Assignment - 02

20/06/2020

RNO: 321803012026
'A' sec, 2nd yers
'CSE'

Python

① What are the data types in python ? Explain

→ There are 5 Datatypes in python Programming

i) Numbers
ii) Lists
iii) Strings
iv) Tuple
v) Dictionary

i Numeric :- A numeric value is any representation of data which has a numeric value. Python identifies three types of numbers:

> integer : Positive or negative whole number

> Float : Any real number with a floating point representation in which a fractional component is denoted by a decimal symbol or scientific notation.

> Complex number : A number with a real and imaginary component represented as

$X + Yj$

> Boolean : Data with one of two built-in values True or False

ii) Sequence Type :

A sequence is an ordered collection of similar or different data types. Python has the following built-in sequence data type

> String: A string value is an order collection of one or more characters put in single, double or triple quotes.

> List: A list object is an orderd collection of one or more data items, not necessary of the same type, put in square brackets.

> Tuple A Tuple object is an orderd collection of one or more data items, not necessarily of the same type, put in paranthesses.

Dictionary: A dictionary object is an unordered collection of data in a key: value pair form. A collection of such pairs is enclosed in curly brackets.

For example:- { 1: "steve", 2: "Bill" }

(2) Briefly explain history of python.

* Python is a widely used general-purpose, high-level programming language. it was initially designed by Guido van Rossum in 1991 and developed by python software foundation.

* Guido van Rossum began doing its application based work in December of 1989 by at Centrum Wiskund & informatica (CWI) which is situated in Netherland.

* The Programming language which Pytho is said to have succeeded is ABC programming Language.

③ Explain all the operators in python?

① Arithmetic operators :-

Arithmetic operators are used to perform mathematical operations like additions, subtraction, multiplication, etc.

Operator

+ , - , * , / , % , // , **

Example.

```
x = 15
y = 4
Print(x+y)
Print(x-y)
Print(x*y)
Print(x/y)
Print(x%y)
Print(x//y)
Print(x**y)
```

② Comparison operator :-

Comparison operator are used to compare values. It returns either True or False

Operators

> , < , == , != , >= , <=

Example:

```
x = 10
y = 20
Print('x > y is;', x > y)
# o/p : False
```

```
Print('x<y is', x<y)
# o/p: True
Print ('x==y is', x==y)
# o/p: False
Print ('x!=y is', x!=y)
# o/p: True.
# o/p Print ('x>=y is', x>=y)
# o/p: False
Print ('x<=y is', x<=y)
# o/p: True
```

3) Logical operators :-

Logical operators are and, or, not operators

and: True if both the operands are true

or : True if either of the operand is true

not: True if operand is false.

Example:-

```
x = True
y = False
Print('x and y is', x and y)
Print('x or y is', x or y)
Print ('not x is', not x)
```

④ Bitwise operators.

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name

Operator

→ Bitwise AND (&)

eg:- x & y = 0

0000 0000

→ Bitwise OR (|)

eg:- x | y = 14

0000 1110

→ Bitwise NOT (~)

eg:- ~x = -11

1111 0101

→ Bitwise XOR (^)

x ^ y = 14

0000 1110

→ Bitwise ~~XOR~~ right shift (>>)

eg:- x >> 2 = 2

0000 0010

→ Bitwise left shift (<<)

eg:- x << 2 = 40

0010 1000

⑤ Assignment operator :-

Assignment operator are used in Python to assign values to variables

eg:- a = 5 → assigns the value 5 on the right to the variable a on the left.

⑥ Special operators

i) identity operator ii) membership operators.

i identity operator.

is and is not are the identity operators in Python

is True if the operands are identical

Eg:- x is True

is not True if the operands are not identical

eg:- x is not True.

Example :-

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
Print(x1 is not y1) # False
Print('x2 is y2') # True.
```

ii) membership operators:

in and not in are the membership operators in Python.

in True if value/variable is found in the sequence.

notin True if value/variable is not found in the sequence.

Example:

```
x = 'Hello world'
x = {1:'a', 2:'b'}

Print ('H' in x) # True
Print ('hello' not in x) # True
Print ('a' in y) # False
```

(4) Explain the features of Python.

> Easy to code

Python is high level programing language. Python is very easy to learn language as compared to other language like C, C#, java etc. Any body can learn python basic in few hours or days. It is also developer friendly language.

> Free and open Source.

Python language is freely available at official website, it is open-source, this means that source code is also available to public.

> object-oriented Language:

One of the key feature of python is object-oriented programming. Python supports object oriented language & concepts of class, objects encapsulation etc.

> High-Level language:

Python is a high level-language. When we write program in python, we do not need to remember the system architecture, nor do we need to manage the memory.

> Python is portable Language:

Python language is also portable language. for eg:- if we have Python code for windows & if we want to run this code on other platform such as Linux, Unix & Mac, then we do not need to change it, we can run this code on any platform.

> Python is Integrated language

Python is also an Integrate language because we can easily integrated python with other language like C, C++ etc.

> Interpreted Language:

Python is an Interpreated Language. because Python code is executed line by line at a time. The source code of python is converted into a immedite form called byte code. there is easier to debug our code.

> Dynamically typed language.

Python is Dynamically-typed language. That means the type for a variable is decided at run time not in advance. because of this feature we don't need to specify the type of variable.

(5) ~~Write~~ Justify why python is interactive interpreted language.

→ Unlike C, C++ etc, python is an interpreted object-oriented programming language. each time a program is run the interpreter checks through the code for errors & then interprets the instructions into machine readable bytecode.

→ If any error is encountered it stops the translation until the error is fixed.