Course
Computer Programming
Python

By

Ms. Varpe Kanchan

OUTLINE

- Fundamentals of Python
  - What is Python
  - Why it is called Python
  - Brief Idea about Python

WHAT IS PYTHON?

https://docs.python.org/2/faq/general.html#general-information

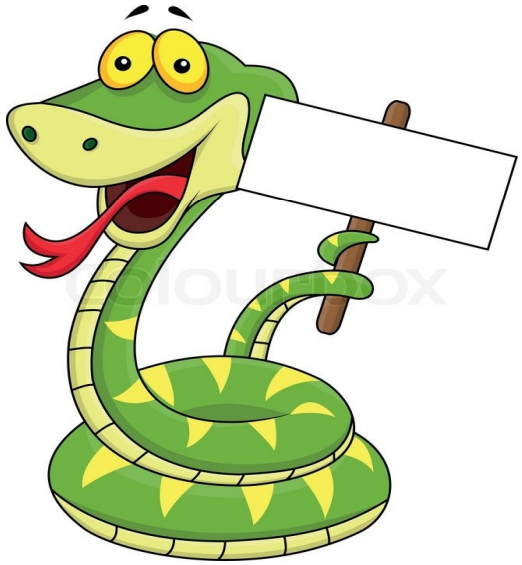
Let's See What is Python?

- Python is an-
  - Interpreted
  - Interactive
  - Object-oriented programming language
- It incorporates-
  - Modules
  - Exceptions
  - Dynamic Typing
  - Very High Level Dynamic Data Types and
  - Classes
- Python combines remarkable power with very clear syntax

LET'S SEE WHAT IS PYTHON? (1)

- **It has interfaces to**
  - Many system calls
  - Libraries
  - To various window systems
- **Extensible in C or C++**
- It is also usable as an extension language for applications that need a programmable interface.
- **Python is portable:**
  - it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.

Why is it called Python?



https://docs.python.org/2/faq/general.html#general-information

- When he began implementing Python, **Guido van Rossum** was also reading the published scripts from **"Monty Python's Flying Circus"**, a BBC comedy series from the 1970s.
- **Van Rossum** thought he needed **a name** that was **short, unique, and slightly mysterious**, so he decided to call the language **Python**.

Brief Idea about Python

https://docs.python.org/3/tutorial/appetite.html

Whetting Your Appetite

- If you do much work on computers, eventually you find that there's some **task you'd like to automate**.
- For example, you may wish
  - to perform a search-and-replace over a large number of text files,
  - or rename and rearrange a bunch of photo files in a complicated way
  - Perhaps you'd like to write a small custom database, or a specialized GUI application,
  - or a simple game

Whetting Your Appetite (1)

- If you're a professional software developer, you may have to work with several C/C++/Java libraries **but find the usual write/compile/test/re-compile cycle is too slow.**
- Perhaps you're writing a test suite for such a library and **find writing the testing code a tedious task.**
- Or maybe you've written **a program that could use an extension language,** and you don't want to design and implement a whole new language for your application.

Whetting Your Appetite (2)

- **Python is just the language for you.**
- You could write a Unix shell script or Windows batch files for some of these tasks, but **shell scripts are best at moving around files** and changing text data, **not well-suited for GUI applications or games.**
- You could write a C/C++/Java program, but it can take **a lot of development time** to get even a first-draft program.
- Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and **will help you get the job done more quickly.**

Whetting Your Appetite (3)

- Python is simple to use,
  - but it is a **real programming language**,
  - offering much more structure and support for large programs
- On the other hand,
  - Python also offers much more error checking than C,
  - being a ***very-high-level language***, it has high-level data types built in, such as **flexible arrays and dictionaries**.
- Because of its more general data types **Python is applicable to a much larger problem domain** than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Whetting Your Appetite (4)

- Python allows you
  - to **split your program into modules**
  - that can be **reused in other Python programs**.
- It comes with **a large collection of standard modules** that you can use as the basis of your programs — or as examples to start learning to program in Python.
- Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Whetting Your Appetite (5)

- Python is an **interpreted language**,
  - which can save your considerable time during program development
  - because no compilation and linking is necessary
- The interpreter can be used interactively, which makes it easy
  - to experiment with features of the language,
  - to write throw-away programs,
  - to test functions during bottom-up program development.

WHETTING YOUR APPETITE (6)

- Python enables programs to be written **compactly and readably.**
- **Programs** written in Python are typically much **shorter** than equivalent C, C++, or Java programs, for several reasons:
  - the **high-level data types** allow you **to express complex operations in a single statement**;
  - **statement grouping** is done **by indentation instead of beginning and ending brackets**;
  - **no variable or argument declarations** are necessary.
- It is also a handy desk calculator.

Whetting Your Appetite (7)

- **Python is *extensible***:
  - if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library).
  - Once you are really hooked, you **can link the Python interpreter into an application written in C** and use it as an extension or command language for that application.

