3rd party packages

Lesson Duration: 60 minutes

Learning Objectives

- Understand what 3rd party packages are
- Read and understand the usage of 3rd party packages from documentation
- Be able to utilise someone else's code

What is a package?

In development, packages usually refer to code written by other developers that we are utilising. You've already done this plenty of times - none of us actually wrote a function called `print()`, `input()`, `type()`, etc, we were just using code written by the creators and maintainers of the Python programming language.

3rd party packages also work on the same principle - its code someone wrote that we can use to solve our problems.

A couple of things to keep in mind:

1. Be careful to only install packages from sources you trust! If you found a package online and plenty of people endorsed it recently, chances are it's safe
2. Check if the package was maintained regularly - if no one updated at least the documentation for the package for the last couple of years, it's more than likely that the project is abandoned

We are going to explore some very basic packages, and we will try to improve our skills to look for them, and how to read the documentation provided.

the PIP package manager

PIP is a package manager that comes installed with every Python distribution. This means that when Python is installed, you also installed PIP.

We need to open up our terminal in order to check and use PIP. You can click on the top of the screen on the terminal tab, or press `CTRL` + `` ` ``. (backtick, not single quote - bottom

left on most keyboards).

Once the terminal is open, we can type the following commands:

> Please note that on Macs you might have to use `pip3` instead of `pip` as a command, and if you encounter an error, you can try the following: On Mac, `python3 -m pip3 install <package name>`, on Windows, `python -m pip install <package name>`. If you still struggle, refer to the video!

```
pip3 --version
```

This should give us a version number if its correctly installed.

Intro to PyPI

PyPI is a great package management tool. We can search for multiple different packages to help us with tasks ranging from data science to machine learning, building desktop apps to even creating video games!

You can search by topic and by name very easily.

Search for "wikipedia"!

The first result coming up should be the one we are looking for. It's an older package, but will serve as a great introduction into using 3rd party packages!

Wikipedia Package link.

If you read the quickstart/documentation links, you can get familiar with the use of the package in no time - for tips, refer to the video!

Using a package

Let's try using this Wikipedia package! First we need to install it.

In the terminal, type `pip install wikipedia` or `pip3 install wikipedia`. If unsuccessful, try `python -m pip install wikipedia` or `python3 -m pip3 install wikipedia`

Once it successfully installed, we can create a new file - let's name it `wikipedia_intro.py`

In this file, we need to use a new feature of Python - `import`. Importing something makes

it available for us to use it.

Based on the documentation, we can save into a variable the result of the `.page()` method call for further work!

Write the following in the file:

```
import wikipedia

london = wikipedia.page("London")
```

If you print out `london`, you will see that its a new data type - this is created by the developers behind the package!

Further investigation of the documentation tells us that we can get the contents of the page with the `.content` attribute (Be aware, this is not a method - no `()` needed!)

```
import wikipedia

london = wikipedia.page("London")

print(london.content)
```

So what could we do if the task is to print out every sentence in this page that contains the word `population`?

First, after briefly googling for it, we could learn about the `.split()` method - it can be called on strings, and then it will separate a string based on the separator we pass in as an argument for the methods (if none is passed, it will split on whitespaces!)

```
import wikipedia

london = wikipedia.page("London")

split_sentences = london.content.split(".")
```

Next, after another round of googling, we can learn about the other use of the `in` keyword - we can use it to check for substrings in a string! So we can loop through all strings, check if the word `population` can be found in that string, then if it does, print it out!

```
import wikipedia

london = wikipedia.page("London")

split_sentences = london.content.split(".")

sentences_with_population_substring = []

for sentence in split_sentences:
    if("population" in sentence):
        sentences_with_population_substring.append(sentence)

print(sentences_with_population_substring)
```

Great work!

Conclusion

In this lesson we learned about how can we use 3rd party packages, where to install them from, and gained first hand experience in researching doucmentation to figure out more complex problem solving!