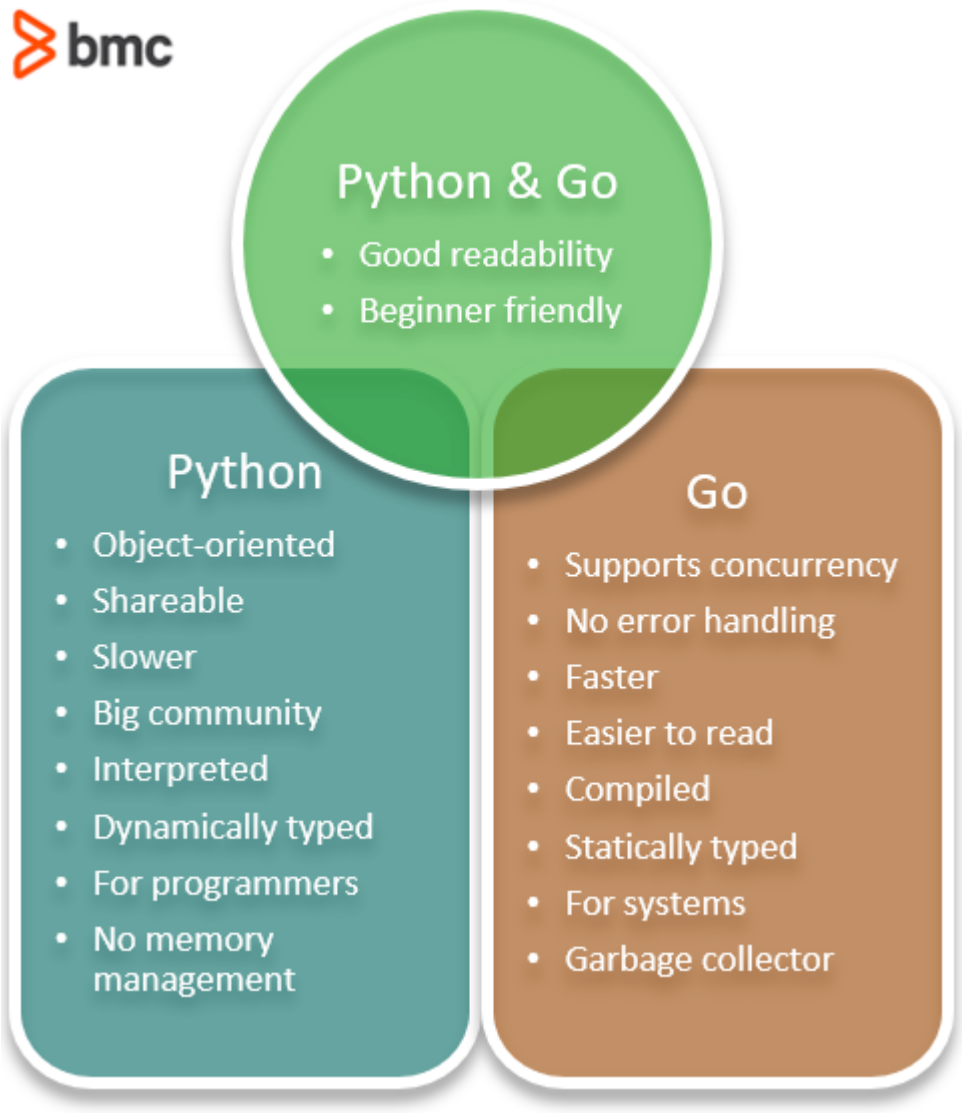
PYTHON VS GO: WHAT'S THE DIFFERENCE?

In this article, we explore the differences, similarities, and use cases for Python and Go, two of the most popular programming languages in the world. Start with this quick comparison of the two, then see how they stack up when it comes to readability, speed, ease of learning, and more.

Python vs Go: At a glance



Python and Go are different, generally serving different purposes. Python is the primary language among data scientists, where Go is the language for server-side commands. Go is the language to use to run software. It is the faster language, performing at Java and C++ speeds.

Python is the language to use for readable, shareable code—hence the large community around it.

Technically, Go is a procedural, functional language built for speed, and Python is an object-oriented, imperative, functional, and procedural language. Go supports concurrency, the ability of an algorithm to run its steps out of order, and Python doesn't.

