Module 2: Python Programming Fundamentals

Premanand S

Assistant Professor,
School of Electronics and Engineering,
Vellore Institute of Technology, Chennai

premanand.s@vit.ac.in

August 9, 2024

Topics to be covered in Module 2,

- Introduction to Python
- Interactive and Script mode
- Indentation
- Comments
- Variables
- Reserved words
- Data Types
- Operators and Precedence
- Expressions Built-in functions
- Importing from packages

Keywords

Keywords

Example (Python Snippet)

```
import keyword
print(keyword.kwlist)
```

Identifiers

Identifiers

- Identifiers are names used to identify a variable, function, class, module, or other object.
- Helps us to differentiate from other names

Identifiers - Norms

- Combination of a-z, A-Z, 0-9 and _
- Cannot start with numbers
- Keywords cannot be used
- Special characters like , , #, $, % not allowed
- Can be of any length
- Case sensitive, Eg: Name name are not same
- Name should be anything but with sense is a good programming habit. Eg: a=‘prem’ can be name =‘prem’
- Use lowercase letters for variable names and functions, and separate words with underscores (snake_case). For example, calculate_average, what_is_your_name
- Use uppercase letters for constants. For example, MAX_SCORE
- Use CamelCase (or PascalCase) for class names. For example, StudentRecord.

Valid Identifiers

Example (Python Snippet)

```
name = "Python"
age = 34
_temperature = 22.5
total_score = 95
MAX_LIMIT = 100

class PythonClass:
    pass
```

Invalid Identifiers

Example (Python Snippet)

```
2nd_value = 50
total-score = 90
def = 10
```

Comments

Comments

- Important while writing program
- Describes about the content of the segment of program or whole program
- # - symbol for comments
- Python interpreter ignores the comment line
- Types : Single line & Multi-line

Single line comment

Example (Python Snippet)

```
# This is a single-line comment
x = 10  # This is an inline comment
```

Multi-line comment

Example (Python Snippet)

```
# This is a multi-line comment
# that spans multiple lines
# using consecutive single-line comments.
name = 'Python'
professional = 'Engineering'
```

Multi-line comment

Example (Python Snippet)

```
"""
This is a multi-line comment
that spans multiple lines
using triple quotes.
"""
```

Docstring

Docstring

- Docstrings are a special kind of comment used to document modules, functions, classes, and methods.
- hey are written using triple quotes and can span multiple lines.
- Docstrings are accessible at runtime using the __doc__ attribute.

Docstring

Example (Python Snippet)

```
def add(a, b):
    """
    This function adds two numbers and returns the result.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The sum of the two numbers.
    """
    return a + b

print(add.__doc__)
```

Feature	Comments	Docstrings
Purpose	Explain specific lines or blocks of code	Document modules, classes, methods, and functions
Syntax	# for single-line comments	Triple quotes (''') or (""") for multi-line
Multi-line Format	Consecutive # or triple quotes	Triple quotes
Runtime Access	Not accessible at runtime	Accessible via `__doc__` attribute
Placement	Anywhere in the code	First statement within a module, class, method, or function
Length	Typically brief	Can be detailed and extensive

Table: Comparison of Comments and Docstrings in Python

Statement

Satement

- Statement is a unit of code that performs an action.
- Each statement in Python is executed by the interpreter, and they form the building blocks of Python programs

Statement - Types

- Expression Statements: x=5
- Assignment Statements: num1 = 25
- Control Flow Statements: if statements: if condition
- Control Flow Statements: for loops: for item in iterable
- Control Flow Statements: while loops: while condition
- Function statement: def my_function(): pass
- Class statement: class my_class(): pass
- Import statement: import math or from math import sqrt
- Return statement: return num1
- Break statement: break
- Continue statement: continue
- Pass statement: pass

Expression Statement

Example (Python Snippet)

```
print("Hello, World!")
```

Assignment Statement

Example (Python Snippet)

```
number = 10
```

Control Flow Statement (if statement):

Example (Python Snippet)

```
if number > 5:
    print("Number is greater than 5")
```

Loop Statement (for loop):

Example (Python Snippet)

```
for i in range(5):
    print(i)
```

Function Definition:

Example (Python Snippet)

```
def greet(name):
    print(f"Hello, {name}!")
```

Class Definition:

Example (Python Snippet)

```
class Person:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print(f"Hello, my name is {self.name}")
```

Import Statement:

Example (Python Snippet)

```
import math
print(math.sqrt(16))
```

Return Statement:

Example (Python Snippet)

```
def add(a, b):
    return a + b
```

Break and Continue Statements:

Example (Python Snippet)

```
for i in range(10):
    if i == 5:
        break  # Exit the loop when i equals 5
    print(i)

for i in range(10):
    if i % 2 == 0:
        continue  # Skip even numbers
    print(i)
```

Pass Statement:

Example (Python Snippet)

```python
def empty_function():
    pass  # Placeholder for future code
```

Print

Print

- print() function in Python is used to display output to the console.
- It can handle various types of data and allows you to format the output in several ways.