Whetting Your Appetite (8)

- By the way, the language is **named after the BBC show** "Monty Python's Flying Circus" and has nothing to do with reptiles.
- Making references to Monty Python skits in documentation is not only allowed, it is encouraged!
- Now that **you are all excited about Python**, you'll want to examine it in some more detail.
- Since the **best way to learn a language is to use it**

PYTHON FEATURES

https://www.cs.columbia.edu › ~hgs › teaching › slides › python

Python features

no compiling or linking	rapid development cycle
no type declarations	simpler, shorter, more flexible
automatic memory management	garbage collection
high-level data types and operations	fast development
object-oriented programming	code structuring and reuse, C++
embedding and extending in C	mixed language systems
classes, modules, exceptions	"programming-in-the-large" support
dynamic loading of C modules	simplified extensions, smaller binaries
dynamic reloading of C modules	programs can be modified without stopping

Python features

Lutz, *Programming Python*

universal "first-class" object model	fewer restrictions and rules
run-time program construction	handles unforeseen needs, end-user coding
interactive, dynamic nature	incremental development and testing
access to interpreter information	metaprogramming, introspective objects
wide portability	cross-platform programming without ports
compilation to portable byte-code	execution speed, protecting source code
built-in interfaces to external services	system tools, GUIs, persistence, databases, etc.

INTERPRETER

- A program that executes instructions written in a high-level language
- There are two ways to run programs written in a high-level language
  - The most common is to compile the program;
  - the other method is to pass the program through an interpreter.

Interpreter Versus Compiler

- Compiler and Interpreter are **two different ways to execute a program** written in a programming or scripting language.
- A compiler **takes entire program and converts it into object code** which is typically stored in a file.
- The object code is also refereed as binary code and can be directly executed by the machine after linking.
- Examples of compiled programming languages are C and C++.

Interpreter Versus Compiler

- An **Interpreter** directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code.
- Examples of interpreted languages are Perl, Python and Matlab.

INTERESTING FACTS ABOUT INTERPRETERS AND COMPILERS

Following are some interesting facts about interpreters and compilers.

1. Both compilers and interpreters convert source code (text files) into tokens, both may generate a parse tree, and both may generate immediate instructions. The basic difference is that a compiler system, including a (built in or separate) linker, generates a stand alone machine code program, while an interpreter system instead performs the actions described by the high level program.
2. Once a program is compiled, its source code is not useful for running the code. For interpreted programs, the source code is needed to run the program every time.
3. In general, interpreted programs run slower than the compiled programs.
4. Java programs are first compiled to an intermediate form, then interpreted by the interpreter.

Interpreter Versus Compiler

- An interpreter translates high-level instructions into an intermediate form, which it then executes.
- In contrast, a compiler translates high-level instructions directly into machine language.
- Compiled programs generally run faster than interpreted programs.
- The advantage of an interpreter, however, is that it does not need to go through the compilation stage during which machine instructions are generated.
- This process can be time-consuming if the program is long.
- The interpreter, on the other hand, can immediately execute high-level programs.

INTERPRETER (2)

- **Interpreter Versus Compiler**
- For this reason, interpreters are sometimes used during the development of a program, when a **programmer wants to add small sections at a time and test them quickly.**
- In addition, interpreters are often used in education because they allow students to program interactively.

INTERPRETER (1)

- **Interpreter Versus Compiler**
- Both interpreters and compilers are available for most high-level languages.
- However, BASIC and LISP are especially designed to be executed by an interpreter.
- In addition, page description languages, such as PostScript, use an interpreter.
- Every PostScript printer, for example, has a built-in interpreter that executes PostScript instructions.

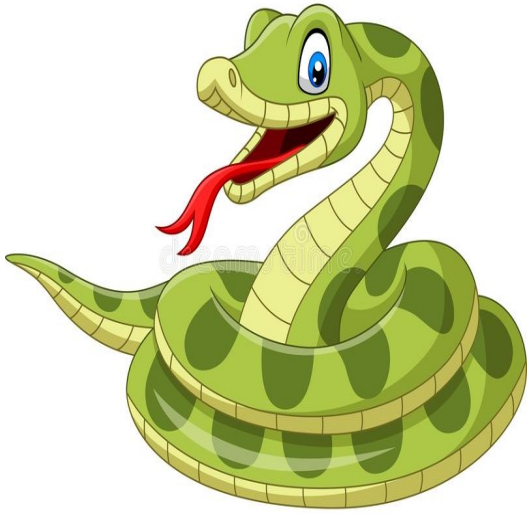
OBJECT-ORIENTED PROGRAMMING (**OOP**)

- Refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure.
- In this way, the data structure becomes an object that includes both data and functions.
- In addition, programmers can create relationships between one object and another.
- For example, objects can inherit characteristics from other objects.

ADVANTAGES OF (OOP)

- One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that'
  - they enable programmers to create modules that do not need to be changed when a new type of object is added.
  - A programmer can simply create a new object that inherits many of its features from existing objects.
  - This makes object-oriented programs easier to modify.

Thank you



Let's play with Python