# Advanced Python Programming (TTPS4850)

**Modality: Virtual Classroom**

**Duration: 4 Days**

**SUBSCRIPTION: No**

## About this course:

Geared for experienced Python programmers, ***Advanced Python Programming*** is a practical, hands-on Python training course that thoroughly explores intermediate to advanced level topics and skills, teaching students how to Leverage OS services, Code graphical interfaces for applications, Create modules, Create and run unit tests, Define classes, Interact with network services, Query databases, Process XML data and much more.  This comprehensive, practical course provides an in-depth exploration of working with the programming language, not an academic overview of syntax and grammar.

The average salary of a Python Programmer is **$111,557** per year.

## Course objectives:

Working within in an engaging, hands-on learning environment, guided by our expert team, attendees will learn to:

- Leverage OS services
- Code graphical interfaces for applications
- Create modules
- Create and run unit tests
- Define classes
- Interact with network services
- Query databases
- Process XML data

## Audience:

This course is appropriate for experienced Python programmers. Students should be able to write simple Python scripts, using basic data types, program structures and the standard Python library.

## Prerequisite:

Students should have practical skills equivalent to or should have received training in the following courses or topics as a pre-requisite:

- TTEY101 Introduction to Python Programming (3 days)
- TTPS4810 Essential Python Programming (4 days)

## Prerequisite:

- There are no prerequisites required for this course

## Course Outline:

### Module 1: Python refresher

Data types
Sequences
Mapping types
Program structure
Files and console I/O
Conditionals
Loops
Built-ins

### Module 2: OS services

The OS module
Environment variables
Launching external commands
Walking directory trees
Paths, directories, and filenames
Working with file systems
Dates and times

### Module 3: Pythonic programming

The Zen of Python
Common idioms
Lambda functions
List comprehensions
Generator expressions
String formatting

### Module 4: Modules and packages

Initialization code
Namespaces
Executing modules as scripts
Documentation
Packages and name resolution
Naming conventions
Using imports

### Module 5: Classes

Defining classes
Instance methods and data
Properties
Initializers
Class and static methods/data
Inheritance

**Module 6: Metaprogramming**

Implicit properties
globals() and locals()
Working with attributes
The inspect module
Decorators
Monkey patching

**Module 7: Programmer tools**

Analyzing programs
Using pylint
Testing code
Using unittest
Debugging
Profiling and benchmarking

**Module 8: Distributing modules**

Distribution concepts
setuptools
Creating setup.py
Building installers
Running installers

**Module 9: Database access**

The DB API
Available Interfaces
Connecting to a server
Creating and executing a cursor
Fetching data
Parameterized statements
Metadata
Transaction control
Other DBMS modules

**Module 10: GUI programming with PyQT4**

About QT4

Getting started with the designer
Widget properties
Predefined dialogs
Generating the UI
Wiring up events
Advanced Topics

**Module 11: Network programming**

Sockets
Clients
Servers
Application protocols
Forking servers
Binary data
The struct module

**Module 12: Threads**

When to use threads?
The Global Interpreter Lock
The threading module
Simple threading
Sharing variables
Threaded servers
The queue module
Debugging threaded programs
Alternatives to threading

**Module 13: XML and JSON**

Working with XML
DOM and Sax
Introducing ElementTree and lxml
Parsing XML
Navigating the document
Creating a new XML document
JSON
Parsing JSON into Python
Converting Python into JSON

**Module 14: Extending Python**

About non-Python modules
Overview of a C extension
Writing C by hand
Loading modules with ctypes

## Module 15: Subprocesses

- Running external commands with subprocess
- Getting command status
- Managing STDOUT, STDERR, and STDIN
- The sh module (non-Windows systems only)
- Creating a simple command
- Keyword arguments
- Running commands in the background
- Piping and redirection
- Working with STDIO
- Exit codes
- Advanced features