upSkill CAMPUS

[Your College Logo]

uct

GGI

Sharpening Minds. Brightening Tomorrow

GULZAR GROUP OF INSTITUTES

SECOND WEEK

PYTHON INTERNSHIP

[Your College Logo]

“INTRODUCTION TO PYTHON”-CONCEPTUAL

(INDUSTRIAL REPORT-WEEK 2)

Prepared by

[Shivam Shriwastav]

Email id:Shivam808047@gmail.com

Executive Summary

This report provides details of the Industrial Internship provided by Upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship is focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.

This weekly report explains the introduction to Python with important concepts.

-E-Book learning Python

-Conditional statement in Python (if, if else, elif if-else)

TABLE OF CONTENTS

1 Preface

Summary of the 2nd week's work.

I undertook this internship project and completed the 2nd-week internship report under the guidance of this associated company. I am grateful to all for their patience and assistance during my online training at their Virtual site named" Upskill Campus". It was a good learning experience for me to work on their weekly project, as the project involved many innovative practices.

1.1.1 Information about the internship position

I joined Upskill campus for an internship program in the position of a **Python Intern**. While the central focus was on focusing in this program wisely and learn effectively, I also handled various other tasks as they occurred.

I want to thank my advisers and everyone at the company for their patience and assistance during my on-site training. Thanks to their guidance, I was able to develop [**PYTHON SKILLS**] and learn about [**PYTHON**]. These skills would help me to expand my resume and advance my career.

.

2 Introduction:Python

Basic concepts:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

There are two major Python versions: Python 2 and Python 3. Both are quite different.

3 E-Book : Learning Python

3.1 History of Python:

The programming language Python was conceived in the late 1980s,[1] and its implementation was started in December 1989[2] by Guido van Rossum at CWI in the Netherlands as a successor to ABC capable of exception handling and interfacing with the Amoeba operating system.[3] Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, *Benevolent Dictator for Life* (BDFL).[4][5] (However, Van Rossum stepped down as leader on July 12, 2018.[6]). Python was named after the BBC TV show *Monty Python's Flying Circus*.[7]

Python 2.0 was released on October 16, 2000, with many major new features, including a cycle-detecting garbage collector (in addition to reference counting) for memory management and support for Unicode. However, the most important change was to the development process itself, with a shift to a more transparent and community-backed process.[8]

Python 3.0, a major, backwards-incompatible release, was released on December 3, 2008[9] after a long period of testing. Many of its major features have also been backported to the backwards-compatible, though now-unsupported, Python 2.6 and 2.7.[10]

3 .2 How to run Python Program?

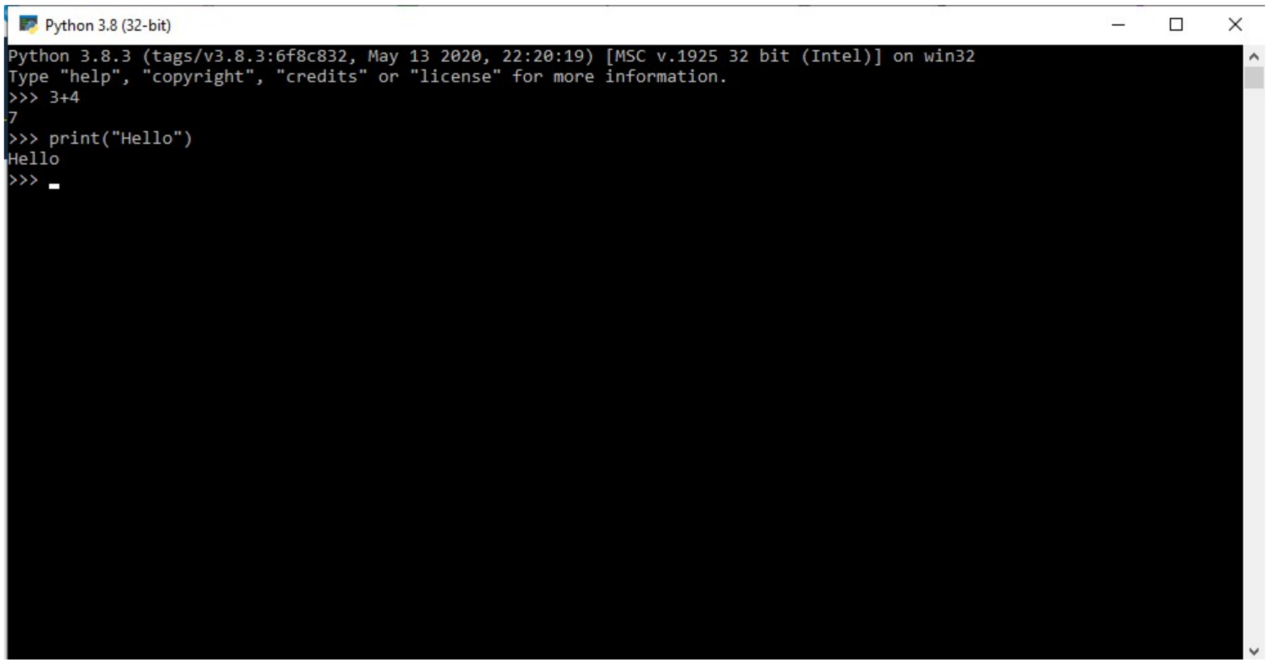
Python programmers must know every possible way to run the Python scripts or code. This is the only way to verify whether code is working as we want. Python interpreter is responsible for executing the Python scripts. Python interpreter is a piece of software which works between the Python program and computer hardware. Here we are describing the series of ways to run Python scripts.

