
Paola Celio | Pietro Corsi | Sergio Lins
PYTHON
BASICS

Lessons November

Date	Time	
November 2	16:00 – 18:00	Computer basic knowledge
November 5	18:00 – 20:00	Computer basic knowledge
November 9	16:00 - 19:00	Programming language introduction
November 12	18:00 – 20:00	Programming language introduction
November 16	16:00 - 19:00	
November 19	18:00 – 20:00	
November 23	16:00 - 19:00	
November 26	18:00 – 20:00	
November 30	16:00 –19:00	

Lessons December

Date	Time
December 4	18:00 – 20:00
December 7	16:00 - 19:00
December 11	18:00 – 20:00
December 14	16:00 - 19:00
December 18	18:00 – 20:00
December 21	16:00 - 19:00

Lesson 3 – Programming Language Introduction

This lesson will be the introduction to programming part so we'll check the Python environment that you've to use and we'll give basic ideas of programming language.

Code (programming) is how you get a computer to do what you want it to do. Laptops, smartphones, video-game consoles, and even washing machines use code to work. And despite advancements in machine learning and AI, somebody has to write that code.

Coders (also known as programmers) are the people who write code. But what is code, really? What exactly are programmers writing when they create software or fine-tune hardware?

Lesson 3 – Programming Language Introduction

Code is made up of *algorithms*: highly structured and ordered lists of instructions, much like a recipe. Because computers aren't actually smart, these instructions are written in *code* that must be parsed and compiled and transformed into *machine language*. Only then can computers perform the actions that programmers dictate. The key is to learn the syntax of the code so you can write words in the required order so that computers can decipher and process them.

Machine language, or machine code, is a low-level language comprised of binary digits (ones and zeros). High-level languages, such as Swift and C++ must be compiled into machine language before the code is run on a computer.

Lesson 3 – Programming Language Introduction

Python is an “interpreted” language.

This means it uses an interpreter. An interpreter is very different from the compiler.

An interpreter executes the statements of code “one-by-one” whereas the compiler executes the code entirely and lists all possible errors at a time.

That’s why python shows only one error message even though your code has multiple errors. This will help you to clear errors easily and it definitely will increase the execution speed.

Lesson 3 – Programming Language Introduction

Since computers are digital devices, they only recognize binary data. Every program, video, image, and character of text is represented in binary. This binary data, or machine code, is processed as input by the CPU. The resulting output is sent to the operating system or an application, which displays the data visually. For example, the ASCII value for the letter "A" is 01000001 in machine code, but this data is displayed as "A" on the screen. An image may have thousands or even millions of binary values that determine the color of each pixel.

That's why we need and interpreter or compiler.

Lesson 3 – Programming Language Introduction

Some people love learning new programming languages. Other people can't imagine having to learn even one. We're going to show you how to think like a coder so that you can confidently learn any programming language you want.

The truth is, once you've learned how to program, the language you use becomes less of a hurdle and more of a formality. Regardless of how simple their introductory language may be, the logic remains the same across everything else children (or adult learners) are likely to encounter later.

Lesson 3 – Programming Language Introduction

With just a little programming experience, which you can gain from any one of several introductory articles on Opensource.com (or other sources), you can go on to learn *any* programming language in just a few days (sometimes less). Now, this isn't magic, and you do have to put some effort into it. And admittedly, it takes a lot longer than just a few days to learn every library available to a language or to learn the nuances of packaging your code for delivery. But getting started is easier than you might think, and the rest comes naturally with practice.

