Learning Import Statements in Python

Made for UMBC IS296 (Introduction to Data Science Concepts)

by Olisaemeka Okolo

Content:

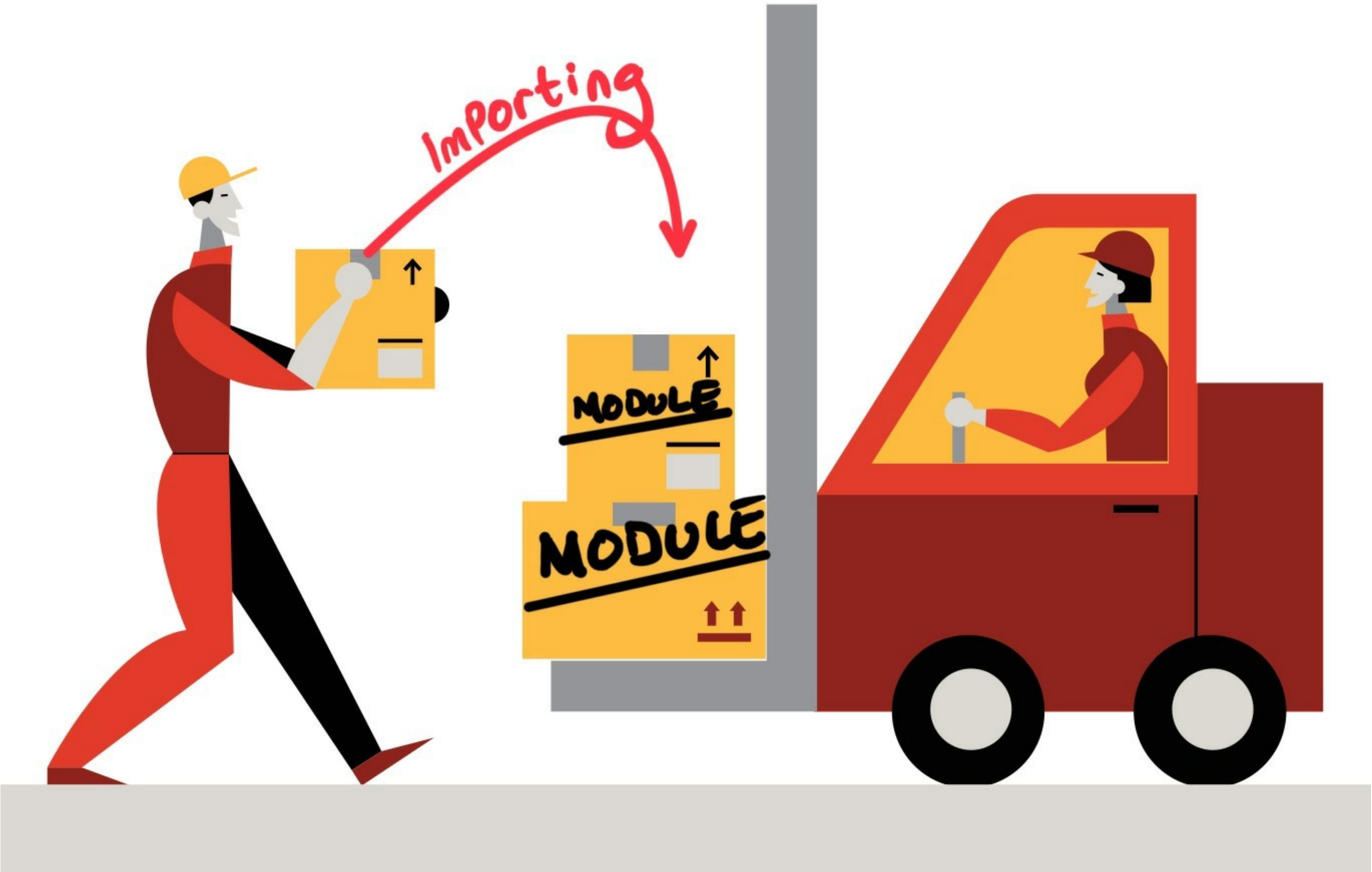
1. What are import statements?
2. Different Types of Import Stements
3. Import Statements

for more information on import statements check this link

Let's start off with defining what a module is. According to Analytics Vidhya -- who is one of my favourite data science bloggers:

In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.

in essence, a module is a toolbox, and you get the tools by importing them



In [ ]:

```
# Example of an import statement
import numpy as np
```

Let's break this down,

1. "**import**" is the instruction
2. "**numpy**" is the module or toolkit that we are using
3. "**as np**" is a pseudonym for numpy so that we don't have to keep typing "numpy" whenever we need to use the tools from the module.

i.e instead of "numpy.array", we can use "np.array"

Different types of import statements

In lab 2, there was a question that required you to use the following code,

In [ ]:

```
from IPython.display import Image
```

Let's break this down as well: This is different that the usual import statements because it doesn't start with the word "import."

The "from" keyword is essentially saying "from this toolbox, import this specific tool"

1. "**from**": Tells python, to search the module
2. "**Ipython.display**" is the module we are searching
3. "**import**" basically tells python to give us this function
4. "**Image**" is the function or tool that we specifically need

You may ask what the point of this is. While I am yet to have found any difference in speed and performance when importing the entire module vs a specific function, a lot of developer say it makes a diffence in speed.

For this class, you are fine with doing either, though some questions require you to import specific functions.

In [ ]:

```
#ignore the output and warning statements
from datascience import *
```

```
/usr/local/lib/python3.7/dist-packages/datascience/tables.py:17: MatplotlibDeprecationWar
ning: The 'warn' parameter of use() is deprecated since Matplotlib 3.1 and will be remove
d in 3.3.  If any parameter follows 'warn', they should be pass as keyword, not positiona
lly.
  matplotlib.use('agg', warn=False)
/usr/local/lib/python3.7/dist-packages/datascience/util.py:10: MatplotlibDeprecationWarni
```

```
ng: The 'warn' parameter of use() is deprecated since Matplotlib 3.1 and will be removed
in 3.3.  If any parameter follows 'warn', they should be pass as keyword, not positionall
y.
  matplotlib.use('agg', warn=False)
```

This is a variation of the "**from x import y**" statement The "*" basically translates to "import everything"

This is different from a regular "**import numpy**" statement because it imports all functions and you do not have to call the module name when using the functions. It saves a bit of typing and is acceptable for this class.

HOWEVER THIS PRACTICE IS NOT COMMONPLACE IN THE PYTHON WORLD.

Since you don't call the names of the modules, it is difficult to trace where a problem is, if it occurs.

You always want to follow conventions because you will almost never be the only person working on your code. You want to make sure that everybody that reads your code can understand it

---

Import statements for this class

For this calss, you won't be using too many different modules.

Below are the 2 import statements you will see most often.

In [ ]:

```
import nunmpy as np
from datascience import *
```

Pro tips:

1. When in doubt, remeber that Python import statements are very similar to english language

"From X, import Y."

1. "from" and "import" are **ALWAYS** lowercase

In [ ]:

```

```

image attributions:

all images are products of the *Ouch* library from icons8

Wrench

Construction

In [ ]:

```

```