Computational Thinking Using Python

CSE1500

[L-T-P-C: 2-0-2-3]

MODULE – 1

Introduction to Computational thinking

Introduction to Python

About Python

- Python was developed by Guido van Rossum at National Research Institute for Mathematics and Computer Science in Netherlands in 1990.
- Rossum wanted the name of his new language to be short, unique and mysterious.
- Inspired by Monty Python's Flying Circus, a BBC comedy series, he named the language Python.

Guido developed Python by taking almost all programming features from different languages

- Functional Programming Features from C
- Object Oriented Programming Features from C++
- Scripting Language Features from Perl and Shell Script
- Python is recommended as first programming language for beginners.

Python Official Logo

Guido van Rossum

Where we can use Python

The most common important and real world application areas of Python are

- Desktop applications
- Web applications
- Database applications
- Network Programming
- Developing games
- Data Analysis applications
- Machine Learning
- Artificial Intelligence applications
- IOT applications

Top Software companies like Google, Microsoft, IBM, Yahoo using Python.

Guido van Rossum

Features of Python

Simple – Syntax is close to English, making code easy to read and write.

Easy to Learn – Few keywords and simple structure; familiar to C programmers.

Open Source – Free to use, modify, and distribute.

High-Level – Uses English-like commands, easy to understand and maintain.

Dynamically Typed – No need to declare variable types; can change types at runtime.

Platform Independent – Runs on any OS via Python Virtual Machine.

Interpreted – We are not required to compile Python programs explicitly. Internally Python interpreter will take care that compilation.

Portable – Python programs run the same on any system via PVM, though OS-specific modules may affect portability.

Procedure & Object-Oriented – Supports both function-based (procedural) and class-based (object-oriented) programming.

Flavors of Python

- CPython – Standard Python for C applications.
- Jython/JPython – For Java applications, runs on JVM.
- IronPython – For C#.NET platform.
- PyPy – Faster execution with built-in JIT compiler.
- RubyPython – For Ruby platforms.
- AnacondaPython – Designed for handling large-scale data processing, predictive analytics and scientific computing, it is called Anaconda Python.

Python Versions:

- Python Versions:
- Python 1.0V introduced in Jan 1994
- Python 2.0V introduced in October 2000
- Python 3.0V introduced in December 2008
- Current Stable: Python 3.13.6 (Aug 6, 2025)

Limitations of Python:

- Performance is not up to the mark because Python is an interpreted language.
- Not commonly used for mobile applications.

Some of the differences between Python 2.x and Python 3.x

1) Print Statement vs. Print Function:
Python 2.x (release year 2000) : print "Hello" (statement)
Python 3.x (release year 2008) : print("Hello") (function, must use parentheses)

2)Integer Division:
Python 2.x: Dividing two integers performs floor division (5 / 2 = 2)
Python 3.x: Dividing integers results in float (5 / 2 = 2.5)
Use // for floor division in Python 3.x to replicate Python 2.x behavior.

3) input() Function:
Python 2.x: input() evaluates expression, raw_input() for string input
Python 3.x: input() reads input as a string.

Some of the differences between Python 2.x and Python 3.x

4) Integer Types:
Python 2.x: int and long types separately
Python 3.x: Only one int type (no separate long)

5) Unicode Handling:
Python 2.x: Strings are ASCII by default, u"string" is Unicode.
Python 3.x: Strings are Unicode by default, b"bytes" for byte data.

Overall Recommendation: Python 3.x is the future, and Python 2.x has reached end-of-life support as of January 1, 2020. Most new projects should use Python 3.x.

Note: The latest stable version is Python 3.13.7, released on August 14, 2025

Charactet Set, Tokens

Python Character set:

Python uses the following character set:

- Letters: Upper case and lower case letters
- Digits: 0,1,2,3,4,5,6,7,8,9
- Special Symbols: Underscore (_), (,), [,], {,}, +, -, *, &, ^, %, $, #, !, Single quote(‘), Double quotes(“), Back slash(\), Colon(:), and Semi Colon (;)
- White Spaces: (‘\t\n\x0b\x0c\r’), Space, Tab.

TOKENS :

- Keywords
- Literals
- Punctuators / Delimiter
- Identifiers
- Operators

Character Set, Tokens

Keywords: Python has the following keywords.

FALSE	None	TRUE	and
as	assert	async	await
break	class	continue	def
elif	else	except	finally
for	from	global	if
import	in	is	lambda
nonlocal	not	or	pass
raise	return	try	while
with	yield	del	

Identifiers / Variables :

Identifier is the name given to various program elements such as variable, array, function, class or other objects. All identifiers must obey the following rules.

