PYTHON REVISION TOUR-I

CHAPTER-1

INTRODUCTION

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.

What can Python do?

- 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 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.

Good to know

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.

Topic One

Tokens in Python

Variables & Assignments

Data Types

PYTHON COMMENTS

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Creating a Comment

Comments starts with a #, and Python will ignore them:

```
#This is a comment
print("Hello, World!")
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
''' This is a multiline comment. '''
```

Variables in Python

Variables are containers for storing data values. A variable can have a short name (like x and y) or a more descriptive name (age, rollno, total_marks).

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Assigning Multiple Values To Python Variables

Python allows you to assign values to multiple variables in one line:

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

And you can assign the *same* value to multiple variables in one line:

x = y = z = "Orange"

Operators

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Data Types:

Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

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

You can get the data type of any object by using the type() function.

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

- int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Topic Two

Decision Making in Python

Looping Structures

DECISION MAKING IN PYTHON

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

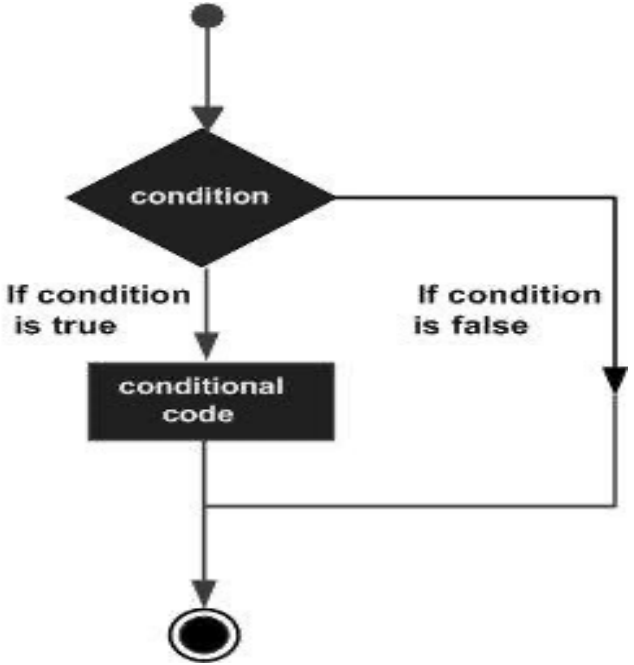
Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

In Python we can achieve decision making by using the below statements:

- ❑ If statements
- ❑ If-else statements
- ❑ Elif statements
- ❑ Nested if and if-else statements
- ❑ Elif ladder

Decision Making Statements

- if statements: An **if statement** consists of a boolean expression followed by one or more statements.
- if...else statements:An **if statement** can be followed by an optional **else statement**, which executes when the Boolean expression is FALSE.
- nested if statementsYou can use one **if** or **else if** statement inside another **if** or **else if** statement(s).



Example

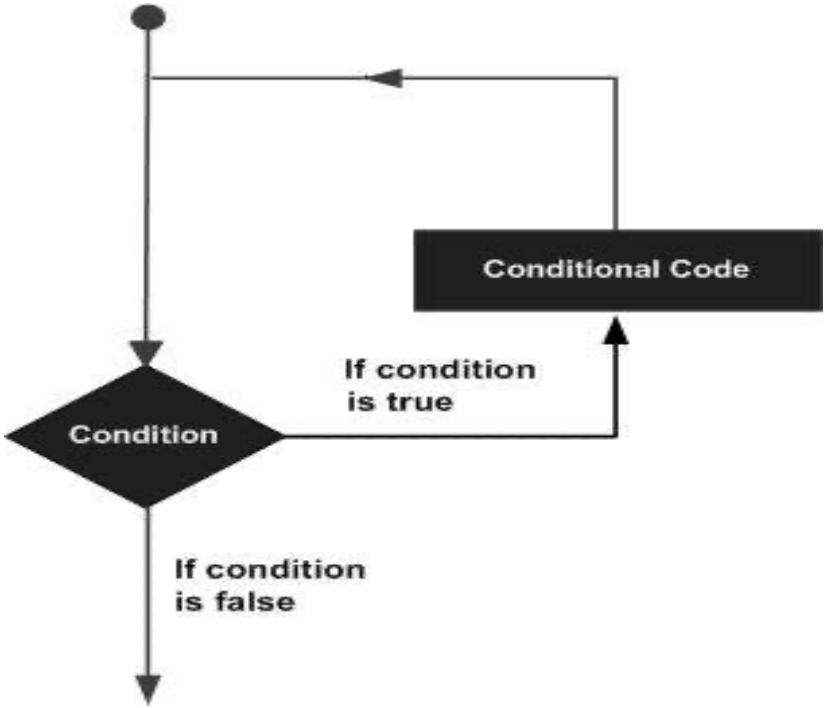
```
x = int(input(“Enter any integer value: "))
if x < 0:
   x = 0
   print('Negative changed to zero')
elif x == 0:
   print('Zero')
elif x == 1:
   print('Single')
else:
   print('More‘)
```

Python Loops

Loops are used in programming to repeat a specific block of code.

