1. What are the new features added in Python 3.8 version?

Languages change. Languages adapt. Python 2 is no longer supported in the 2020s.

Python is the most popular programming language in 2019 given the rise of data science. Nevertheless, It is normal to feel a bit overwhelmed with the sheer amount there is to learn. Syntax keeps changing. Many new forms of expressions are added in every update. It is difficult to keep track of changes in Python.

There are some features that I wish I had known earlier.

If you also feel the same way, then bad news for you. Python 3.8 was released recently. Do not fear. I have summarized 6 new features in Python 3.8 that every Python beginner should learn.

1. Assignment expressions — readability

Also known as the walrus operator, it is a new operator with syntax := that allows you to assign values to variables as part of a larger expression. This is arguably the most discussed new feature in Python 3.8. Here comes an example.

The assignment expression b := a\*\*2 assigns variable b to a value that is the square of a, which is 36 in this case, and then checks if the value of b is greater than 0.

Assignment expressions can sometimes make your code more compact and readable. Beware not to abuse it as it could make your code harder to understand than necessary in some scenarios.

# DON'T DO THIS!  
a = 5  
d = [b := a+1, a := b-1, a := a\*2]

This operator already exists in other (older) programming languages, and I expect many programmers switching to Python would (ab)use this new feature. Get familiar with it before it is everywhere. Here is a walrus.

2. Arguments types — robustness

Parameters in a Python function can accept two types of arguments.

Positional arguments that are passed positionally

Keyword arguments that are supplied by keywords

In the following example, the values of both parameters a and b can be supplied by positional or keyword arguments. Flexible.

def my\_func(a, b=1):  
 return a+bmy\_func(5,2) # both positional arguments  
my\_func(a=5,b=2) # both keyword arguments

The new version of Python offers a way to specify parameters that can only accept either positional or keyword argument using syntax / and \* for separation.  
\* The latter syntax \* is not new in Python 3.8

In the following example, the first two parameters a and b are positional-only, the middle two c and d can be positional or keyword, and the last two e and f are keyword-only.

Why would you decide to sacrifice flexibility? You should preclude keyword arguments when the parameter names are unhelpful or arbitrary. It can also avoid breaking your code if the parameter names in functions are changed in the future. It contributes to a more robust code.

3. f-string 2.0 — debugging

Python f-string is a game-changer. It is a readable and elegant string formatting syntax that embeds expressions inside strings. This is done by syntax f'{expr}' where an expression is enclosed by curly brackets inside an f-string with an f before quotation marks.

The new update enables the using equal sign = as a format specifier in expressions inside f-strings with syntax f'{expr=}'. The output string will include both the variable name and its value, with an equal sign in between = as shown below.

We often want to print out values of variables for documentation or debugging purposes. This allows easy debugging with minimal effort.

4. Reversible dictionary — order

Dictionaries are now iterable in reversed insertion order using [reversed()](https://docs.python.org/3/library/functions.html#reversed).

5. New modules — metadata

There is a new importlib.metadata module that allows you to read metadata from third-party packages. You can extract version numbers of packages in your scripts.

6. Continue — finally

It used to be illegal to use continue statement in a finally clause due to a problem with the implementation. Not anymore.

The Takeaways

Note that I did not mention some advanced features that have little to do with how a normal programmer writes codes for small projects. They include multiprocessing shared memory, new Pickle protocol etc. for those who are interested.

There you are — 6 new Python features that even Python beginners could benefit from. Before jumping into Python 3.8, be sure you are familiar with some basic Python features. You can [sign up for my newsletter](http://edenau.mailchimpsites.com/) to receive updates on my new articles. If you are interested in Python, the following articles might be useful:

2 . What is monkey patching in Python?

Monkey Patching is an exciting topic of Python. Monkey-patching is a term that refers to modifying a class or module at a run time. In simple words, a class or module's work can be changed at the runtime. Let's understand this concept by real-life example.