- Is a sequence of characters that consists of letters, digits and underscore
- Can be of any length
- Starts with a letter which can be either lower or upper case
- Can start with an underscore '_'
- Cannot start with a digit
- Cannot be a keyword.

Charactet Set, Tokens

Literals / Constants :

In Python, a literal is a fixed value directly written in the code, representing data. They are the actual values assigned to variables or used in expressions.

Types of literals in Python:

- Numeric literals (integer, float, complex)
- Ex: 100, -25, 3.14, -0.75, 4 + 3j
- Boolean literals

  Ex. True, False

- String literals:

  Ex. “Presidency”
  ‘ Bengaluru’

- Special Literal

  Ex. None

Operators in Python

Operators :

An operator is a symbol that performs an operation. Python has the following operators.

- Arithmetic operators
- Assignment operators
- Unary minus operator
- Relational operators
- Logical operators
- Boolean operators
- Bitwise operators

Punctuators / Delimiters Python

Punctuators / Delimiters Python

In Python, punctuators (sometimes called delimiters) are symbols that perform a special role in Python like grouping, punctuation and assignment. Python uses the following symbols and symbol combinations as delimiters.

( ) [ ] { }

, : . ‘ = ;

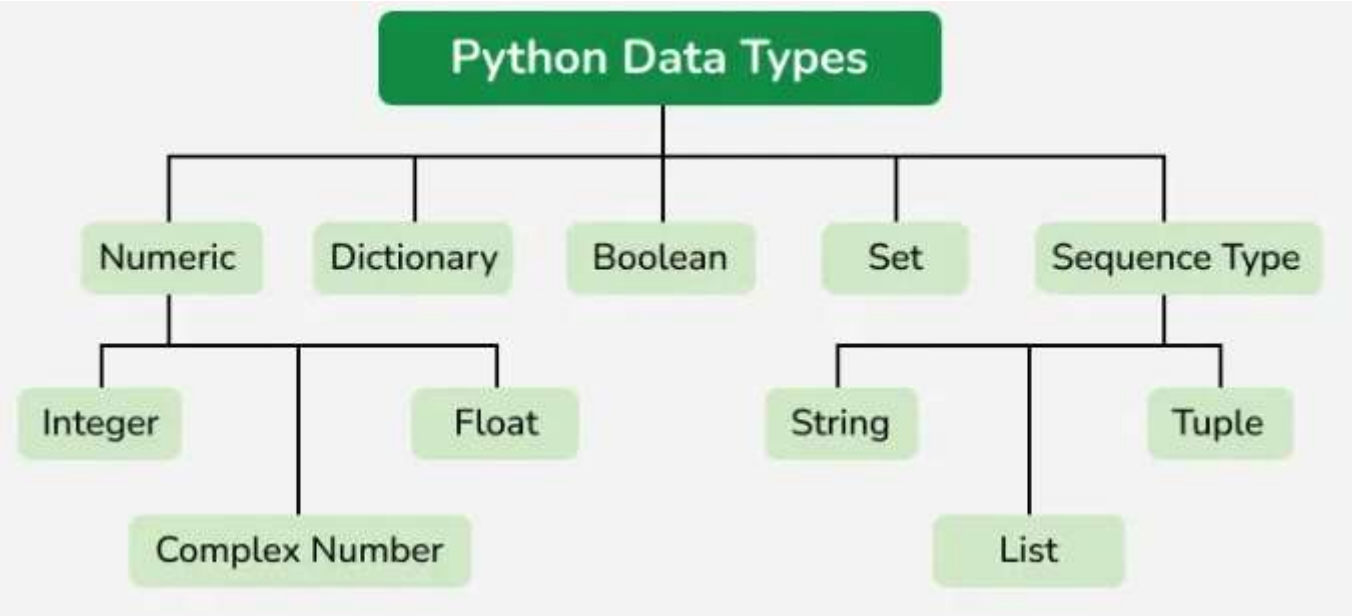
+= -= *= /= //= %=

&= |= ^= >>= <<= **=

Data types in Python

Data type determines the set of values that a data item can take and the operations that can be performed on the item.

In other words, the type of data value we can store in an *identifier* such as a variable, is called its data type.



Data types in Python

The following are standard or built-in data types in Python:

- Numeric - int, float, complex
- Sequence Type - string, list, tuple
- Mapping Type - dict
- Boolean - bool
- Set Type - set, frozenset
- Binary Types - bytes, bytearray

Note: *In Module 1, we will discuss the int, float, complex, string, and bool data types only. Other data types will be discussed in subsequent modules*.

Input / output operations (functions) in python

Input Statement:

To accept input from keyboard, Python provides the input() function. This function takes a value from the keyboard and returns it as a string.

Note : By default, it returns the user input in form of a string.