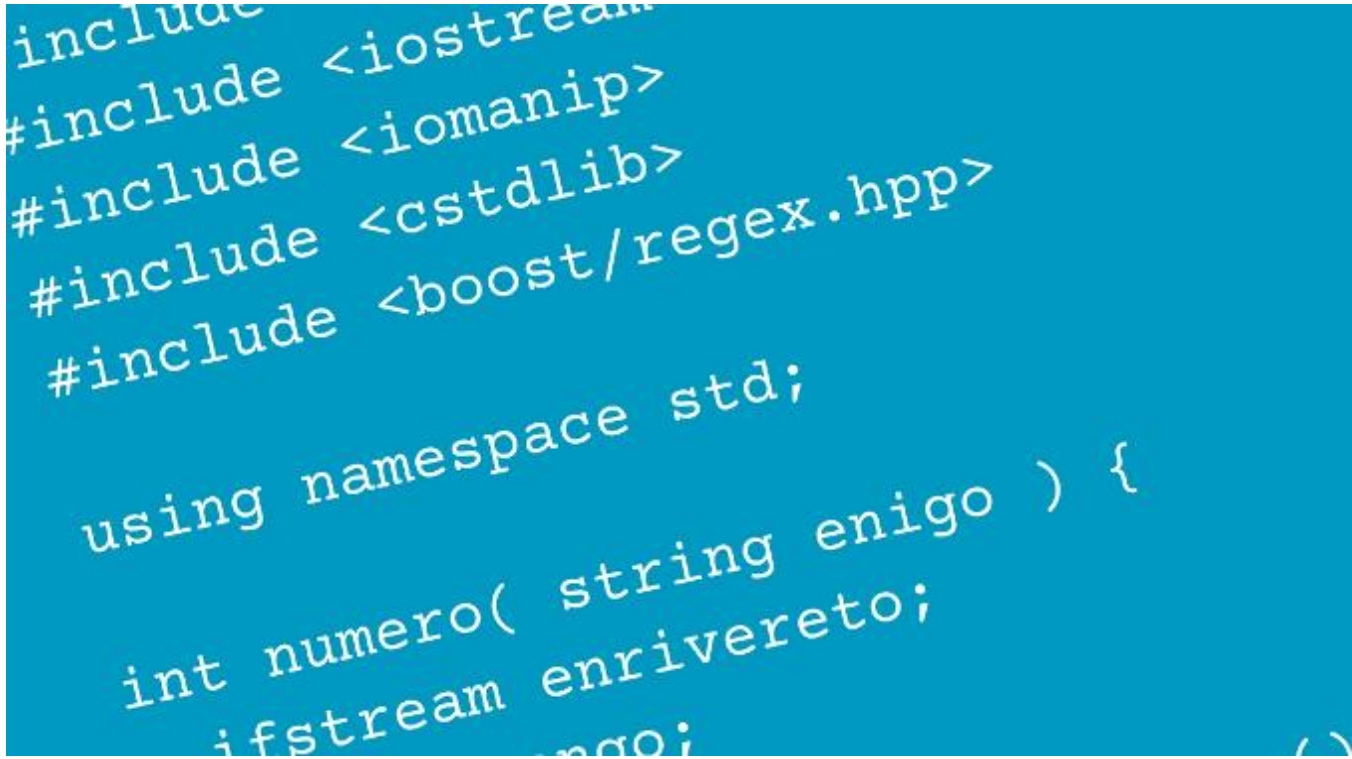
Check this link before proceding.

When experienced programmers sit down to learn a new language, they're looking for five things. Once you know those five things, you're ready to start coding.

Lesson 3 – Programming Language Introduction

Here you can find 5 of most important things that you have to know to learn a programming languange:

1. Syntax
2. Built-in and Conditionals
3. Data Types
4. Operators and parsers
5. Functions



Lesson 3 – Programming Language Introduction

Syntax:

The syntax of a language describes the structure of code. This encompasses both how the code is written on a line-by-line basis as well as the actual words used to construct code statements.

Python, for instance, is known for using indentation to indicate where one block ends and another one starts:

```
while j < rows:
    while k < columns:
        tile = Tile(k * w)
        board.add(tile)
        k += 1
    j += 1
    k = 0
```

You can also read this code like while (something happens) increase j by one and put k egual 0. Something happens is: while k is lesser than columns this something (2 lines with tile word) and increment k by 1.

Lesson 3 – Programming Language Introduction

Built-in and Conditionals:

A programming language, just like a natural language, has a finite number of words it recognizes as valid. This vocabulary can be expanded with additional libraries, but the core language knows a specific set of keywords. Most languages don't have as many keywords as you probably think. Even in a very low-level language like C, there are only 32 words, such as for, do, while, int, float, char, break, and so on.

Knowing these keywords gives you the ability to write basic expressions, the building blocks of a program. Many of the built-in words help you construct conditional statements, which influence the flow of your program. For instance, if you want to write a program that lets you click and drag an icon, then your code must detect when the user's mouse cursor is positioned over an icon. The code that causes the mouse to grab the icon must execute only *if* the mouse cursor is within the same coordinates as the icon's outer edges. That's a classic if/then statement, but different languages can express that differently.

Lesson 3 – Programming Language Introduction

Built-in and Conditionals:

Python uses a combination of if, elif, and else but doesn't explicitly close the statement:

```
if var == 1:
   # action
elif var == 2:
       # some action
    else:
       # some other action
```

While there are small variations in word choice and syntax, the basics are always the same. Learn the ways to define conditions in the programming language you're learning, including if/then, do...while, and case statements.

While there are small variations in word choice and syntax, the basics are always the same. Learn the ways to define conditions in the programming language you're learning, including if/then, do...while, and case statements.

Lesson 3 – Programming Language Introduction

Data Types:

Code deals with data, so you must learn how a programming language recognizes different kinds of data. All languages understand integers and most understand decimals and individual characters (a, b, c, and so on). These are often denoted as int, float and double, and char, but of course, the language's built-in vocabulary informs you of how to refer to these entities.

Sometimes a language has extra data types built into it, and other times complex data types are enabled with libraries. For instance, Python recognizes a string of characters with the keyword str, but C code must include the string.h header file for string features.

Libraries can unlock all manner of data types for your code, but learning the basic ones included with a language is a sensible starting point.

