Data Science, Machine Learnig And Artifcial Intelligence Using Python

Presented by
Mr. Bandenawaz Bagwan

What we will be learning today

- Introduction to Python
- Variables and Data types in Python
- Modules ,Libraries and Packages
- Controlflow statements
- Functions in Python
- Numpy Module
- Pandas Module
- Data Visualization with matplotlib

Python

- Python is a dynamic, interpreted (bytecode-compiled) language that is used in a wide range of domains and technical fields.
- It was developed by Guido van Rossum in 1991.
- It was mainly developed for code readability and its syntax is such that it allows programmers to code/express concepts in fewer lines of code.
- Python is a general-purpose programming language that is becoming more and more popular for
  - performing data analysis,
  - automating tasks,
  - learning data science,
  - machine learning, etc.

Where is Python used?

Python is used by the novice programmer as well as by the highly skilled professional developer.

- Web and Internet development: Python is used on the server side to create web applications.
- Software development: Python is used to create GUI applications, connecting databases, etc.
- Scientific and Numeric applications: Python is used to handle big data and perform complex mathematics.
- Education: Python is a great language for teaching programming, both at the introductory level and in more advanced courses.
- Desktop GUIs: The Tk GUI library2 included with most binary distributions of Python is used extensively to build desktop applications.
- Business Applications: Python is also used to build ERP and ecommerce systems.

Features of Python

- Simple
- Easy to Learn
- Free and Open Source
- High-level
- Dynamically Typed
- Portable/Platform Independent/Cross Platform
- Interpreted
- Extensive Libraries
- Garbage Collection

Variables and Data Types in Python

- A variable can be thought of as a container having a name which is used to store a value.
- **Variable Declaration and Assignment**
  - #Creating a variable
  - price = 226 #sign = a.k.a. Assignment operator. A variable is created the moment we assign the first value to it.
  - print(price)
  - 226 # Output

Variable Naming Conventions

- A variable name must start with a letter or the underscore character.
  - stock = 'AAPL' # Valid name
  - _name = 'AAPL' # Valid name
- A variable name cannot start with a number
  - 1stock = 'AAPL' # Invalid name
  - 1_stock = 'AAPL' # Invalid name

A variable name can only contain alpha-numeric characters(A-Z, a-z, 0-9) and underscores(_).

- #Valid name. It starts with a capital letter.

Stock = 'AAPL’

- #Valid name. It is a combination of alphabets and the underscore.

stock_price = 226.41

Data Types

- Python basic data types:
  - Integer
  - Float
  - String
  - Boolean
  - Lists
  - Tuples
  - Dictionaries

Modules, Packages and Libraries

- Module:
  - module allows us to organize Python code in a systematic manner.
  - It can be considered as a file consisting of Python code.
  - A module can define functions, classes and variables. It can also include runnable code.
  - Python has a way to put a code definition in a file and use them in another script or directly in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or in the program that we code.
- create a file called arithmetic.py in the current directory with the following contents:

```
• def addition(a, b):
 """Returns the sum of of two numbers"""
   return a + b
def multiply(a, b):
"""Returns the product of two numbers"""
    return a * b
def division(dividend, divisor):
"""
Performs the division operation between the dividend
and divisor
"""“
   return dividend / divisor
def factorial(n):
"""Returns the factorial of n"""
   i = 0
   result = 1
  while(i != n):
     i = i + 1
    result = result * i
  return result
```

We are now ready to import this file in other scripts or directly into the Python interpreter

```
import arithmetic
result = arithmetic.addition(2, 3)
print(result)
#Now call the other functions with different values and print the result
```

- A module name is available as a string within a script or the interpreter as the value of the global variable __name__.
  - arithmetic.__name__
  - 'arithmetic’ #output

Packages

- Packages can be considered as a collection of modules.
- It is a way of structuring Python's module namespace by using "dotted module names".
- Here's a possible package structure to make our lives easier.

```
strats/                 Top-level package
    __init__.py             Initialize strats package
    data/               Sub-package for data
        __init__.py
        equity.py           Equity module
        currency.py
        options.py
```

Control Flow Statements

- **Conditional Statements**
  - The if statement
  - The elif clause
  - The else clause

- **Loops**
  - The while statement
  - The for statement
  - The range() function
  - Nested loops

- **Loop control statements**
  - The break keyword
  - The continue keyword
  - The pass keyword

Functions in Python

A function is a block of code(that performs a specific task) which runs only when it is called.

