1. Why Python?: Similar but different

**Python starts counting from zero, which should sound familiar.**

In fact, Python has a lot in **common** with other programming languages. There are **variables**, **loops**, **conditionals**, **functions**, and the like. In this, our opening chapter, we take you on a **high-level whistle-stop tour** of Python’s basics, introducing the language without getting too much into the weeds. You’ll learn how to **create** and **run** code with Jupyter Notebook (running inside VS Code). You’ll see how lots of programming functionality comes **built-in** to Python, which you’ll **leverage** to get stuff done. You’ll also learn that although Python shares a lot of its ideas with other programming languages, how they manifest in your Python code can be, well, **different**. Now, don’t get the wrong idea here: we’re talking different **good**, not different *bad*. Read on to learn more…

Najvažnije:

1. Python je čitljiviji jer ne zahtijeva točke sa zarezom i manje koristi zagrade. Bez točke sa zarezom ili nepotrebnih zagrada. Jednostavno i čitljivo.

# Python kod

x = 5

if x > 0:

print("Broj je pozitivan")

1. Indentacija s praznim prostorom signalizira blokove koda, što čini kod dosljednim i čitljivim.
2. Python ima bogatu standardnu biblioteku i PyPI, online repozitorij trećih strana Python modula. PyPI omogućuje instaliranje dodatnih modula koje su stvorili drugi programeri.

# Instalacija paketa

pip install requests

# Korištenje paketa

import requests

response = requests.get('https://example.com')

print(response.text)

1. Python naglašava ponovno korištenje postojećeg koda, a ne stalno pisanje novog.
2. Pythona odlikuje jednostavnost i svestranost, koristeći ugrađene funkcije poput len.

lista = [1, 2, 3, 4]

duljina = len(lista)

print(duljina)

1. Python je popularan zbog čitljivosti, bogate standardne biblioteke i zajednice. Ugrađena knjižnica math omogućuje rad s matematičkim funkcijama bez dodatnih instalacija.

import math

print(math.sqrt(16))

1. Ime je dobio po Monty Python's Flying Circus, a ne po zmiji. <https://docs.python.org/3/faq/general.html#why-is-it-called-python>
2. what a REPL is, look here: <https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop>
3. **BIF” is shorthand for “built-in function.** functions (BIF): type and len
4. **“PSL” is short-hand for the “Python Standard Library**, **The PSL represents a large body of tested code that you don’t have to write, just use.** <https://docs.python.org/3/library/index.html> A paper with text and arrows

   Description automatically generated
5. Known as *PyPI* (and pronounced “pie-pea-eye”) Python Package Index and lives here: <https://pypi.org>.

**Bullet Points**

* Python is, out of the gate, designed to support the creation of code that is easy to **read**.
* Python code is also easy to **run**. Although a number of ways exist to allow you to do this, in this book **VS Code** together with **Jupyter Notebook** are your go-to tools when experimenting and running your Python code.
* To get going and be productive with Jupyter Notebook, you need to learn a single keyboard combination: **Shift+Enter**.
* In order to ensure you only ever write new code when absolutely necessary, Python comes chock-full of **built-in** technology.
* The built-in functions (**BIFs**) are always available, and provide a lots of **generic** functionality.
* The **len** BIF reports the size of an object.
* The **def** keyword is used to define a **function**.
* The **range** BIF produces a specified list of numbers (and is really useful with loops when you need to iterate a specific number of times).
* Talking of loops, Python provides the **for** loop, which iterates a specific number of times.
* The **set** BIF creates a set. Sets are one of Python’s Big 4 built-in data structures.
* The **print** BIF displays an object’s value on screen. When a collection of objects are printed, the **print** BIF displays horizontally across the screen (which often comes in handy).
* The **dir** BIF returns a list of any object’s attributes.
* A common idiom is to combine the **dir** BIF with the **print** BIF creating (what we like to refer to as) the **print dir combo mambo**.
* Some of the attributes shown by the combo mambo refer to **methods** that can be applied to the object, for example deck.remove.
* Attributes with leading and trailing double underscores are *special*, so special in fact that you can ignore them for now.
* A **list** is made up from a collection of objects surrounded by square brackets, and is one of Python’s built-in Big 4 data structures.
* A **tuple** is made up from a collection of object’s surrounded by parentheses, and is another of the Big 4.
* The final built-in data structure is the **dictionary**, which wasn’t used in this chapter (only mentioned). This doesn’t mean dictionaries aren’t cool. They are.
* The **type** BIF can report any object’s type.
* The **in** keyword was shown in two places in this chapter. Once within a **for** loop where it identified the collection to be iterated over, and again on its own when it was used to determine if one object is contained within another (aka *search*).
* The **in** keyword is often used within the conditional part of Python’s **if** statement.
* When you need a variable but either can’t think of a decent name for your variable or don’t need to remember a value by name, use Python’s **default variable**: a single underscore character (i.e., \_). You’ll often see the default variable used with loop code.
* Python’s **if** statement can have an optional **else** part.
* Python has two built-in **boolean** values: True and False.
* The **PSL** has nothing to do with coffee, but everything to do with the **Python Standard Library**. The PSL is a large collection of built-in modules (which come with Python) and can be used all over the place to do many useful things.
* If the PSL isn’t enough for you, check out **PyPI**, the **Python Package Index**, an online repository of shareable Python modules. It’s often the case some of the code you need has already been written and uploaded to PyPI as a shareable module. Feel free to “leverage” as needed.
* There are other useful **keyboard shortcuts** that you can use when working within Jupyter. Our nine essential shortcuts are coming up after this chapter’s crossword solution (in three pages’ time).

A notebook key sequence with scissors and writing

Description automatically generated with medium confidence

A crossword puzzle with many squares

Description automatically generated

**Across**

1. A built-in function that tells you what something is.

3. Objects surrounded by [ and ].

4. Shorthand for built-in function.

7. Reports on an object’s size.

8. Generates a collection of numbers.

9. The Python Package Index.

11. Includes a module in your code.

12. Objects surrounded by { and }.

14. This chapter’s missing Big 4.

15. This operator can find things.

16. Use together with Shift to run.

17. Enlightenment, Python-style.

**Down**

1. Objects surrounded by ( and ), and it is one of the Big 4, too.

2. The optional part of an if statement.

5. Loops a specific number of times.

6. A character used as Python’s default variable.

9. It’s not a *Pumpkin Spice Latte*, but shares the same acronym.

10. Displays to screen.

12. A bunch of characters.

13. It’s either True or False.

14. Part of the *combo mambo*.

A crossword puzzle with two people in the background

Description automatically generated