Practical Session 3

Foundations Spatial Data Science

Today Goals & Aims

- Review Python List concepts
- Understand Dictionary
- Overview Data Structure
- Thinking like a Programmer

Today Goals & Aims

Term Calendar

	Weekly Topic		WORKSHOP Lead	WORKSHOP Date (Monday)	PRACTICAL Date Groups 1,2,3 (Tuesday)	PRACTICAL Date Groups 4,5,6 (Wednesday)
1	Getting Oriented	initiate	David, Nicolas	4 Oct	4 Oct	5 Oct
2	Foundations (Part 1)	initiate	Nicolas	11 Oct	11 Oct	12 Oct
3	Foundations (Part 2)	initiate	Nicolas	18 Oct	18 Oct	19 Oct
4	Objects & Classes	initiate	David	25 Oct	25 Oct	26 Oct
5	Numeric Data	engage	David	1 Nov	1 Nov	2 Nov
	Reading Week					
6	Spatial Data	engage	Nicolas	15 Nov	15 Nov	16 Nov
7	Textual Data	engage	Nicolas	22 Nov	22 Nov	23 Nov
8	Visualising Data	solve	David	29 Nov	29 Nov	30 Nov
9	Classifying Data	solve	David	6 Dec	6 Dec	7 Dec
10	Clustering Data	solve	Nicolas	13 Dec	13 Dec	14 Dec

Python Lists vs. Dictionaries

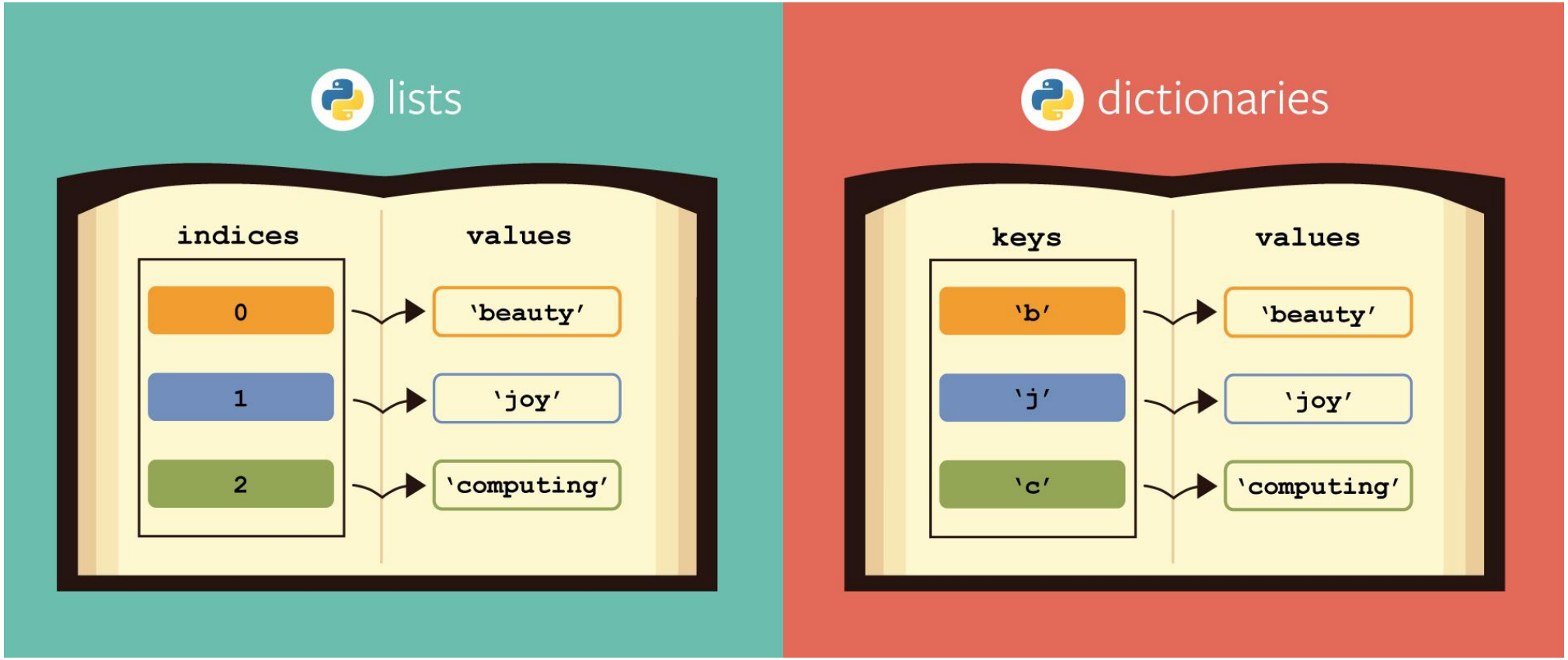


Image Source: https://www.analyticsvidhya.com/blog/2021/06/working-with-lists-dictionaries-in-python/

Python Dictionaries

A python dictionary is a **mutable, nested** container that consists of a collection of **key-value pairs**.

