GOVERNMENT POLYTECHNIC AMRAVATI
DIPLOMA PROGRAMME IN COMPUTER ENGINEERING

COURSE : PROGRAMMING WITH PYTHON

COURSE CODE :CM5461

Unit 1
Introduction

RATIONALE

- Python is used for developing desktop GUI applications, websites and web applications. Also, as a high level programming language it allows you to focus on core functionality of the application by taking care of common programming tasks. This course is designed to help the students to understand fundamental syntactic information about ‘Python’. Also it will help the students to apply the basic concepts, program structure and principles of ‘Python’ programming paradigm to build given application. The course is basically designed to create a base to develop foundation skills of programming language.

COURSE OUTCOMES (COs)

At the end of this course, student will be able to: -

- Write and execute simple ‘Python’ programs.
- Write ‘Python’ programs using arithmetic expressions and control structure.
- Develop ‘Python’ programs using List, Tuples and Dictionary.
- Develop/Use functions in Python programs for modular programming approach.
- Develop ‘Python’ programs using File Input/output operations.
- Write ‘Python’ code using Classes and Objects.

UNIT 1 :- CONTENTS

Why Learn Python?

- Python is easy to learn.Its syntax is easy and code is very readable.
- Python has a lot of applications. It's used for developing web applications, data science, rapid application development, and so on.
- Python allows you to write programs in fewer lines of code than most of the programming languages.
- The popularity of Python is growing rapidly. Now it's one of the most popular programming languages.

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.
- It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

History of Python

The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions.

Python Features

- Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as string, int, etc.

Features in Python

There are many features in Python, some of which are discussed below –

1. Easy to code:

   Python is a high-level programming language.Python is very easy to learn the language as compared to other languages like C, C#, Java script, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

Python Features

2. Free and Open Source:

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword.

Download Python Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

Python Features

3.Object Oriented Language

One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

4.GUI Programming Support:

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.PyQt5 is the most popular option for creating graphical apps with Python.

Python Features

5. High-Level Language:

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

6. Extensible feature:

Python is a **Extensible** language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language.

Python Features

7. Python is Portable language:

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language:

Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

Python Features

9. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. There is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called **byte code**.

Python Features

10. Large Standard Library

Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

11.Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) 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.

.

- Most of the programming languages like C, C++, and Java use braces { } to define a block of code. Python, however, uses indentation.
- A code block (body of a function, loop, etc.) starts with indentation and ends with the first unintended line. The amount of indentation is up to you, but it must be consistent throughout that block.
- Generally, four whitespaces are used for indentation and are preferred over tabs. Here is an example.

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

- The enforcement of indentation in Python makes the code look neat and clean. This results in Python programs that look similar and consistent.
- Indentation can be ignored in line continuation, but it's always a good idea to indent. It makes the code more readable. For example:

```
if True:
    print('Hello') a = 5
```

and

```
if True: print('Hello'); a = 5
```

- both are valid and do the same thing, but the former style is clearer.
- Incorrect indentation will result in IndentationError.

Indentation

- Leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements.
- First, tabs are replaced (from left to right) by one to eight spaces such that the total number of characters up to and including the replacement is a multiple of eight (this is intended to be the same rule as used by Unix). The total number of spaces preceding the first non-blank character then determines the line's indentation. Indentation cannot be split over multiple physical lines using backslashes; the whitespace up to the first backslash determines the indentation.

Identifiers

- Identifiers (also referred to as names) are described by the following lexical definitions:

```
identifier  :   (letter|"_") (letter | digit |"_")*
letter      :     lowercase | uppercase
lowercase :   "a"..."z"
uppercase :   "A"..."Z"
digit       :      "0"..."9"
```

Identifiers are unlimited in length. Case is significant.

Keywords

The following identifiers are used as reserved words, or *keywords* of the language, and cannot be used as ordinary identifiers. They must be spelled exactly as written here:

And else global not try class except elif from

del or while continue exec import pass

for def finally in print is Raise assert

lambda return break if

Keywords are the reserved words in Python.

We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language. In Python, keywords are case sensitive.

There are 33 keywords in Python 3.7. This number can vary slightly over the course of time. All the keywords except True, False and None are in lowercase and they must be written as they are. The list of all the keywords is given below.

