Python is widely used in real estate technology when building [machine learning](https://www.xome.com/blog/how-the-machine-learning-lifecycle-helps-improve-avms/) models for tasks such as property valuation. At Xome, we use Python to help build our digital solutions that help home buyers and sellers research properties. Even though Python is known to be one of the slower languages compared to C and C++, Python 3.11 and future releases seek to change that.

Python 3.11 came out in 2022 as a part of the Faster CPython effort, which makes Python faster through its optimizations to the interpreter. Because of this, Python 3.11 runs between 10% to 60% faster than 3.10 with an average speed up of 1.25x. An added benefit is that this new release doesn’t require any changes to your code. Instead, Faster CPython optimizes for common code patterns observed.

## **What is Faster CPython?**

Faster CPython is a project that came about from core developer Mark Shannon’s proposal [PEP 659](https://peps.python.org/pep-0659/) to make Python five times faster over several releases. Since the four-stage process proposed was too much effort for one person to complete, Microsoft got involved and hired a team of developers along with Mark Shannon and Python’s creator Guido Van Rossum.

There are many improvements in Python 3.11 and Python 3.12 based on this project.

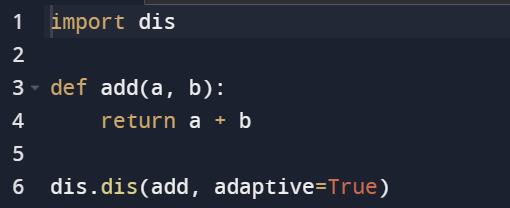
## **Specializing Adaptive Interpreter in Python 3.11**

One notable speed improvement with Python 3.11 is faster code execution. Python code is translated into bytecode and that is the code the interpreter runs. Python 3.11 uses the specializing adaptive interpreter to optimize operations by changing and adapting the bytecode upon execution.

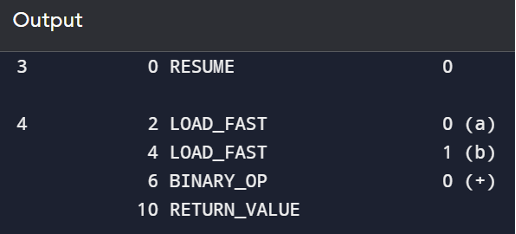
**This happens through two steps: quickening and specialization.**

* **Quickening:** The process of choosing candidates for specialization by looking for any bytecode that is executed several times. Any that would benefit from specialization are then replaced by adaptive versions during this step.
* **Specialization:** Bytecode is replaced through a family of specialized instructions.

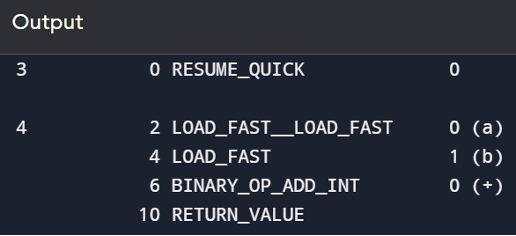
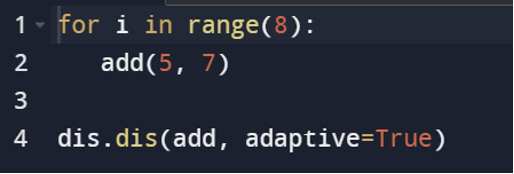
Let’s look at an example of the bytecode instructions using the **dis** package. The **dis** package can help analyze bytecode by disassembling it.



The output of the bytecode instructions should look like this:



When running code where we are adding 5 and 7, 8 times, the output of the bytecode instructions changes.



The bytecode **BINARY\_OP** is now replaced by **BINARY\_OP\_ADD\_INT**. The Python interpreter detected that there were multiple calls done repeatedly with two integers, and then specialized the bytecode based on that detection. The specialized bytecode is what will run in the future.

## **Faster Startup Times**

When the Python interpreter initializes upon an executed script, it can cost programs a few milliseconds before it runs. That startup time can significantly affect your program’s performance.

Overhead happens when Python imports modules, and you can use the -X importtime option to see the time spent importing different modules. You will see from these numbers that imports are happening faster with Python 3.11. This is because it no longer stores bytecode in the **\_\_pycache\_\_** directory. Modules for startup are frozen, and Code Objects and bytecode are statically allocated by the interpreter, reducing the steps in execution making it quicker to load them into memory.

As a result, the interpreter startup time is now 10-15% faster in Python 3.11 with the biggest impact for short-running programs.

## **Changes in Python 3.12**

Python 3.11 and 3.12 mark a significant milestone in speeding up computations in Python. As Python continues to evolve, it becomes increasingly powerful, making it an attractive choice for building digital property solutions. You can keep up with the Faster CPython project progress on [GitHub](https://github.com/faster-cpython/ideas).