BRIEF INTRODUCTION TO PYTHON

Marjan Mehrdadi

TABLE OF CONTENTS

NOTE

this presentation is provided to organize the subjects, and help for a better understanding of the code sample while containing a more extended description for python basics that may be missed or overlooked on the sample.

INTRODUCTION

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

REFERENCES

There is a wide range of books, websites and courses available that may help your journey of getting started with this language.

Python is known to be a fairly simple, beginner friendly language, which makes it specially easy for people already familiar with coding.

Main references used for this brief introduction are:

1. www.w3schools.com
2. Machine leaning in the oil and gas industry

CODE EDITORS

If you want to learn to program in Python, you'll need a code editor or an IDE.

An integrated development environment (IDE) is a software application that helps programmers to develop software efficiently. It increases developer productivity by combining common developer tools such as software editing, building, testing, debugging, and packaging in one easy-to-use graphical user interface (GUI). Other popular features include code refactoring, code search, code auto-completion, and continuous integration/continuous deployment (CI/CD).

An IDE can also provide many more features and these generally vary with each IDE.

While code editors are similar to text editors, they are designed to both ease and speed up code development via sophisticated built-in capabilities and functionalities. Sometimes code editors can be mistaken for IDEs, but the main difference between the two is that IDEs provide more powerful tools to simplify the coding process.

CODE EDITORS

Some common code editor/IDEs for python may include:

1. PyCharm
2. PyDev
3. IDLE
4. Visual studio code
5. Jupyter notebook

Besides looking into advantages and disadvantages of each editor, choosing a suitable editor is mainly based on the preference of developer, and their project's needs.

Although my own preferred IDE is Jupyter notebook, some problems are encountered while trying to access this editor in Iran, thus PyChram is used as second best option for me personally.

SYNTAX

The Python syntax defines a set of rules that are used to create Python statements while writing a Python Program. The Python Programming Language Syntax has many similarities to Perl, C, and Java Programming Languages. However, there are some definite differences between the languages.

Python is meant to be an easily readable language. Its formatting is visually uncluttered and often uses English keywords where other languages use punctuation.

VARIABLES

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

•A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables: A variable name must start with a letter or the underscore character

•A variable name cannot start with a number

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

•Variable names are case-sensitive (age, Age and AGE are three different variables)

•A variable name cannot be any of the Python keywords.

Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.
Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.
To create a global variable inside a function, you can use the global keyword.

DATA STRUCTURES

Lists

Are just like dynamic sized arrays, declared in other languages (vector in C++ and ArrayList in Java). Lists need not be homogeneous always which makes it the most powerful tool in Python.

Tuple

A Tuple is a collection of Python objects separated by commas. In some ways, a tuple is similar to a list in terms of indexing, nested objects, and repetition but a tuple is immutable, unlike lists that are mutable.

Set

A Set is an unordered collection data type that is iterable, mutable, and has no duplicate elements. Python's set class represents the mathematical notion of a set.

Dictionary

in Python is an ordered (since Py 3.7) [unordered (Py 3.6 & prior)] collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized.

LIST METHODS

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

DATA TYPES

example	Data type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

ARITHMETIC OPERATORS

operator	name	example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

ASSIGNMENT OPERATORS

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

COMPARISON OPERATORS

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

MEMBERSHIP OPERATORS

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

BITWISE OPERATORS

Operator		Description	
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2

OTHER OPERATORS

Logical Operators	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)
Identity Operators		
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

PYTHON CLASS-OBJECTS

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

The __init__() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

For more information see:

1. https://www.w3schools.com/python/python_classes.asp
2. https://www.w3schools.com/python/python_inheritance.asp

PYTHON LIBRARIES

Scikit-learn is one of the most popular open source machine learning libraries in Python, which is built on other open source Python libraries, including NumPy, SciPy, and Matplotlib. Scikit-learn was originally developed in 2007 as a part of the Google Summer of Code project. It provides the implementation of a wide range of supervised and unsupervised learning algorithms with a Python programming interface.

PYTHON LIBRARIES

TensorFlow is another open source software library developed by Google for numerical computations, which is highly popular for machine learning applications, such as shallow artificial neural networks and deep learning. TensorFlow allows the creation of dataflow graphs using tensors, which are multidimensional arrays through which computation occurs. The library is built to support parallel runs on multiple CPUs and GPUs and provides wrappers for many programming languages, such as Python, C++, and Java.

PYTHON LIBRARIES

Keras, which means horn in Greek, is an open source high-level neural network library, which is very user friendly, modular, and easy to work with Python. It does not handle low-level computation the way TensorFlow does but uses a backend engine to perform computations for the development of models. Keras uses the TensorFlow backend by default. There are many other open source libraries, such as Theano, PyTorch, OpenCV, and Apache Spark ML, available for developing machine learning models.

NOTE

Other discussed features in code sample:

1. Assign multiple variables
2. Commenting
3. Access list items
4. Random
5. If…else
6. While loops
7. For loops
8. Functions
9. Range
10. Iterator
11. Dates
12. Math module
13. Try…except

Also see:
https://www.w3schools.com/python/python_string_formatting.asp
https://www.w3schools.com/python/python_classes.asp
https://www.w3schools.com/python/python_inheritance.asp
https://www.w3schools.com/python/python_datetime.asp
And the mentioned book “Machine leaning in the oil and gas industry ” second and third chapters

THANK YOU

Marjan Mehrdadi
marjanmehrdadit@gmail.com