Comments

A comment starts with a hash character (#) that is not part of a string literal, and ends at the end of the physical line. A comment signifies the end of the logical line unless the implicit line joining rules are invoked. Comments are ignored by the syntax; they are not tokens.

```
#This is a comment
#print out Hello
print('Hello')
```

Comments

A comment starts with a hash character (#) that is not part of a string literal, and ends at the end of the physical line. A comment signifies the end of the logical line unless the implicit line joining rules are invoked. Comments are ignored by the syntax; they are not tokens.

```
#This is a comment
#print out Hello
print('Hello')
```

Python Variables

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data that can be changed later in the program. For example,

```
number = 10
```

Constants

A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.

You can think of constants as a bag to store some books which cannot be replaced once placed inside the bag.

Literals

Literal is a raw data given in a variable or constant. In Python, there are various types of literals they are as follows:

Numeric Literals

Numeric Literals are immutable (unchangeable). Numeric literals can belong to 3 different numerical types: Integer, Float, and Complex.

String literals

A string literal is a sequence of characters surrounded by quotes. We can use both single, double, or triple quotes for a string. And, a character literal is a single character surrounded by single or double quotes.

Boolean literals

A Boolean literal can have any of the two values: True or False.

```
x = (1 == True)

y = (1 == False)

a = True + 4

b = False + 10

print("x is", x)

print("y is", y)

print("a:", a)

print("b:", b)
```

Boolean literals

x is True

y is False

a: 5

b: 10

Special literals
Python contains one special literal i.e. None. We use it to specify that the field has not been created.

Special literals

```
drink = "Available"
food = None
def menu(x):
    if x == drink:
        print(drink)
    else:
        print(food)
menu(drink)
menu(food)
```

Input and Output

Python provides numerous built-in functions that are readily available to us at the Python prompt.

Some of the functions like input() and print() are widely used for standard input and output operations respectively. Let us see the output section first.

Python Output Using print() function

We use the print() function to output data to the standard output device (screen).

print('This sentence is output to the screen')

a = 5 print('The value of a is', a)

Input and Output

In the second print() statement, we can notice that space was added between the string and the value of variable a. This is by default, but we can change it.

The actual syntax of the print() function is:

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

Here, objects is the value(s) to be printed.

The sep separator is used between the values. It defaults into a space character. After all values are printed, end is printed. It defaults into a new line. The file is the object where the values are printed and its default value is sys.stdout (screen).

Input and Output

```
print(1, 2, 3, 4)

print(1, 2, 3, 4, sep='*')

print(1, 2, 3, 4, sep='#', end='&')
```

Output :

```
1 2 3 4

1*2*3*4

1#2#3#4&
```

Output formatting

Sometimes we would like to format our output to make it look attractive. This can be done by using the str.format() method. This method is visible to any string object.

```
>>> x = 5; y = 10
>>> print('The value of x is {} and y is {}'.format(x,y))
 The value of x is 5 and y is 10
```

Input and Output

Output formatting

Here, the curly braces {} are used as placeholders. We can specify the order in which they are printed by using numbers (tuple index).

```
print('I love {0} and {1}'.format('bread','butter'))

print('I love {1} and {0}'.format('bread','butter'))

I love bread and butter

I love butter and bread
```

Input and Output

We can even use keyword arguments to format the string.

```
>>> print('Hello {name}, {greeting}'.format(greeting = 'Good morning', name = 'John'))
Hello John, Good morning
```

Input and Output

Python Input

Up until now, our programs were static. The value of variables was defined or hard coded into the source code.

To allow flexibility, we might want to take the input from the user. In Python, we have the input() function to allow this. The syntax for input() is:

```
input([prompt])
```

Input and Output

where prompt is the string we wish to display on the screen. It is optional.

```
>>> num = input('Enter a number: ')
Enter a number: 10
>>> num
'10‘
```

Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use int() or float() functions.

Input and Output

```
>>> int('10')
10
>>> float('10')
10.0
```

Input and Output

Python Import

When our program grows bigger, it is a good idea to break it into different modules.

A module is a file containing Python definitions and statements. Python modules have a filename and end with the extension .py.

Definitions inside a module can be imported to another module or the interactive interpreter in Python. We use the import keyword to do this. For example, we can import the math module by typing the following line:

Input and Output

import math

We can use the module in the following ways:

```
import math
print(math.pi)
3.141592653589793
```

Now all the definitions inside math module are available in our scope. We can also import some specific attributes and functions only, using the from keyword. For example:

```
>>> from math import pi
>>> pi
3.141592653589793
```