- The operating system command-line or terminal.
- The Python interactive mode.
- The IDE or Text editor
- The file manager of system.

The Python interactive mode

To run the Python code, we can use the Python interactive session. We need to start Python interactive session, just open a command-line or terminal in start menu, then type in python, and press enter key.

Here is the example of how to run Python code using interactive shell.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+4
7
>>> print("Hello")
Hello
>>> _
```

It allows us to check every piece of code, and this facility makes it an awesome development tool. But once we close the session it will lose all code that we have written.

Below are the few options to exit the interactive mode.

- Type built-in functions **quit()** or **exit()**. Or
- Type the enter **ctrl+ Z** key combination to end the current session of Python interactive shell.

3.3 Built-in Types

The following sections describe the standard types that are built into the interpreter.

The principal built-in types are numerics, sequences, mappings, classes, instances and exceptions.

Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but `None`.

Some operations are supported by several object types; in particular, practically all objects can be compared for equality, tested for truth value, and converted to a string (with the `repr()` function or the slightly different `str()` function). The latter function is implicitly used when an object is written by the `print()` function.

3.4 Truth Value Testing

Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations below.

By default, an object is considered true unless its class defines either a `__bool__()` method that returns `False` or a `__len__()` method that returns zero, when called with the object. 1 Here are most of the built-in objects considered false:

- constants defined to be false: `None` and `False`
- zero of any numeric type: `0`, `0.0`, `0j`, `Decimal(0)`, `Fraction(0, 1)`
- empty sequences and collections: `''`, `()`, `[]`, `{}`, `set()`, `range(0)`

Operations and built-in functions that have a Boolean result always return `0` or `False` for false and `1` or `True` for true, unless otherwise stated. (Important exception: the Boolean operations `or` and `and` always return one of their operands.)

3.5 Python Version List

Python programming language is being updated regularly with new features and supports. There are lots of update in Python versions, started from 1994 to current release.

A list of Python versions with its released date is given below.

Python Version	Released Date
Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003
Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008

Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012
Python 3.4	March 16, 2014