In short, if you are working with data and your audience is people, use Python. If you are working with servers, use Go.

The Python programming language

Python was first released in 1991. Designed by Guido van Rossum, Python's design philosophy centered around code readability.Python is an interpreted, high-level, general purpose programming language. It is object-oriented.

With its design focused on readability, the Python community will grade each other's code based on

how Pythonic the code is. Because of its readability, Python is great for:

- Learning to program
- Getting ideas down fast
- Sharing code with others

The Go programming language

Short for Golang, Go was first designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson in 2007.

Go is a statically typed, compiled programming language that is open-sourced and maintained by Google. Go is a part of the C-Family programming languages, and it uses a garbage collector to handle memory leaks. When designing Golang, its creators wanted to improve on what already existed, and one of those elements was readability.

Readability

The base standard for unreadable code is Java and C++. Both Python and Go wished to improve upon them. There are a few changes Go makes. Similar to Python, Go does away with:

- The colons at the ends of lines
- The use of brackets and parentheses.

If you're familiar with C-Level languages, then this should look pretty familiar:

Go example:

```
package main

import "fmt"

func split(sum int) (x, y int) {
        x = sum * 4 / 9
        y = sum - x
        return
}

func main() {
        fmt.Println(split(17))
}
```

Returns: 7 10

Python example (Non-Pythonic):

```
even_numbers = []

for number in range(10):
 if number % 2 == 0:
```

```
    even_numbers.append(number)

print(even_numbers)
```

Python example (Pythonic):

```
even_numbers =
print(even_numbers)
```

Returns:

Error handling

Most IDEs will help spot errors in Python code as it is written. Unlike Python, Go has no error handling. It is built for people who already know how to code. While easier to read and type, its simplicity comes at a cost of leaving greater room for errors in the code, meaning there will be more time spent debugging the code.

Speed

When it comes to speed, Go is fast. Go was meant to be fast, whereas that is not Python's sole aim. On most benchmarks, Go beats Python by far. Go even beats Java's speed, which is widely considered to be significantly faster than Python. If it comes down to needing a program to load software quickly, Go is the way to Go.

To demonstrate, here's what one Stack Overflow user said about Go:

"I may have implemented this incorrectly because the results do not make sense. I have a Go program that counts to 1000000000; it finishes in less than a second. On the other hand, I have a Python script; it finishes in a few minutes. Why is the Go version so much faster? Are they both counting up to 1000000000 or am I missing something?"

The answer: Go is just that fast. (Mostly due to static versus dynamic types and compiling). Here are some examples from The Computer Language Benchmarks Game:

n-body

source	secs	mem	gz	busy	cpu load
Go	**6.34**	1,792	1200	6.44	1% 100% 0% 0%
Python 3	545.25	8,160	1196	558.88	1% 1% 1% 100%

spectral-norm

source	secs	mem	gz	busy	cpu load
Go	**1.43**	2,668	548	5.70	99% 99% 99% 100%
Python 3	116.23	50,080	407	463.55	100% 100% 100% 100%

mandelbrot

source	secs	mem	gz	busy	cpu load
Go	**3.73**	28,764	905	14.88	100% 99% 99% 100%
Python 3	170.25	47,568	688	680.12	100% 100% 100% 100%

Libraries

Python is 16 years older than Go, so a much larger community has built up around it. With that community, it has a huge amount of support on Stack Overflow geared towards all types: beginners, lessons, tutorials and how-to's, and, ultimately, libraries. Likely, you will never have to start from scratch in Python.

The Python libraries in particularly are vast. They can get anyone writing code to do what they want on Day One. Exploring data tables is easy with Pandas, and machine learning is simple with TensorFlow and PyTorch.