For example:

```
name = input()   # this will wait till we enter a string

print(str)

Raj kumar
```

Input / output operations (functions) in python

Inside the input() function, user may give a prompt text. When you run the program it displays the prompt message

Example 1: name = input("Enter your name : ") # name will be in string form

Example 2: marks = int(input("Enter your marks "))

we use int() built-in function from Python's standard library. It converts a string object to an integer (type cast it to integer).

Example 3: amount = float(input("Enter Amount : "))

Similarly Python's float() function converts a string into a float.

print () statement / function:

To display output or results, Python provides the print() function. This function can be used in different formats which are discussed hereunder.

1) The print("string") Statement

A string represents a group of characters. When a string is passed to the print() function, the string is displayed as it is.

Example: print("Hello Dear")

Hello Dear

Input / output operations (functions) in python

2) The print() function issues a newline character ('\n') at the end, by default. To use any other character as a separator, define a **sep** parameter for the print() function

```
city="Bengaluru“
state="Karnataka“
country="India“
print(city, state, country, sep=',’)
```

Output: Bengaluru,Karnataka,India

3) We can use escape characters also

```
print("Hello \n World")
print("Hello\tWorld")
```

Input / output operations (functions) in python

4) We can use repetetion operator (*) in the string

```
print(5*"Presi")
print("Presi"*5)
```

Output : PresiPresiPresiPresiPresi

5) We can use + operator also

```
print("Presidency"+"University")
```

Output : PresidencyUniversity

Input / output operations (functions) in python

6) print using the format() Method

We can format output using the format() method, it is part of the built-in string class and allows for complex variable substitutions and value formatting.

Syntax of str.format() function is:

{}{}.format(value-1, value-2,...,value-n)

where {}{} are place holders created by the user in the string.

value1,value2,...,valuen They could be variables, integers, floating-point numbers, strings, characters etc.

Example 1)

```
name="Skanda"
age=18
print ("my name is {} and my age is {} years".format(name, age))
```

Output: my name is Skanda and my age is 18 years

Input / output operations (functions) in python

Example)

```
name="Skanda“
age=18
cgpa=7.6

print ("my name is {0} my age is {1} and cgpa is {2}".format(name,age,cgpa))
```

Output : my name is Skanda my age is 18 and cgpa is 7.6

Here, the index 0 corresponds to the variable represented by name. Likewise the indices 1 and 2 for age and cgpa

Activity for student: Learn remaining print formats from text books / reference books / Online Resources

Indentation in Python

Indentation in Python

In Python, Indentation is used to define blocks of code. It tells the Python interpreter that a group of statements belongs to a specific block. All statements with the same level of indentation are considered part of the same block. Indentation is achieved using whitespace (spaces or tabs) at the beginning of each line. The most common convention is to use 4 spaces or a tab, per level of indentation.

Comments in Python

this is single comment line

"""“ This is a comment
written in
more than just one line """“

Note : A comment can also begin and end with three single quotes

Expressions :

Expressions :

An expression is a combination of operands (variable / value / constant) and operators.

Operators are symbols that represent particular actions (such as addition, subtraction, comparison etc.,)

Any expression evaluates to a single value, which becomes the value of the expression.

Examples :

a+b # arithmetic expression

a>b # relational expression or condition which results True or False

a>b and a>c # logical expression or condition which also results True or False

Conditionals / Control statements:

They are divided into

- Decision Making Statements / Branching Statements / Conditional Statements
- Iterative / Looping Statements

Conditional statements in Python are used to execute certain blocks of code based on specific conditions.

- if statement
- if … else statement
- if … elif … else statement
- match

For syntax and description students can refer word document / text books / reference books / Online Resources

match – case statement in python

match statement:

A Python match-case statement takes an expression and compares its value to successi given as one or more case blocks.

It allows matching values against patterns and executing code based on the first matching pattern.

Note : With the release of Python 3.10, a pattern matching technique called match-case has been introduced, which is similar to the switch-case construct available in C/C++/Java etc.

```
match expression:
    case pattern1:
        # block of code
    case pattern2:
        # block of code
    case _:
        # default block (like "else")
```

expression → the expression/value being tested.

case → defines a pattern to match against the expression.

_ (underscore) → wildcard pattern, matches anything (default case).

match – case statement in python

This is how it works:

- The match expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Default Value :

The underscore _ is a special wildcard pattern that matches anything. It's often used as the final case to catch all unmatched cases, similar to the else in an if-elif statement.

Combined Cases in Match Statement :

We can combine multiple patterns / cases with the OR operator represented by "|" symbol.

The code block will execute if the expression matches any of the patterns.

Using "if" in "Case" Clause

Match allows us to include if statement in the case clause for conditional computation of match variable.

Click this word document for additional notes

Microsoft Word Document