Print - Syntax

Example (Python Snippet)

```
print(*objects, sep=' ', end='\n')
```

Print - String

Example (Python Snippet)

```
print(" <statement> ")
print('<statement>')
String ?
```

Print - Try for the output

Example (Python Snippet)

```
print("Hello world!)
print("Hello world!)
print("Hello world!')
```

Print - Modifiers

Example (Python Snippet)

```
print("She said:"Hello" and then left")    ?
```

Print - Modifiers

Example (Python Snippet)

```
print('She said:"Hello" and then left')

print("She said: 'Hello' and then left")

print("She said:\"Hello\" and then left")
```

Multiple line to print

Example (Python Snippet)

```
print('Hello world!')
print('Hello world!')
print('Hello world!')
?
```

Multiple line to print

Example (Python Snippet)

```
print('Hello world!\n Hello world! \n Hello world!')
```

Concatenation

Example (Python Snippet)

```
print('Hello' + 'Python')
print('Hello'+' '+'Python')
print('Hello' + ' Python')
print('Hello'+ '      Python')
```

Spacing

Example (Python Snippet)

```
print('Hi guys! did you notice something?')
  print('?')    #one space or indentation space ?



     print('Did you see now?') # new cell
```

Debug it!

Example (Python Snippet)

```
print(Day1 - String Manipulation’)
print(’String concatenation is done with ’+’ sign’)
print(’eg. print(’hello’+’world’)’)
print((’newlines can be created with a backslash and n’)
```

Solution - Debug it!

Example (Python Snippet)

```
print('Day1 - String Manipulation')
print("String concatenation is done with '+' sign")
print('eg. print(\'hello'+'world\')')
print('newlines can be created with a backslash and n')
```

Indentation

Indentation

- Indentation in Python is a critical part of the syntax and is used to define the structure and flow of the code.
- Unlike many other programming languages that use braces or keywords to define blocks of code, Python uses indentation levels to determine the grouping of statements.

Indentation - Basic Example

Example (Python Snippet)

```
if True:
    print("This is indented to show it is part of if block.")
    print("This line is also part of the if block.")
print("This line is not indented, so it’s outside if block.")
```

Indentation - Function Definition

Example (Python Snippet)

```
def greet(name):
    print(f"Hello, {name}!")
    print("Welcome to the Python tutorial.")
```

Indentation - Class Definition

Example (Python Snippet)

```
class Person:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print(f"Hello, my name is {self.name}")
```

Indentation - Loops

Example (Python Snippet)

```
for i in range(5):
    print(i)
    if i % 2 == 0:
        print(f"{i} is even.")
```

Indentation - Conditional Statements

Example (Python Snippet)

```
x = 10
if x > 5:
    print("x is greater than 5.")
    if x < 15:
        print("x is also less than 15.")
```

Input function

Input function

- The input() function in Python is used to take input from the user.
- It reads a line from the input (usually from the keyboard), converts it to a string (if necessary), and returns that string.

Input vs Print functions

Example (Python Snippet)

```
print('What is your name?')

input('What is your name?')
```

Input + print + comment functions

Example (Python Snippet)

```
#printing by getting the data from user
print('Hello' +' '+ input('What is your name?'))
```

Len function

Len function

- The len() function in Python is used to determine the length (i.e., the number of items) of an object.
- The object can be a sequence (such as a string, list, tuple, or range) or a collection (such as a dictionary, set, or frozen set).

Len functions

Example (Python Snippet)

```
string = "Hello, world!"
print(len(string))
```

Print + input + Len functions

Example (Python Snippet)

```
name = input('Whats your name?')
print(len(name))

or

print(len(input('whats your name?')))
```

Variables

Variables

- Variables are used to store data that can be referenced and manipulated in a program.
- A variable is essentially a name given to a data value, and once you assign a value to a variable, you can use that variable name to access the stored value.
- Dynamically typed
- Assigning values
- Case-sensitive
- Naming conventions as like Identifiers

Importance of variables

Example (Python Snippet)

```
input('Whats your contact number?')

python_sir = input('Whats your contact number?')

print(python_sir)
```

Why the name - VARIABLE?

Example (Python Snippet)

```
name = 'Prem'
name = 'Anand'
name = 'Premanand'
print(name)
```

Why the name - VARIABLE?

Example (Python Snippet)

```
name = 'Prem'
name = 38
name = 'Python'
name = 2.0
print(name)
```

Problem1: Swapping the strings from users data, by using temporary variable

Example (Python Snippet)

```
Input:
string1 = 'Hello'
string2 = 'World'

Output:
string1 = 'World'
string2 = 'Hello'
```

Answer1: Swapping the strings from users data, by using temporary variable

Example (Python Snippet)

```
string1 = input('Whats the first string?')
string2 = input('Whats the second string?')

string3 = string1
string1 = string2
string2 = string3

print('First string:', string1)
print('Second string:', string2)
```

QUIZ 1

Example (Understanding)

```
1. Which line of Python code is valid?
a. var a = 2
b. a = 12
c. a:12
d. 12 = a
```

QUIZ 2

Example (Understanding)

```
2. Which is the best variable name for players1 username?
a. p1 user name = 'ghilli_boyz'
b. 1_player_username = 'ghilli_boyz'
c. player1_username = 'ghilli_boyz'
d. ply1 = 'ghilli_boyz'
```

Problem2: Write a Python program

Example (Apply print, input, variables)

```
You're working on a fun project to generate unique band
names. The program asks the user for the name of the city
they grew up in, their pet's name, and their favorite music
genre. It then combines these inputs to suggest a
potential band name.
```

Answer2: Band name generation!

Example (Apply print, input, variables)

```
print('Welcome to the band name generator!')
city = input('What is the name of the city you grew up in?\n')
pet_name = input('What is the name of your pet?\n')
music_genre = input('What is your favorite music genre?\n')
print('Your band name could be ' + city + '_' + pet_name +
'_' + music_genre)
```