You can view it as a container that stores mappings of **unique keys** to associated **values**.

How to create Dictionaries?

1. **Using curly braces.**
   a. dict = {‘key’: ‘value’}
2. **Using dict() built-in function.**
   a. dictionary = dict(key=value)

Iterate through dictionaries

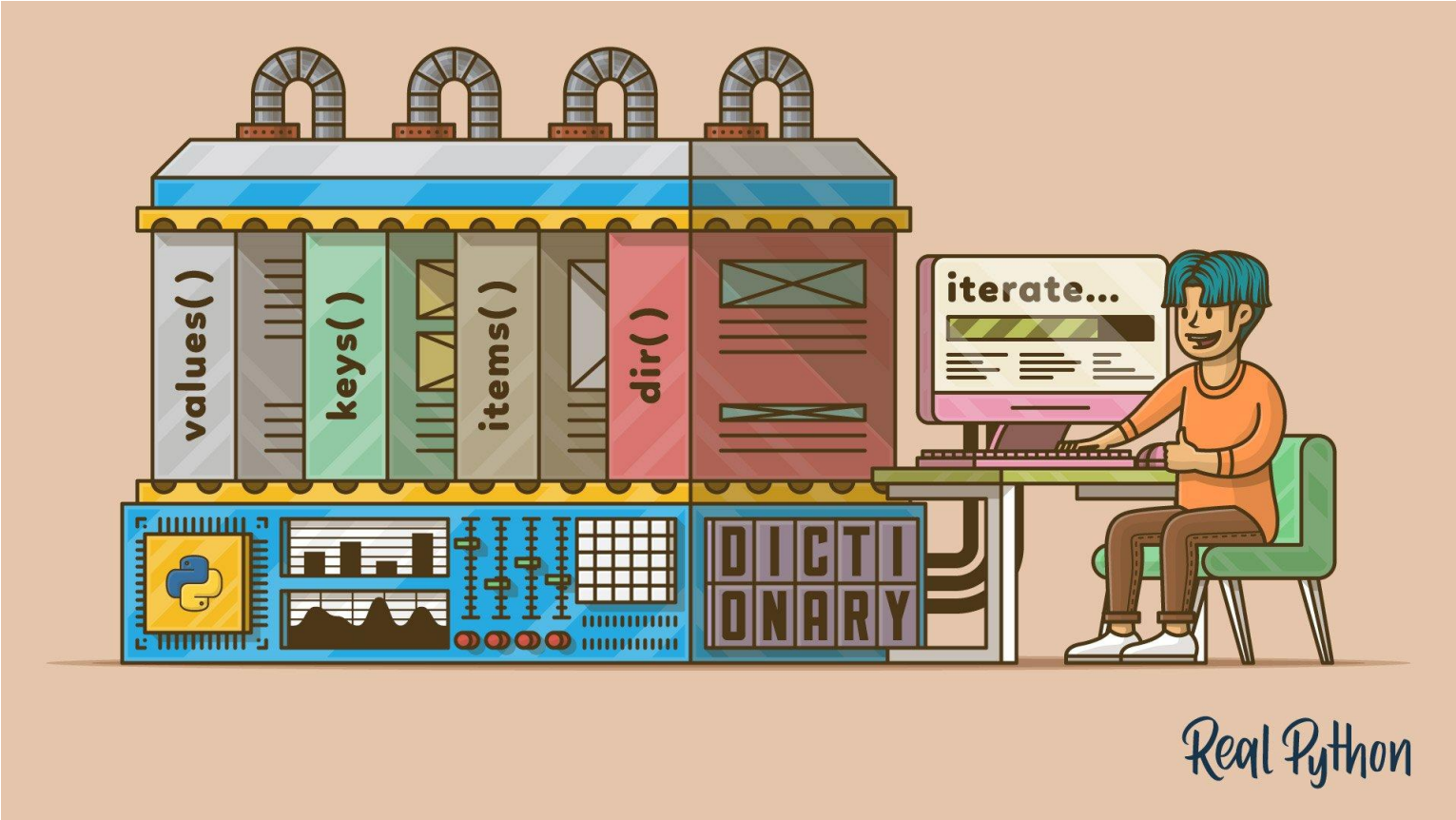


Image Source: https://realpython.com/iterate-through-dictionary-python/

Iterate through dictionaries

dict = dict(A=‘Amsterdam’, B=‘Belfast’, C=‘Cork’)

dict = {‘A’: ‘Amsterdam’, ‘B’: ‘Belfast’, ‘C’: ‘Cork’}

dict.values() -> Return values : [‘Amsterdam’, ‘Belfast’, ‘Cork’]

dict.keys() -> Return keys : [‘A’, ‘B’, ‘C’]

dict.items() -> Return items:[('A', 'Amsterdam'), ('B', 'Belfast'), ('C', 'Cork')]