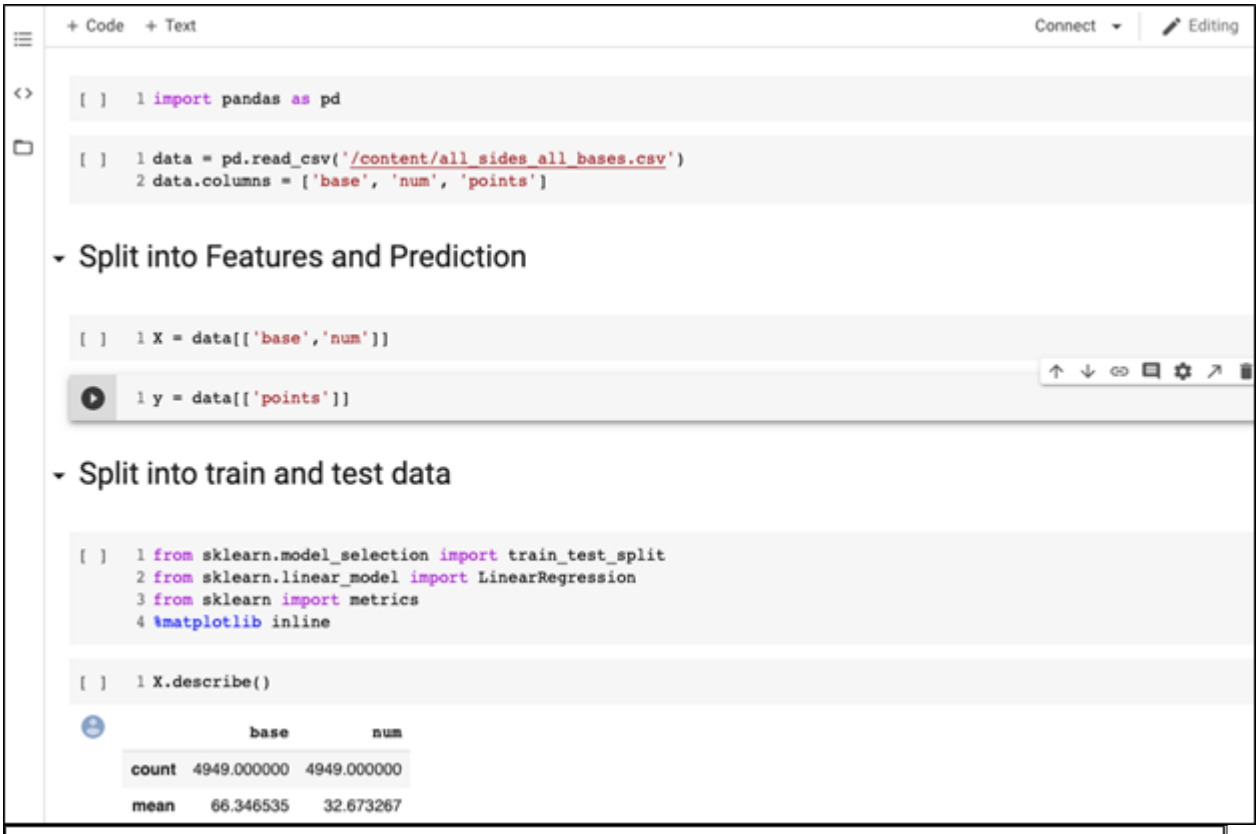
(Learn more about Python development tools.)

