Type annotation allows you to clearly communicate the argument types and return type of functions in your code. It’s like giving yourself and other developers hints about what kind of data the variable is supposed to hold. This has several benefits: It reduces the chance of common mistakes, helps in documenting your code for others to reuse, and allows integrated development software (IDEs) and other tools to give you better feedback.

In this reading, you will learn more about annotating variables by type and best practices.

## How to annotate a variable

Think of annotating a variable as if you were to put a label on a container—and anything in that container should hold what the label is describing. Let’s take a look at an example:

name: str = “Betty”

The variable name is declared using a colon (:) which is annotated with the type **str**, indicating that the name variable should hold a string value. And look, it does! Betty—or any name for that matter—is a string, and we know it’s a string because it is in quotes. Let’s look at another example where a variable holds an integer value.

**age: int = 34**

In this example, **age** is the variable, and **int** is the type annotation that provides you and other developers a hint that the age variable should store an integer value.

**Pro tip:** If a function expects a list of integers, you should annotate it as **List[int]**, not just **List**. Being specific with your types can catch more potential bugs and misunderstandings.

## Dynamic typing

Many languages, such as C# or Java, require you to declare variable types, but not Python. One of the great things about Python is that the type of variable can change over time as new values are assigned to it. For example:

**a = 3                  #a is an integer**

**a = “Hello world”      #a is now a string**

Dynamic typing allows programmers to write code more quickly and offers flexibility because you don’t have to explicitly declare the type of variable.

**Note:** Python decides which of the built-in types the variable is and, therefore, how it should behave. For more information, refer to this article on [Built-in types](https://docs.python.org/3/library/stdtypes.html).

## Duck typing

This form of typing comes from the saying, “If it walks like a duck and quacks like a duck, it must be a duck.” Python will infer the variable type at runtime and decide which behaviors are available to the given object.

**a = “Hello world”      #looks like a string**

## Annotating variables with type comments

Another way to annotate variables is to use type comments where the interpreter will ignore the comments.

**captain = “Picard”      # type: str**

**Note:** This way of annotating variables might be useful for cases when you need to know what types belong to which variables but do not want the overhead of using a line interpreter (linter) or IDE on this specific variable.

## Annotating variables directly

Let’s use the same example above to annotate a variable directly.

**captain: str = “Picard”**

**Note:** You might hear annotating variables directly called the more “modern” way to annotate a variable.

Another advantage is that you can use automated tools such as linters, or mypy, to check types to make code more resilient. Most modern IDEs, such as VS Code and JetBrains PyCharm, scan code for type annotations and can use it to help you write better code more quickly!

## How type annotations affect runtime behavior

Any time a library is called, or an IDE works to scan your code, more computational overhead is required.

**Pro tip:** Be strategic when annotating variables by type. This can add unnecessary overhead when overused.

Type annotation is less common with Python users in data science, as it can be burdensome to manually map data every time a new set of data comes in. On the other hand, when doing object-oriented programming or writing functions, using type annotations becomes extremely important because it helps clarify code since you are dealing with more than just the built-in types.

## Key takeaways

Annotating variables by type provides programmers with benefits to make the code easier to read and understand. Python provides different options on how to annotate variables, so choose how you want to annotate them. Just be cautious of over-annotating, creating unnecessary overhead to your code.

## Resources for more information

* Python built-in types <https://docs.python.org/3/library/stdtypes.html>