Spot the difference ?

```
opt_1 = [{'name': 'Amsterdam',
          'country': 'NL'},
          {'name': 'Belfast',
          'country': 'UK'},
          {'name' : 'Cork',
          'country': 'IE'}]
```

```
opt_2 = { 'Amsterdam': {
              'country': 'NL'},
          'Belfast': {
              'country': 'UK'},
          'Cork': {
              'country': 'IE'}}
```

Spot the difference ?

```
opt_1 = [{'name': 'Amsterdam',
          'country': 'NL'},
          {'name': 'Belfast',
          'country': 'UK'},
          {'name' : 'Cork',
          'country': 'IE'}]
```

A list of dictionaries.

```
opt_2 = { 'Amsterdam': {
              'country': 'NL'},
          'Belfast': {
              'country': 'UK'},
          'Cork': {
              'country': 'IE'} }
```

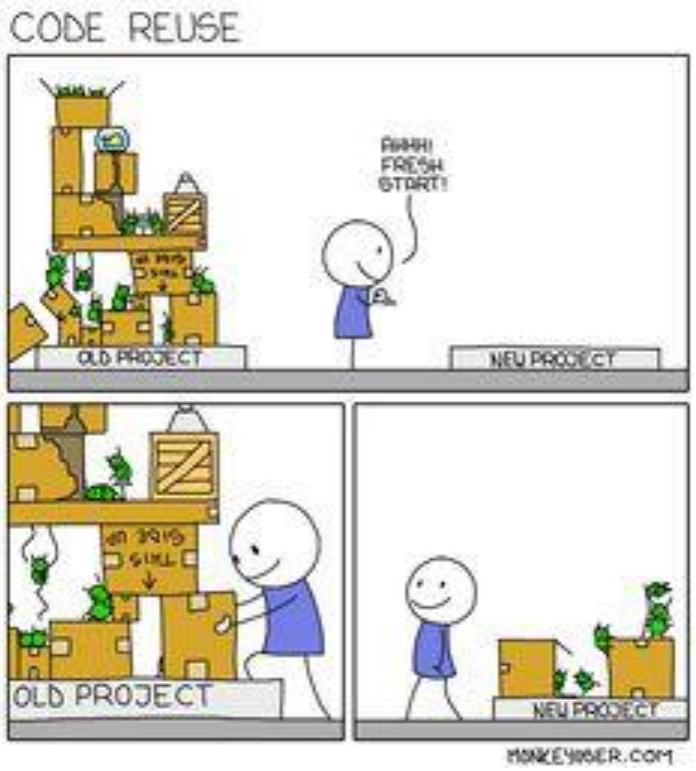
A nested dictionary.

"A dictionary of 3 dictionaries"

Data Structure in Python!

Data Structure	Ordered	Mutable	Constructor	Example
List	Yes	Yes	`[ ]` or `list()`	`[5.7, 4, 'yes', 5.7]`
Tuple	Yes	No	`( )` or `tuple()`	`(5.7, 4, 'yes', 5.7)`
Set	No	Yes	`{}`* or `set()`	`{5.7, 4, 'yes'}`
Dictionary	No	Yes**	`{ }` or `dict()`	`{'Jun': 75, 'Jul': 89}`

Thinking like a programmer!



Thinking like a programmer!

1. **Understand the Problem**
   a. What is the goal? What do we want to achieve?
2. **Use Functions & Packages**
   a. What are the functions and packages out there?
3. **Recycle Code**
   a. Stackoverflow - Did someone already encounter the problem?
4. **Make a Plan - Break into steps.**
   a. What are the separated tasks?