Python has two primitive loop commands:

- for loops
- while loops



for loop in Python

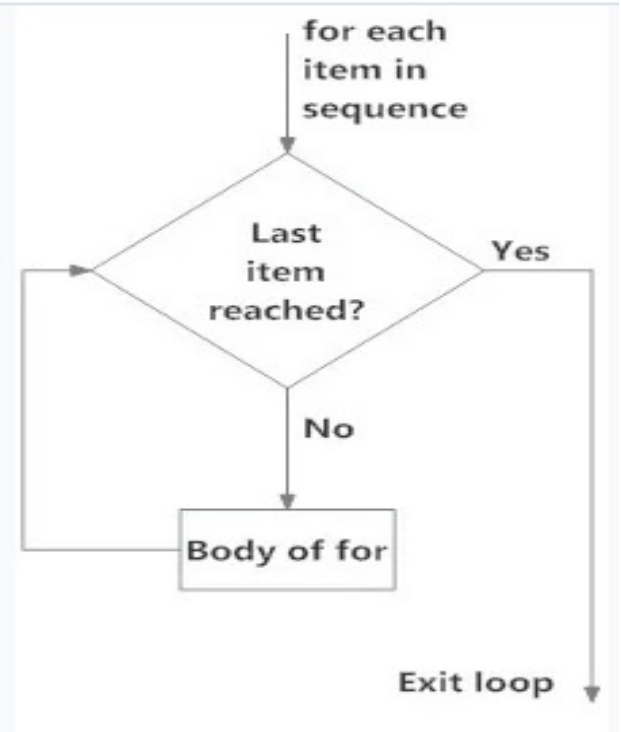
The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called traversal.

Syntax:

for val in sequence:

    Body of for

Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.



The range() function

- We can generate a sequence of numbers
using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).

- We can also define the start, stop and step size as range(start, stop, step_size); where step_size defaults to 1 if not provided.

- We can use the range() function in for loops to iterate through a sequence of numbers. It can be combined with the len() function to iterate through a sequence using indexing.

for loop with else

- A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop exhausts.
- The break keyword can be used to stop a for loop. In such cases, the else part is ignored.
- Hence, a for loop's else part runs if no break occurs.

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

```
adj = ["red", "big", "tasty"]
fruits =["apple", “mango", “kiwi"]
for x in adj:
  for y in fruits:
          print(x, y)
```

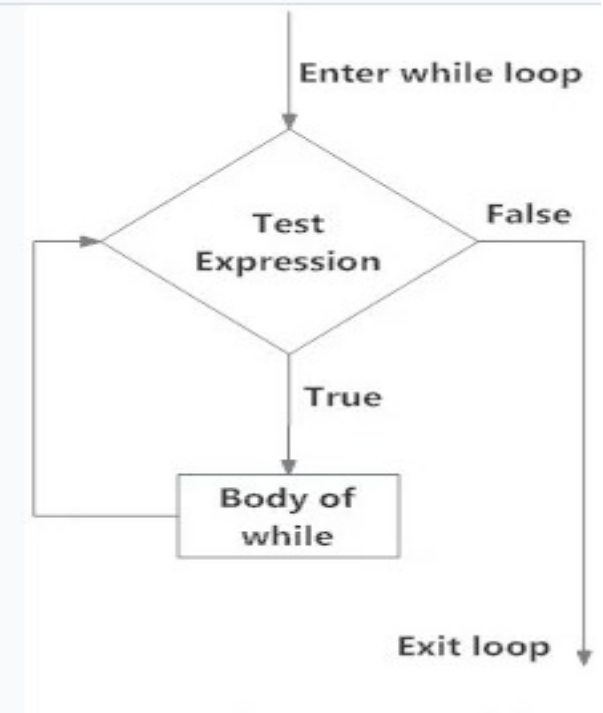
while loop in Python

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

We generally use this loop when we don't know the number of times to iterate beforehand.

Syntax:

```
while test_expression:
    Body of while
```



while loop with else

- Same as with for loops, while loops can also have an optional else block.
- The else part is executed if the condition in the while loop evaluates to False.
- The while loop can be terminated with a break statement. In such cases, the else part is ignored. Hence, a while loop's else part runs if no break occurs and the condition is false.

Example

```
i = 1

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

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements:

- break statement: Terminates the loop statement and transfers execution to the statement immediately following the loop.
- continue statement: Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
- pass statement: The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. Or in simple terms, for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

Thanks !!!

You can contact at anu.arora@gnauniversity.edu.in

Any Questions???