Lab
Principles of Programming Languages

eNotes: CSE/IT/MCA/BCA/MSc

BTCS303 - Lab
Principles of Programming Languages

Exp-3

Understand OOP: Python

About Python

- **Python** is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- It was created by Guido van Rossum during 1985- 1990.
- Like Perl, Python source code is also available under the GNU General Public License (GPL).
- It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes.
- It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming.

Why Learn Python..

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

eNotes: CSE/IT/MCA/BCA/MSc

Characteristics of Python

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

What are applications of Python

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

General of Python

Display WelCome

```
#This is a comment
print("WelCome")
```

Multi Line Comments

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Variables

```
x = 5
y = "Rama"
print(x)
print(y)
```

Casting

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

```
x = 5
y = "Rama"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

Single or Double Quotes

```
x = "Rama"
# is the same as
x = 'Rama'
```

Case-Sensitive

Variable names are case-sensitive.

```
a = 4
A = "Rama"
print(a) #4
print(A) #Rama
```

Many Values to Multiple Variables

```
x, y, z = "Orange", "Banana", "Cherry"

print(x) #Orange
print(y) #Banana
print(z) #Cherry
```

One Value to Multiple Variables

```
x = y = z = "Orange"
print(x) #Orange
print(y) #Orange
print(z) #Orange
```

Unpack a Collection

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x) #apple
print(y) #banana
print(z) #cherry
```

Output Variables

```
x = "awesome"
print("Python is " + x)

x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

```
x = 5
y = 10
print(x + y)

x = 5
y = "Rama"
print(x + y)
```

Python Data Types

[eNotes: CSE/IT/MCA/BCA/MSc]

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Example of Python Data Type

eNotes: CSE/IT/MCA/BCA/MSc

• x = "Hello World"	str
• x = 20	int
• x = 20.5	float
• x = 1j	complex
• x = ["apple", "banana", "cherry"]	list
• x = ("apple", "banana", "cherry")	tuple
• x = range(6)	range
• x = {"name" : "John", "age" : 36}	dict
• x = {"apple", "banana", "cherry"}	set
• x = frozenset({"apple", "banana", "cherry"})	frozenset
• x = True	bool
• x = b"Hello"	bytes
• x = bytearray(5)	bytearray
• x = memoryview(bytes(5))	memoryview

Python If ... Else

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b“)
```

Python Loop

eNotes: CSE/IT/MCA/BCA/MSc

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
for x in range(6):
  print(x) #1 2 3 4 5 6
for x in range(2, 6):
  print(x)
for x in range(2, 7, 2):
  print(x) #2 4 6
```

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Python Functions

- ```
  def my_function():
    print("Hello from a function")
  ```

  my_function() #Hello from a function

- ```
  def my_function(fname):
    print(" Welcome " + fname)
  ```

  my_function("**Rama**") # Welcome Rama