

(2) Identifiers can include letters, numbers and the underscore characters (_). They should always start with a non numeric character. Special Symbols such as ?, #, $, % and @ are not allowed in identifiers.

Ex: Salary, name11, gross_income etc-

Salary = 15000.85

This is a variable
Its name is 'Salary'.
Hence this name 'Salary'
is identifies

Floating point literal

(3) Keywords / Reserve words: Keywords are the words that are already reserved for some specific purpose in Python programming language.

④ The names of these reserved words should not be used as identifiers. The following are the available reserved words in python:

and	continue	lambda	def	import
as	class	True	global	is
not	for	false	if	with
in	while	try	else	del
break	pass	except	from	elif ...

⑤ Python Indentations

Indentation refers to spaces that are used in the beginning of a statement. The statements with same indentation belong to same group called a suite

By default python uses 4 spaces but it can be increased or decreased by the programmer.

Eg:

```
if x==1:
    print('a')
    print('b')
    if y==2:
        print('c')
        print('d')
print('end')
```

In the above example,

```
if x==1:
print('end')
```

} belongs to same group as they dont have spaces before them.

In the next level, the following statements are typed with 4 spaces before them and hence they are at the same level (same suite).

```
print('a')
print('b')
if y==2:
```

} Same group (or) suite

In the next level, two statements are typed with 8 spaces before them hence they belong to same group

```
print('c')
print('d')
```

} Same group (or) suite

Benefits of Indentation:

→ It is used to properly structure the code.

→ Python indentation rules are simple. Most of the python IDE's automatically indent the code for you.

Disadvantages of Indentation:

→ Since whitespaces are used for indentation, if the code is large and indentation is corrupted, then it is really difficult to correct the code

→ most of the programming languages uses braces for indentation, so programmers coming from, like Java background finds it hard at first to adjust the idea of using whitespaces for the indentation.

⇒ Explain Input and output Statements

Input and Output

TO provide input to a computer, Python provides some Statements which are called Input Statements Similarly to display the output, these are output Statements available in Python:

Output Statements

TO display output as results, Python provides the print() function. this function can be used in different formats

1. The print() Statement:

when print() function is called simply, It will throw the cursor to the next line. It means blank line will be displayed

2. The print("String") Statement

when a String is passed to the print() function, the String is displayed as it is.

Ex: print("HELLO")

O/P: HELLO

3. The print (variable list) Statement :-

we can also display the values of variables using the print() function.

Ex:
```
a, b = 2, 4
print(a)
2
print(a, b)
2 4 # values are separated by a space by default
```

to separate the output with comma, we should use 'sep' attribute as shown below.

```
print(a, b, sep=',')
```

o/p: 2,4

```
print(a, b, sep=':')
```

o/p: 2:4

```
print(a, b, sep='----')
```

o/p: 2----4

4. The print (object) Statement

we can pass objects like lists, tuples, or dictionaries to the print() function to

display the elements of those objects.

For Example:

```
lst = [10, 'A', 'Hai']
print(lst)
```

o/p: [10, 'A', 'Hai']

⑤ The print("String", variablelist)

The most common use of the print() function is to use Strings along with variables inside the print() function.

```
a=2
print(a, " is an even number")
```

o/p: 2 is even number

⑥ The print(formatted String) Statement

Explain formatted output Statement

The output displayed by print() function can be formatted as we like. The Special operator % (percent) can be used for this purpose.

print ('formatted string' %. (variable list))

- In the formatted string we can use

%i (or) %d → integer number

%f → float numbers

%s → strings

Ex: x = 10
print ('value = %i' %. x)

O/p: value = 10.

Ex: x, y = 10, 20
print ('x = %i y = %d' %. (x, y))

Ex: num = 123.456789
print ('The value is: %f' %. num)

O/p: The value is 123456789

print ('The value is: %8.2f' %. num)

O/p: The value is 123.45

Inside the formatted String, we can use replacement field which is denoted by a pair of curly braces {}. we can mention names or indexes in these replacement fields. Those names or indexes represent the ~~order. The~~ order of the values.

After the formatted String, we should write member operator(.) and then format() method where we should mention the values to be displayed. Consider the general format given below:

print('Formatted String with replacement fields'.format(values))

Ex: n1, n2, n3 = 10, 20, 30

print('number1={0}, number2={1}, number3={2}'.format(n1, n2, n3))

(or)

print('number1={}, number2={}, number3={}'.format(n1, n2, n3))

(or)

print(f'number1={n1}, number2={n2}, number3={n3}')

Input Statement

To accept input from keyboard, Python provides the input() function. this function takes a value from the keyboard and returns it as a String.

For Example:

```
Sto= input( ) #this will wait till we enters a String
Raj Kumar #enters the String

print (Sto)
```

o/p:- Raj kumar

we can write a message inside the input() function as:

```
Sto= input ('Enter your name')

Enter your name

print (sto)
```

o/p: Raj Kumar

→ we can use the int() function before the input() function to accept an integer from the keyboard as:

```
x=int(input('Enter a number'))
Enter a number 10
print(x)
```

o/p: 10

To accept a float value from keyboard

```
x=float(input('Enter a number'))
Enter a number 123.45
print(x)
```

o/p: 123.45

⇒ input() can be used to accept one or more values

```
#accept full name
name= input('Enter full name')
#separate first name, middle name and sur name
fname,mname,sname= input('Enter full name:').split()
```

split() function will split the entire fullname with space as a delimeters. The split values will then be assigned to fname, mname & lname.

→ If we are to receive multiple init values, we can receive them as Strings and then convert them into ints.

```
n1,n2,n3= input('Enter three values:').split()

n1,n2,n3= int(n1), int(n2), int(n3)
```

(or)

in a more compact manner using a feature called List comprehension

```
n1,n2,n3 =[int(n) for n in input('Enter three values').split()
```

⇒ input() can be used to receive arbitary number of values

```
numbers=[int(x) for x in input('Enter values').split()]

for n in numbers:
    print(n+10)
```

⇒ input() can be used to receive different data types of values at a time:

```
data = input('Enter name, age and salary:').split()

name = data[0]
age = int(data[1])
salary = float(data[2])
```

✓

(*)