- From the definition, it can be inferred that writing such block of codes, i.e. functions, provides benefits such as
  - Reusability: Code written within a function can be called as and when needed. Hence, the same code can be reused thereby reducing the overall number of lines of code.
  - Modular Approach: Writing a function implicitly follows a modular approach. We can break down the entire problem that we are trying to solve into smaller chunks, and each chunk, in turn, is implemented via a function.

There are three types of functions in Python:

- Built-in functions such as print to print on the standard output device, type to check data type of an object, etc. These are the functions that Python provides to accomplish common tasks.
- User-Defined functions: As the name suggests these are custom functions to help/resolve/achieve a particular task.
- Anonymous functions, also known as lambda functions are custom made without having any name identifier.
- **User defined functions**
  - As Python programmers, we might need to break down the programming challenge into smaller chunks and implement them in the form of custom or user defined functions.
  - Functions are defined using the def keyword, followed by an identifier name along with the parenthesis, and by the final colon that ends the line
  - def greet():
  - """Block of statement.
  - or Body of function.
  - """
  - print(' Hello from inside the function!')

NumPy Module

- NumPy, an acronym for Numerical Python, is a package to perform scientific computing in Python efficiently.
- It includes random number generation capabilities, functions for basic linear algebra, Fourier transforms as well as a tool for integrating Fortran and C/C++ code along with a bunch of other functionalities.
- NumPy is an open-source project and a successor to two earlier scientific Python libraries: Numeric and Numarray.
- It can be used as an efficient multi-dimensional container of generic data. This allows NumPy to integrate with a wide variety of databases seamlessly
- NumPy is not a part of the Python Standard Library and hence, as with any other such library or module, it needs to be installed on a workstation before it can be used.
  - pip install numpy

Pandas Module

- Pandas is a Python library to deal with sequential and tabular data.
- It includes many tools to manage, analyze and manipulate data in a convenient and efficient manner.
- Pandas is built on top of the Numpy library and has two primary data structures viz.
- Series (1-dimensional) and DataFrame (2- dimensional). It can handle both homogeneous and heterogeneous data, and some of its many capabilities are:
  - ETL tools (Extraction, Transformation and Load tools)
  - Dealing with missing data (NaN)
  - Dealing with data files (csv, xls, db, hdf5, etc.)
  - Time-series manipulation tools

Why Pandas

- It includes a multitude of tools to work with these data types, such as:
  - Indexes and labels.
  - Searching of elements.
  - Insertion, deletion and modification of elements.
  - Apply set techniques, such as grouping, joining, selecting, etc.
  - Data processing and cleaning.
  - Work with time series.
  - Make statistical calculations
  - Draw graphics
  - Connectors for multiple data file formats, such as, csv, xlsx, hdf5, etc.

Pandas Series

- The first data structure in Pandas that we are going to see is the Series.
- They are homogeneous one-dimensional objects, that is, all data are of the same type and are implicitly labeled with an index.
- We can have a Series of integers, real numbers, characters, strings, dictionaries, etc.
- We can conveniently manipulate these series performing operations like adding, deleting, ordering, joining, filtering, vectorized operations, statistical analysis, plotting, etc.

Let's see some examples of how to create and manipulate a Pandas Series:

Pandas DataFrame

- The second data structure in Pandas that we are going to see is the DataFrame.
- Pandas DataFrame is a heterogeneous two-dimensional object, that is, the data are of the same type within each column but it could be a different data type for each column and are implicitly or explicitly labeled with an index.
- We can think of a DataFrame as a database table, in which we store heterogeneous data.
- For example, a DataFrame with one column for the first name, another for the last name and a third column for the phone number, or a dataframe with columns to store the opening price, close price, high, low, volume, and so on.
- The index can be implicit, starting with zero or we can specify it ourselves, even working with dates and times as indexes as well. Let’s see some examples of how to create and manipulate a Pandas DataFrame.

Data Visualization with Matplotlib

- Matplotlib is a popular Python library that can be used to create data visualizations quite easily.
- It is probably the single most used Python package for 2D-graphics along with limited support for 3D-graphics.
- It provides both, a very quick way to visualize data from Python and publication-quality figures in many formats.
- It was designed from the beginning to serve two purposes:
  - Allow for interactive, cross-platform control of figures and plots
  - Make it easy to produce static vector graphics files without the need for any GUIs.

Thank You !!!

Presented by
Mr. Bandenawaz Bagwan