When we work on a large project, we may encounter a situation where the third-party library is not working well. So we attempt to revise (or change) it from our project. This process is known as monkey patching in Python. Generally, it is avoided by the developer. However, it is a part of the development process.

In monkey patching, we can reopen the class and modify its behavior

We will learn how we can use monkey-patching in using the Python code.

We know that [Python](https://www.javatpoint.com/python-tutorial)

is a dynamic language; classes are mutable, so we can alter them when we want. Let's understand the following example.

Example -

import inspect

class MonkeyPatch:

    def \_\_init\_\_(self, n1):

        self.n1 = n1

    def add(self, other):

        return (self.n1 + other)

obj1 = MonkeyPatch(10)

obj1.add(20)

print(inspect.getmembers(obj1, predicate=inspect.ismethod))

Output:

30

[('\_\_init\_\_', >), ('add', >)]

What is use of monkey patching in Python?

In Python, the term monkey patch refers to dynamic (or run-time) modifications of a class or module. In Python, we can actually change the behavior of code at run-time. We use above module (monk) in below code and change behavior of func() at run-time by assigning different value

1. . What is the difference between a shallow copy and deep copy?

Difference between Shallow and Deep copy of a Class

In Shallow copy, a copy of the original object is stored and only the reference address is finally copied. In Deep copy, the copy of the original object and the repetitive copies both are stored.

What is Shallow copy?

A bitwise copy of an object, where a new object is created and it has the same copy of the values in the original object, is called a Shallow copy. If any of the object fields refer to the other object then in such cases only the reference address is copied.

What is Deep copy?

When the process of copying occurs repetitively and a copy of the object is always copied in another object, then it is called deep copy. In this process, initially, a new collection of the object is constructed, and then the copies of the child object frequently populate those found in the original.

Difference between Shallow copy and Deep copy

|  |  |  |
| --- | --- | --- |
| S.No. | Shallow Copy | Deep Copy |
| 1. | In Shallow copy, a copy of the original object is stored and only the reference address is finally copied. | In Deep copy, the copy of the original object and the repetitive copies both are stored. |
| 2. | Shallow copy is faster than Deep copy. | Deep copy is slower than Shallow copy. |
| 3. | The changes made in the copied object also reflect the original object. | There is no reflection on the original object when the changes are made in the copied object. |
| 4. | It stores references of the object in the main memory. | It stores copies of the object values. |

1. . What is the maximum possible length of an identifier?

An identifier can have a maximum length of 79 characters in Python.

Python is one of the most popular programming languages. Guido van Rossum created it, and it was released in 1991. The language is often used in web development (server-side), software development, mathematics, and system scripting. The majority of modern operating systems are written in C, which means that compilers and interpreters are also written in C. This is also true for Python; its most popular/traditional implementation is written in C and is known as CPython. Regardless of whether you are an experienced programmer with C++, Java, or Python, Python is extremely suitable for beginners.

The Python programming language can be used on a server to write web applications. Workflows can also be created using Python. The language is capable of connecting to databases. It is also capable of reading and writing files. Python can deal with large amounts of data and process complex mathematical operations. Rapid prototypes can be created with Python, or production-ready applications can be developed with Python.

With some similarities to the English language and influences from mathematics, Python was designed to be comprehensible. Unlike many other programming languages, Python uses a new line to specify the end of a command instead of a semicolon or parentheses. In Python, whitespace is used to mark the scope of loops, functions, and classes using indentation. Most programming languages do the same thing by using curly brackets

1. What is generator comprehension?

A generator comprehension is a single-line specification for defining a generator in Python. It is absolutely essential to learn this syntax in order to write simple and readable code. Note: Generator comprehensions are not the only method for defining generators in Python.

So what's the difference between Generator Expressions and List Comprehensions? The generator yields one item at a time and generates item only when in demand. Whereas, in a list comprehension, Python reserves memory for the whole list. Thus we can say that the generator expressions are memory efficient than the lists.