Python Instance Variables

Summary: in this tutorial, you'll learn about Python instance variables including data variables and function variables.

Introduction to the Python instance variables

In Python, class variables (https://www.pythontutorial.net/python-oop/python-class-variables/) are bound to a class (https://www.pythontutorial.net/python-oop/python-class/) while instance variables are bound to a specific instance of a class. The instance variables are also called instance attributes.

The following defines a `HtmlDocument` class with two class variables:

```
from pprint import pprint


class HtmlDocument:
    version = 5
    extension = 'html'
```

```
pprint(HtmlDocument.__dict__)

print(HtmlDocument.extension)
print(HtmlDocument.version)
```

Output:

```
mappingproxy({'__dict__': <attribute '__dict__' of 'HtmlDocument' objects>,
              '__doc__': None,
              '__module__': '__main__',
              '__weakref__': <attribute '__weakref__' of 'HtmlDocument' objects>,
              'extension': 'html',
              'version': 5})
```

The `HtmlDocument` class has two class variables: `extension` and `version`. Python stores these two variables in the `__dict__` attribute.

When you access the class variables via the class, Python looks them up in the `__dict__` of the class.

The following creates a new instance of the `HtmlDocument` class:

```
home = HtmlDocument()
```

The `home` is an instance of the `HtmlDocument` class. It has its own `__dict__` attribute:

```
pprint(home.__dict__)
```

The `home.__dict__` is now empty:

```
{}
```

The `home.__dict__` stores the instance variables of the `home` object like the `HtmlDocument.__dict__` stores the class variables of the `HtmlDocument` class.

Unlike the `__dict__` attribute of a class, the type of the `__dict__` attribute of an instance is a dictionary (https://www.pythontutorial.net/python-basics/python-dictionary/) . For example:

```
print(type(home.__dict__))
```

Output:

```
<class 'dict'>
```

Since a dictionary (https://www.pythontutorial.net/python-basics/python-dictionary/) is mutable (https://www.pythontutorial.net/advanced-python/python-mutable-and-immutable/) , you can mutate it e.g., adding a new element to the dictionary.

Python allows you to access the class variables (https://www.pythontutorial.net/python-oop/python-class-variables/) from an instance of a class. For example:

```
print(home.extension)
print(home.version)
```

In this case, Python looks up the variables extension and version in `home.__dict__` first. If it doesn't find them there, it'll go up to the class and look up in the `HtmlDocument.__dict__` .

However, if Python can find the variables in the `__dict__` of the instance, it won't look further in the `__dict__` of the class.

The following defines the `version` variable in the `home` object:

```
home.version = 6
```

Python adds the `version` variable to the `__dict__` attribute of the `home` object:

```
print(home.__dict__)
```

The `__dict__` now contains one instance variable:

```
{'version': 6}
```

If you access the version attribute of the `home` object, Python will return the value of the `version` in the `home.__dict__` dictionary:

```
print(home.version)
```

Output:

```
6
```

If you change the class variables, these changes also reflect in the instances of the class:

```
HtmlDocument.media_type = 'text/html'
print(home.media_type)
```

Output:

```
text/html
```

Initializing instance variables

In practice, you initialize instance variables for all instances of a class in the `__init__` (https://www.pythontutorial.net/python-oop/python-__init__/) method.

For example, the following redefines the `HtmlDocument` class that has two instance variables `name` and `contents`

```
class HtmlDocument:
    version = 5
    extension = 'html'
    def __init__(self, name, contents):
        self.name = name
        self.contents = contents
```

When creating a new instance of the `HtmlDocument`, you need to pass the corresponding arguments like this:

```
blank = HtmlDocument('Blank', '')
```

Summary

- Instance variables are bound to a specific instance of a class.
- Python stores instance variables in the `__dict__` attribute of the instance. Each instance has its own __dict__ attribute and the keys in this `__dict__` may be different.
- When you access a variable via the instance, Python finds the variable in the `__dict__` attribute of the instance. If it cannot find the variable, it goes up and look it up in the `__dict__` attribute of the class.