Lesson 3 – Programming Language Introduction

Operators and parsers:

- Once you understand the types of data a programming language deals in, you can learn how to analyze that data. Luckily, the discipline of mathematics is pretty stable, so math operators are often the same (or at least very similar) across many languages. For instance, adding two integers is usually done with a + symbol, and testing whether one integer is greater than another is usually done with the > symbol. Testing for equality is usually done with == (yes, that's two equal symbols, because a single equal symbol is usually reserved to *set* a value).
- There are notable exceptions to the obvious in languages like Lisp and Bash, but as with everything else, it's just a matter of mental transliteration. Once you know *how* the expression is different, it's trivial for you to adapt. A quick review of a language's math operators is usually enough to get a feel for how math is done.
- You also need to know how to compare and operate on non-numerical data, such as characters and strings. These are often done with a language's core libraries. For instance, Python features the split() method to parse different part of your variable.

Lesson 3 – Programming Language Introduction

Functions:

- Code usually isn't just a to-do list for a computer. Typically when you write code, you're looking to present a computer with a set of theoretical conditions and a set of instructions for actions that must be taken when each condition is met. While flow control with conditional statements and math and logic operators can do a lot, code is a lot more efficient once functions and classes are introduced because they let you define subroutines. For instance, should an application require a confirmation dialogue box very frequently, it's a lot easier to write that box *once* as an instance of a class rather than re-writing it each time you need it to appear throughout your code.

- You need to learn how classes and functions are defined in the programming language you're learning. More precisely, first, you need to learn whether classes and functions are available in the programming language. Most modern languages do support functions, but classes are specialized constructs common to object-oriented languages.

Lesson 3 – Programming Language Introduction

Installing Python

Before learning Python, you may need to install it.

Linux: If you use Linux, Python is already included, but make sure that you have Python 3 specifically. To check which version is installed, open a terminal window and type:

python --version

Should that reveal that you have version 2 installed, or no version at all, try specifying Python 3 instead:

python3 --version

Lesson 3 – Programming Language Introduction

Installing Python

If that command is not found, then you must install Python 3 from your package manager or software center. Which package manager your Linux distribution uses depends on the distribution. The most common are **dnf** on Fedora and **apt** on Ubuntu. For instance, on Fedora, you type this:

sudo dnf install python3

MacOS: If you're on a Mac, follow the instructions for Linux to see if you have Python 3 installed. MacOS does not have a built-in package manager, so if Python 3 is not found, install it from python.org/downloads/mac-osx. Although your version of macOS may already have Python 2 installed, you should learn Python 3.

Windows: Microsoft Windows doesn't currently ship with Python. Install it from python.org/downloads/windows. Be sure to select **Add Python to PATH** in the install wizard. Read my article How to Install Python on Windows for instructions specific to Microsoft Windows.

Lesson 3 – Programming Language Introduction

Using an IDE

Now you need to write your code there are a lot of programs that help you in cheching syntax or giving hints one of most famous is Anaconda. But you can write your program also with a simple text editor your favourite taking into account that you've to save the file with the extension .py

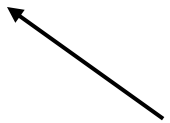
We're going to check how we can write a file on our computer and also make some easy exercise to check filesystem and write output on file.

All these steps will be useful to be more friendly with our computer.

Lesson 3 – Programming Language Introduction

Python from command line

```
(base) Paolas-iMac-2:~ paolacelio$ python
Python 2.7.14 (default, Mar  9 2018, 23:57:12)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```



Wrong version the 2.7 is no more supported but is the default python on a lots of Operative System

Lesson 3 – Programming Language Introduction

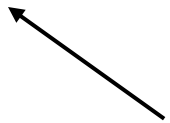
Python from command line

(base) Paolas-iMac-2:~ paolacelio$ python3

Python 3.7.7 (default, Mar 10 2020, 15:43:03)

[Clang 11.0.0 (clang-1100.0.33.17)] on darwin

Type "help", "copyright", "credits" or "license" for more information.



If Python version is atleast 3.3 that's correct atm the last version is 3.8.

Lesson 3 – Programming Language Introduction

Python from command line

```
>>> license()
A. HISTORY OF THE SOFTWARE
==========================


[…]
All Python releases are Open Source (see http://www.opensource.org for
the Open Source Definition).  Historically, most, but not all, Python
Hit Return for more, or q (and Return) to quit: q
```

Lesson 3 – Programming Language Introduction

Python from command line

>>> help()

Welcome to Python 3.7's help utility!

If this is your first time using Python, you should definitely check out the tutorial on the Internet at https://docs.python.org/3.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".

Lesson 3 – Programming Language Introduction

Python from command line

help>

You are now leaving help and returning to the Python interpreter.

If you want to ask for help on a particular object directly from the

interpreter, you can type "help(object)". Executing "help('string')"

has the same effect as typing a particular string at the help> prompt.

>>> quit()

(base) Paolas-iMac-2:~ paolacelio$

Lesson 3 – Programming Language Introduction

https://opensource.com/article/20/9/scratch

https://techterms.com/definition/machine_language

Laboratorio di gestione dati di Cristian Stanescu

https://opensource.com/article/20/10/learn-any-programming-language?utm_medium=Email&utm_campaign=weekly&sc_cid=7013a00000267rgAAA

http://savedev.blogspot.com/2014/10/tecniche-di-programmazione-procedurale.html

Python installation: https://opensource.com/article/17/10/python-101