Python

PYTHN200: Practical Introduction to Python Development

$3,195.00

- 5 Days
- Learn Python at Interface!
- Developer instructor Judy Lipinski
-

Upcoming Dates

Course Description

Python is a very popular open-source, high-level programming language used for a broad assortment of programming tasks. This course is designed to introduce best practices in Python development, and to build a strong foundation for students to go on to use Python in creating one-off utilities, processing data, creating automated test scripts, machine learning, and web development.

A balanced mixture of theory and labs demonstrate practical usage of Python and are designed to give students real-world understanding and experience.

Course Outline

0. Introductions

a. Course Objectives and overview of what teams will be building
b. Course Logistics
c. Getting Course Demos & Solutions using Git

1. Overview of Python

a. What is Python?
b. Where Python is used: machine learning , web , scripting
c. Python history
d. Python 2 versus 3

2. Setting up environment and leveraging tools

a. Development environment Setup
b. Installing python on different Operating Systems (OS)
c. Using the Python Interpreter
d. Scripting versus interactive mode
e. Jupyter Notebook
f. Python editors and IDEs
g. Using VSCode for python
h. Getting help with pydoc

3. Python basics in Interactive mode and simple scripts

a. Interactive mode versus scripts
b. Creating variables and working with built≠in types
c. Naming conventions & keywords
d. Representing Strings and working with operators & methods
e. Numeric types and math operations
f. Converting types
g. Using print() to display to the screen
h. Executing scripts
i. Script comments

4. Conditional Programming & Flow control

a. Using indentations for defining blocks
b. When to use if, elif and else
c. Boolean operators
d. Relational operators
e. Creating and controlling loops

5. Debugging in Python

a. What is debugging and why do we use it?
b. Using the pdb Module
c. Tracing
d. Stepping Through Code
e. Seeing Where You Are with l
f. Printing Variables
g. Finding a Bug
h. Working interactively

6. Leveraging Functions

a. Defining functions
b. Attribute references and instantiation
c. Default arguments
d. Keyword arguments
e. Arbitrary argument lists
f. Using argument lists
g. Documentation Strings
h. Function annotations
i. Recursion

7. Data Structures and their practical applications

a. Immutable/Mutable types
b. Sequence Types - list, tuple , range
c. Storing and selecting multiple values in Lists
d. Working with immutable Tuples
e. Optimizing loops using range()
f. Storing and working with unique values in Sets
g. Mapping with Dictionaries

8. Errors and Exceptions

a. Syntax Errors
b. Runtime Errors
c. Exceptions
d. Built-in Exceptions
e. Handling Exceptions: try & except
f. Handling Exceptions: finally
g. Raising Exceptions
h. Ignoring Exceptions
i. User-defined Exceptions
j. Clean-up Actions

9. Defining reusable Modules

a. Module definition basics
b. Import Modules
c. Reload Modules
d. Python Standard Library
e. Using packages to structure module namespace
f. Installing 3rd party packages with PIP

10. Input and Output

a. Using the built-in sys module
b. Getting input from the command line
c. Getting input from the keyboard
d. Reading and Writing Files with File Objects

11. Classes & Object-Oriented Programming (OOP)

a. What is OOP?
b. Python Classes and creating objects
c. Scope and Namespaces
d. Defining instance and class attributes
e. Method Objects
f. Inheritance and Polymorphism
g. Multiple Inheritance in Python
h. Overloading operators

12. Exploring the Standard Library modules

a. The os module
b. Logging
c. Math module
d. 're' Regular Expression tools
e. Dates and Times
f. Timeit
g. Email package
h. Json package
i. Multi-threading
j. Data Compression and Archiving

13.Functional Programming

a. Lambda Functions
b. Recursion
c. List Comprehensions
d. Iterators, Generators, Yield
e. Map, reduce, filter

14. Unit Testing Options

a. What is a Unit Test?
b. The unittest framework, nose, and PyTest
c. Asserting
d. Creating Unit Tests
e. Organizing Test Code
f. Discovering Unit Tests
g. Running Unit Tests
h. Skipping Tests

15. Examples of python usage

a. Scripting
b. Raspberry Pi
c. Web, Django and Flask
d. Machine Learning

Audience

Those with experience with basic programming concepts such as loops and variables.

Prerequisites

Students should know how to navigate the Windows operating system. Ideally students already have some understand variables, functions and for loops - but demos and guidance will be given to help them to achieve a working application.

What You Will Learn

- Understand Python's features, tools and what it's used for today
- Know how to set up Python and necessary development environment
- Write and run python programs/scripts
- Learn basic syntax and control flow in Python
- Leverage Python data structures and methods in practical applications
- Organize code with functions, classes, modules and packages
- Use object-oriented programming in Python
- Debug python code and identify bugs
- Explore functional programming techniques
- Read and write files with formatted string output
- Import and explore commonly used standard library modules
- Handle exceptions and errors
- Introduce methods for Unit Testing Python code
- Use PIP to install and manage third-party Python packages and modules