Python 3.5	September 13, 2015
Python 3.6	December 23, 2016
Python 3.7	June 27, 2018
Python 3.8	October 14, 2019

3.6 Tips to Keep in Mind While Learning Python

The most common question asked by the beginners - **"What is the best way to learn Python"?** It is the initial and relevant question because first step in learning any programming language is to know how to learn.

The proper way of learning will help us to learn fast and become a good Python developer.

In this section, we will discuss various tips that we should keep in mind while learning Python.

3.6.11. Make it Clear Why We Want to Learn

The goal should be clear before learning the Python. Python is an easy, a vast language as well. It includes numbers of libraries, modules, in-built functions and data structures. If the goal is unclear then it will be a boring and monotonous journey of learning Python. Without any clear goal, you perhaps won't make it done.

So, first figure out the motivation behind learning, which can anything be such as knowing something new, develop projects using Python, switch to Python, etc. Below are the general areas where Python is widely used. Pick any of them.

- Data Analysis and Processing
- Artificial Intelligence
- Games
- Hardware/Sensor/Robots
- Desktop Applications

Choose any one or two areas according to your interest and start the journey towards learning Python.

4 Conditional Statements in Python:

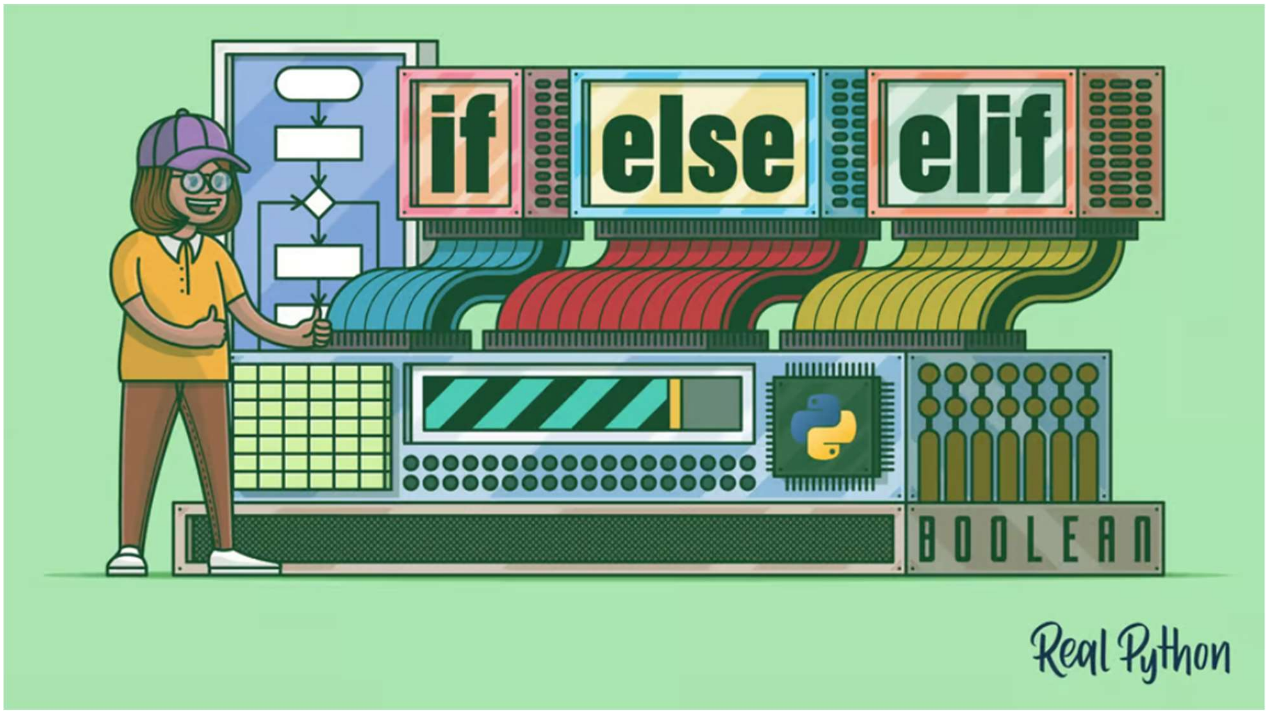


Table of Contents

4.1 Introduction to the `IF` Statement

We'll start by looking at the most basic type of `if` statement. In its simplest form, it looks like this:

```
if <expr>:
    <statement>
```

In the form shown above:

- `<expr>` is an expression evaluated in a Boolean context, as discussed in the section on Logical Operators in the Operators and Expressions in Python tutorial.
- `<statement>` is a valid Python statement, which must be indented. (You will see why very soon.)

If `<expr>` is true (evaluates to a value that is "truthy"), then `<statement>` is executed. If `<expr>` is false, then `<statement>` is skipped over and not executed.

Note that the colon (`:`) following `<expr>` is required. Some programming languages require `<expr>` to be enclosed in parentheses, but Python does not.

Here are several examples of this type of `if` statement:

```
>>>
>>> x = 0
>>> y = 5

>>> if x < y:                            # Truthy
...     print('yes')
...
yes
>>> if y < x:                            # Falsy
...     print('yes')
...
```

```
>>> if x:                             # Falsy
...     print('yes')
...
>>> if y:                             # Truthy
...     print('yes')
...
yes

>>> if x or y:                        # Truthy
...     print('yes')
...
yes
>>> if x and y:                       # Falsy
...     print('yes')
...