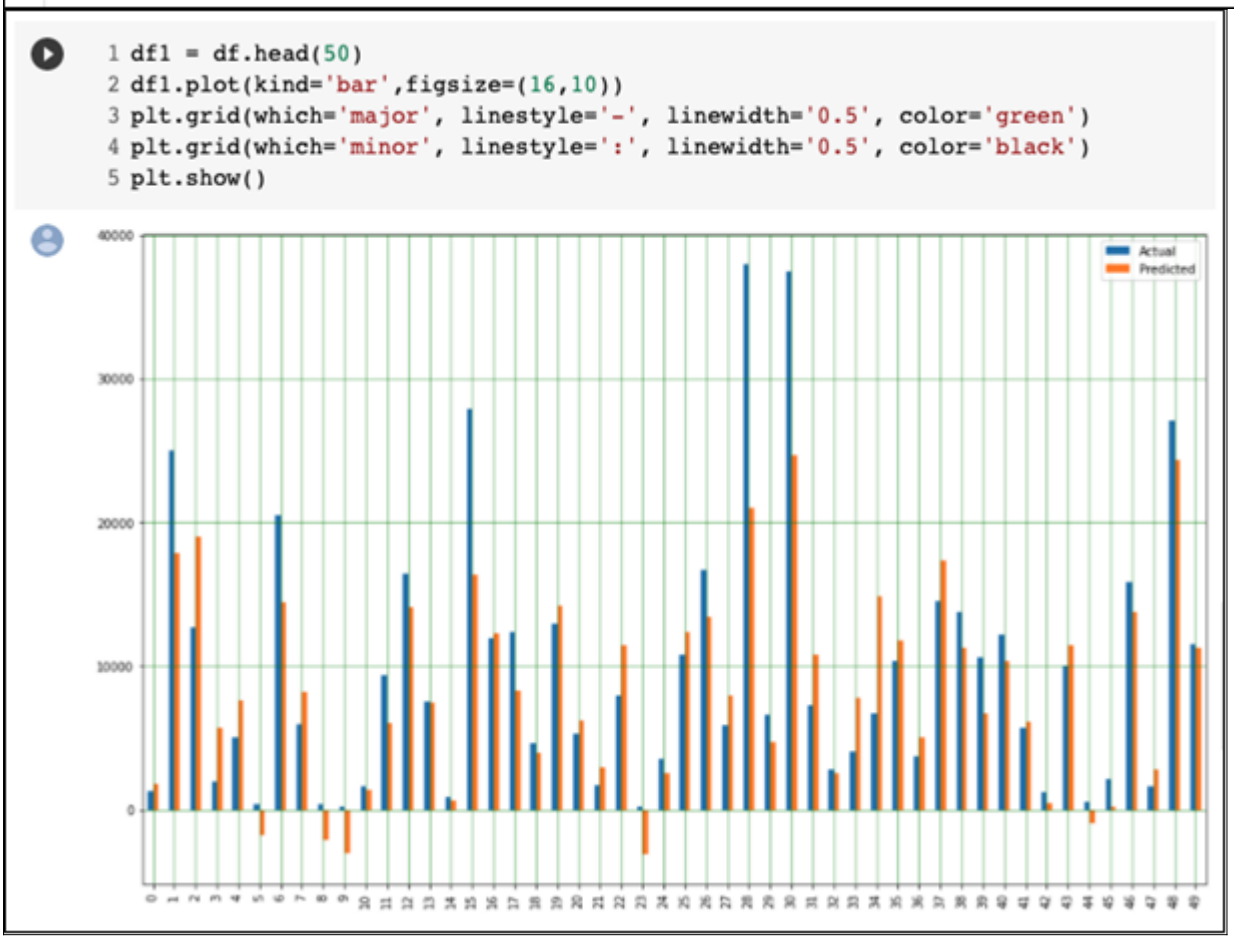
Go does not offer this support. You would have to write all your own scripts to search through data tables. Machine learning would be a huge endeavor to develop in Go. (The top Go libraries all have to do with server maintenance, which shows a lot about the Go community and Go use cases at this point in time.)

Sharing

Sharing executable Python code with Jupyter notebooks is one of the common workflow tools in the data science world. Jupyter Notebooks and Google Colab notebooks allow Python users to read and execute code in a very interactive, reading and writing experience.

Sharing and showing these kinds of charts in Go would be very difficult to get up and running.





Easy to learn

Programming languages are being designed better and smarter. The readability of any code makes it easy to learn. In the coding world, it's accepted truth that if you learn one language, you can learn any other—you just need to dive in.

Discussing the differences matters only to the nit-picky and those who can already read and write in one language. These differences aren't significant in a way that should concern a person's choice to get started in programming. Instead, these differences matter only when choosing which language is best to execute a function.

In the end: Go or Python?

Go and Python are both easy to use and to learn. Go is really fast. Python has a ton of community support.

Right now, in Go's development, Go is mostly used for server-side applications. Python is the data scientist's go-to language, and likely will be for a long time to come. All the ML library developers are putting their time into developing libraries for Python. Go may get there in time, but for now, there is room for two languages in the programmer's toolkit. Use the best for your use case.

Additional resources

For more on programming, explore these resources:

- BMC DevOps Blog
- Python vs Java: Why Python is Becoming More Popular than Java
- Application Developer Roles and Responsibilities
- Enabling the Citizen Data Scientists
- Top 17 Programming & Software Dev Conferences of 2020
- Top DevOps Books to Read in 2020