

## CURSO PYTHON

### OBJETIVO

Que el estudiante aprenda a desarrollar scripts y programas con Python mediante el uso y aplicación n de los tipos de datos, funciones, módulos, librerías y metodologías orientadas a objetos que proporciona el lenguaje.

### TEMARIO

- **About Python**
  - Why should I use Python?
  - What Python does well
  - What Python doesn't do as well
- **Getting started**
  - Installing Python
  - IDLE and the basic interactive mode
  - Using IDLE's Python Shell window
  - Hello, world
  - Using the interactive prompt to explore Python
- **The Quick Python overview**
  - Python synopsis
  - Built-in data types
  - Control flow structures
  - Module creation
  - Object-oriented programming
- **The essentials**
  - The absolute basics
  - Indentation and block structuring
  - Differentiating comments
  - Variables and assignments
  - Expressions
  - Strings
  - Numbers
  - The None value
  - Getting input from the user
  - Built-in operators
  - Basic Python style
- **Lists, tuples, and sets**
  - Lists are like arrays
  - List indices
  - Modifying lists
  - Sorting lists
  - Other common list operations
  - Nested lists and deep copies



- Tuples
- Sets
- **Strings**
  - Strings as sequences of characters
  - Basic string operations
  - Special characters and escape sequences
  - Converting from objects to strings
  - Using the format method
  - Formatting strings with %
  - Bytes
- **Dictionaries**
  - What is a dictionary?
  - Other dictionary operations
  - Word counting
  - What can be used as a key?
  - Sparse matrices
  - Dictionaries as caches
  - Efficiency of dictionaries
- **Control flow**
  - The while loop
  - The if-elif-else statement
  - The for loop
  - List and dictionary comprehensions
  - Statements, blocks, and indentation
  - Boolean values and expressions
  - Writing a simple program to analyze a text file
- **Functions**
  - Basic function definitions
  - Function parameter options
  - Mutable objects as arguments
  - Local, nonlocal, and global variables
  - Assigning functions to variables
  - lambda expressions
  - Generator functions
  - Decorators
- **Modules and scoping rules**
  - What is a module?
  - A first module
  - The import statement
  - The module search path
  - Private names in modules
  - Library and third-party modules
  - Python scoping rules and namespaces
- **Python programs**
  - Creating a very basic program
  - Making a script directly executable on UNIX
  - Scripts on Mac OS X
  - Script execution options in Windows
  - Scripts on Windows vs. scripts on UNIX
  - Programs and modules
  - Distributing Python applications
- **Using the filesystem**
  - Paths and pathnames
  - Getting information about files
  - More filesystem operations
  - Processing all files in a directory subtree
- **Reading and writing files**
  - Opening files and file objects
  - Closing files
  - Opening files in write or other modes
  - Functions to read and write text or binary data
  - Screen input/output and redirection
  - Reading structured binary data with the struct module
  - Pickling objects into files
  - Shelving objects



- **Exceptions**
  - Introduction to exceptions
  - Exceptions in Python
  - Using with
- **Classes and object-oriented programming**
  - Defining classes
  - Instance variables
  - Methods
    - Class variables
  - Static methods and class methods
  - Inheritance
  - Inheritance with class and instance variables
  - Private variables and private methods
  - Using @property for more flexible instance variables
  - Scoping rules and namespaces for class instances
  - Destructors and memory management
  - Multiple inheritance
- **Graphical user interfaces**
  - Installing Tkinter
  - Starting Tk and using Tkinter
  - Principles of Tkinter
  - A simple Tkinter application
  - Creating widgets
  - Widget placement
  - Using classes to manage Tkinter applications
  - What else can Tkinter do?
  - Alternatives to Tkinter
- **Advanced language features**
  - Regular expressions
  - What is a regular expression?
  - Regular expressions with special characters
  - Regular expressions and raw strings
  - Extracting matched text from strings
  - Substituting text with regular expressions
- **Packages**
  - What is a package?
  - A first example
  - A concrete example
  - The \_\_all\_\_ attribute
  - Proper use of packages
- **Data types as objects**
  - Types are objects, too
  - Using types
  - Types and user-defined classes
  - Duck typing
- **Advanced object-oriented features**
  - What is a special method attribute?
  - Making an object behave like a list
  - Giving an object full list capability
  - Subclassing from built-in types
  - When to use special method attributes
  - Metaclasses
  - Abstract base classes
- **Where can you go from here?**
  - Testing your code made easy(-er)
  - Why you need to have tests
  - The assert statement
  - Tests in docstrings: doctests
  - Using unit tests to test everything, every time
- **Moving from Python 2 to Python 3**
  - Porting from 2 to 3
  - Testing with Python 2.6 and -3
  - Using 2to3 to convert the code
  - Testing and common problems
  - Using the same code for 2 and 3



- **Using Python libraries**
  - “Batteries included”—the standard library
  - Moving beyond the standard library
  - Adding more Python libraries
  - Installing Python libraries using setup.py
  - PyPI, a.k.a. “the Cheese Shop”
- **Network, web, and database programming**
  - Accessing databases in Python
  - Network programming in Python
  - Creating a Python web application
  - Sample project—creating a message wall

Duración: 50 horas

Documentación: The Quick Python Book, Diploma expedido por PLCT