>>> if 'aul' in 'grault':             # Truthy
...     print('yes')
...
yes
>>> if 'quux' in ['foo', 'bar', 'baz']:  # Falsy
...     print('yes')
...
```

Note: If you are trying these examples interactively in a REPL session, you'll find that, when you hit `Enter` after typing in the `print('yes')` statement, nothing happens.

Because this is a multiline statement, you need to hit `Enter` a second time to tell the interpreter that you're finished with it. This extra newline is not necessary in code executed from a script file.

4.2 The `else` and `Elif` Clauses

Now you know how to use an `if` statement to conditionally execute a single statement or a block of several statements. It's time to find out what else you can do.

Sometimes, you want to evaluate a condition and take one path if it is true but specify an alternative path if it is not. This is accomplished with an `else` clause:

```
if <expr>:
    <statement(s)>
else:
    <statement(s)>
```

If `<expr>` is true, the first suite is executed, and the second is skipped. If `<expr>` is false, the first suite is skipped and the second is executed. Either way, execution then resumes after the second suite. Both suites are defined by indentation, as described above.

In this example, `x` is less than `50`, so the first suite (lines 4 to 5) are executed, and the second suite (lines 7 to 8) are skipped:

>>>

```
 1>>> x = 20
 2
 3>>> if x < 50:
 4...     print('(first suite)')
 5...     print('x is small')
 6... else:
 7...     print('(second suite)')
 8...     print('x is large')
 9...
10(first suite)
11x is small
```

Here, on the other hand, x is greater than 50, so the first suite is passed over, and the second suite executed:

```
>>>
>>> x = 120
>>>
>>> if x < 50:
...     print('(first suite)')
...     print('x is small')
... else:
...     print('(second suite)')
...     print('x is large')
...
(second suite)
x is large
```

There is also syntax for branching execution based on several alternatives. For this, use one or more `elif` (short for *else if*) clauses. Python evaluates each `<expr>` in turn and executes the suite corresponding to the first that is true. If none of the expressions are true, and an `else` clause is specified, then its suite is executed:

```
if <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
elif <expr>:
    <statement(s)>
    ...
else:
    <statement(s)>
```

An arbitrary number of `elif` clauses can be specified. The `else` clause is optional. If it is present, there can be only one, and it must be specified last:

```
>>>
>>> name = 'Joe'
>>> if name == 'Fred':
```

```
...     print('Hello Fred')
... elif name == 'Xander':
...     print('Hello Xander')
... elif name == 'Joe':
...     print('Hello Joe')
... elif name == 'Arnold':
...     print('Hello Arnold')
... else:
...     print("I don't know who you are!")
...
Hello Joe
```

At most, one of the code blocks specified will be executed. If an `else` clause isn't included, and all the conditions are false, then none of the blocks will be executed.

Note: Using a lengthy `if/elif/else` series can be a little inelegant, especially when the actions are simple statements like `print()`. In many cases, there may be a more Pythonic way to accomplish the same thing.

Here's one possible alternative to the example above using the `dict.get()` method:

>>>

```
>>> names = {
...     'Fred': 'Hello Fred',
...     'Xander': 'Hello Xander',
...     'Joe': 'Hello Joe',
...     'Arnold': 'Hello Arnold'
... }

>>> print(names.get('Joe', "I don't know who you are!"))
Hello Joe
>>> print(names.get('Rick', "I don't know who you are!"))
I don't know who you are!
```

Recall from the tutorial on Python dictionaries that the `dict.get()` method searches a dictionary for the specified key and returns the associated value if it is found, or the given default value if it isn't.

An if statement with elif clauses uses short-circuit evaluation, analogous to what you saw with the and and or operators. Once one of the expressions is found to be true and its block is executed, none of the remaining expressions are tested. This is demonstrated below:

```
>>>
>>> var  # Not defined
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    var
NameError: name 'var' is not defined

>>> if 'a' in 'bar':
...     print('foo')
... elif 1/0:
...     print("This won't happen")
... elif var:
...     print("This won't either")
...
foo
```

The second expression contains a division by zero, and the third references an undefined variable var. Either would raise an error, but neither is evaluated because the first condition specified is true.

4.3 Code submission (Github link) :

https://github.com/shivam808047/python_internship/blob/main/quiz%20game%20by%20python.py

4.4 Report submission(Github link):

5.My learnings….

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

Uses:

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

GOOD TO KNOW….

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.