Python - Building Blocks

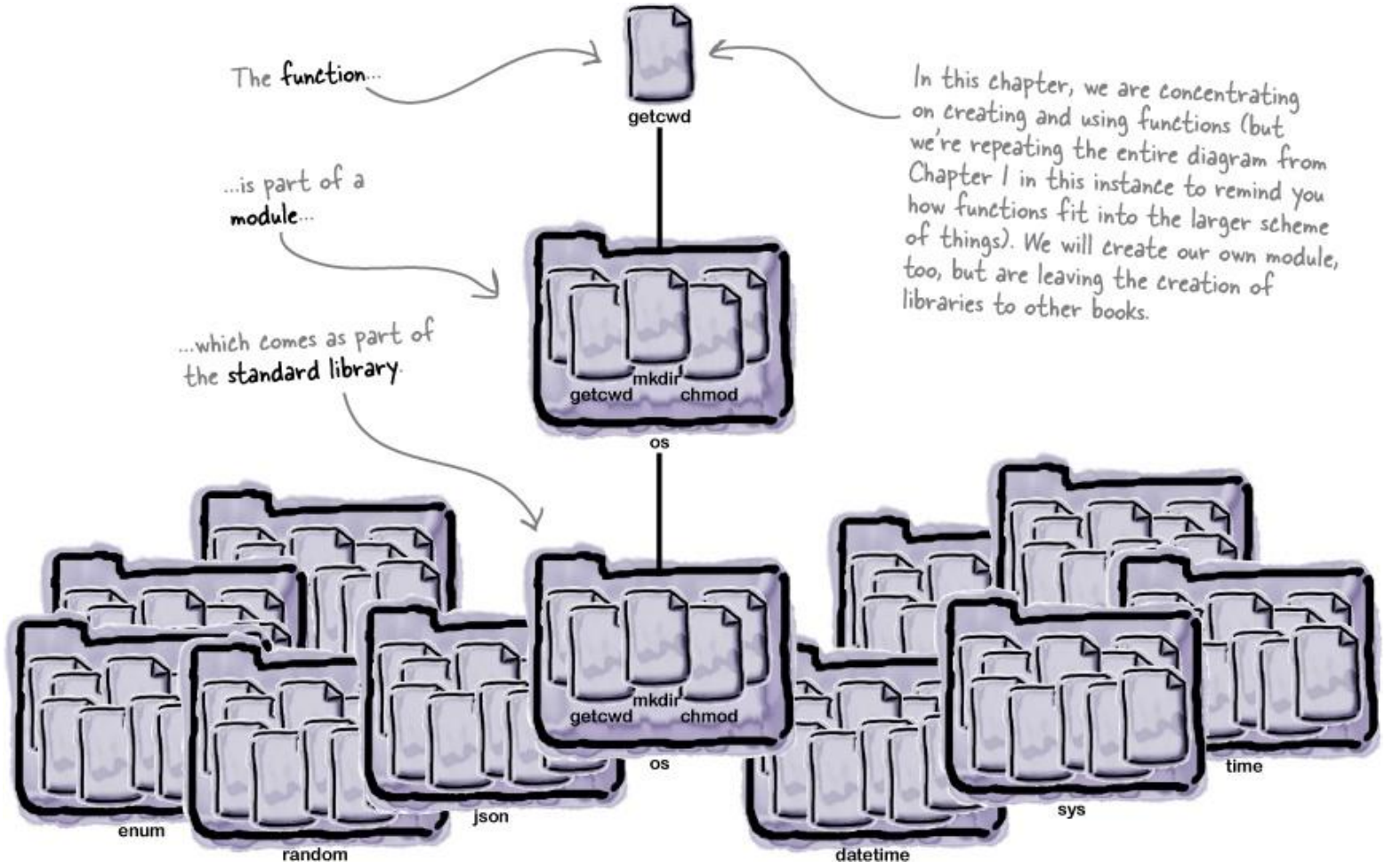


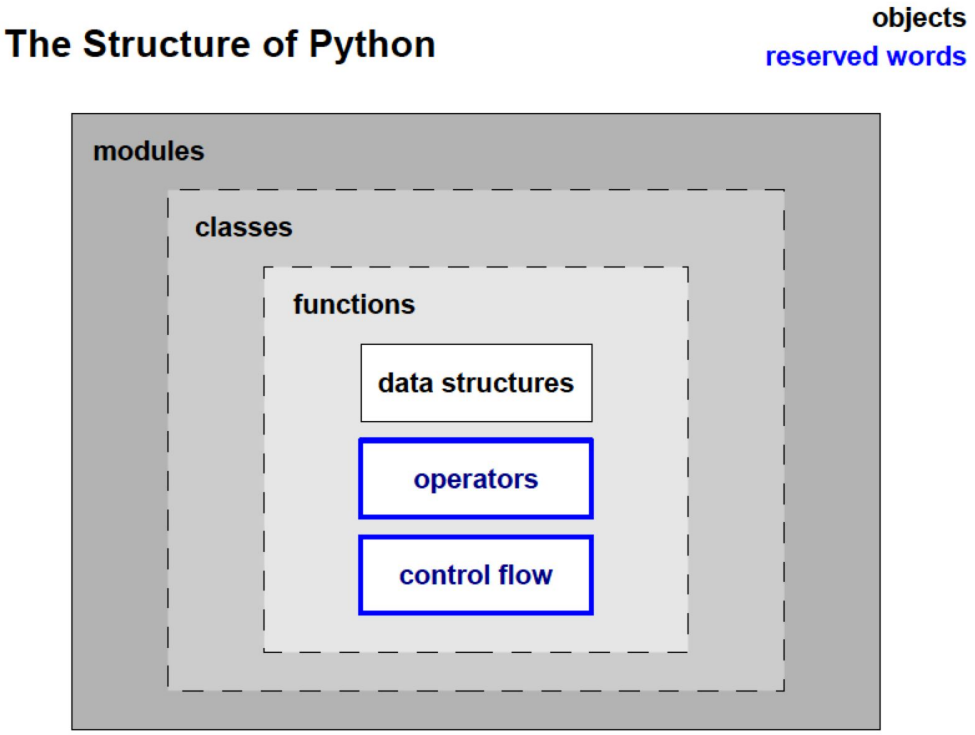
Image Source: https://learning.oreilly.com/library/view/head-first-python/9781491919521/ch04.html#introducing_functions

Python - Building Blocks

Modular programming: Process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or pieces.

1. **Function** - Collection of variables and expressions
2. **Class** - Collection of functions
3. **Module** - Python script file (.py), which is a collection of class, functions, expressions, and variables with specific functionality.
4. **Packages** - Solution to manage Python modules.

Python Modules and Packages



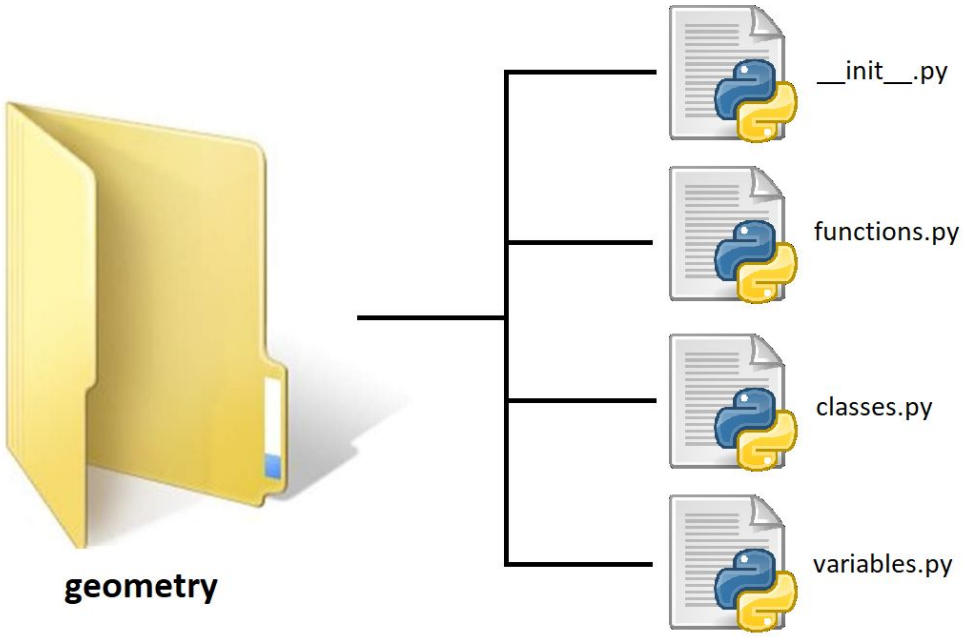


Image Source: https://gtpb.github.io/PPB18/assets/4_PythonStructureModulesImport

Example - Building Blocks

Imagine we are asking a computer to prepare a raspberry pie. We have a set of **ingredients** (raspberries, butter, flour, almond), a set of **instructions** (gounding almond, mixing butter and flour), a set of **measurements** (preparation time, ingredient quantity).

We found the recipe for our raspberry pie in a **recipe** book that contain different sections to host dinner party (**starters, cocktails, desserts**).

Example - Building Blocks

How would you define the functions, modules, packages for the computer to bake the raspberry pie?

Example - Building Blocks

How would you define the functions, modules, packages for the computer to bake the raspberry pie?

Functions: Ingredients(), Instructions(), Measurements()

Module: baking.py

Package: Recipe (contains the modules baking.py, cocktail.py, starter.py)

How to use Modules and Packages

import module

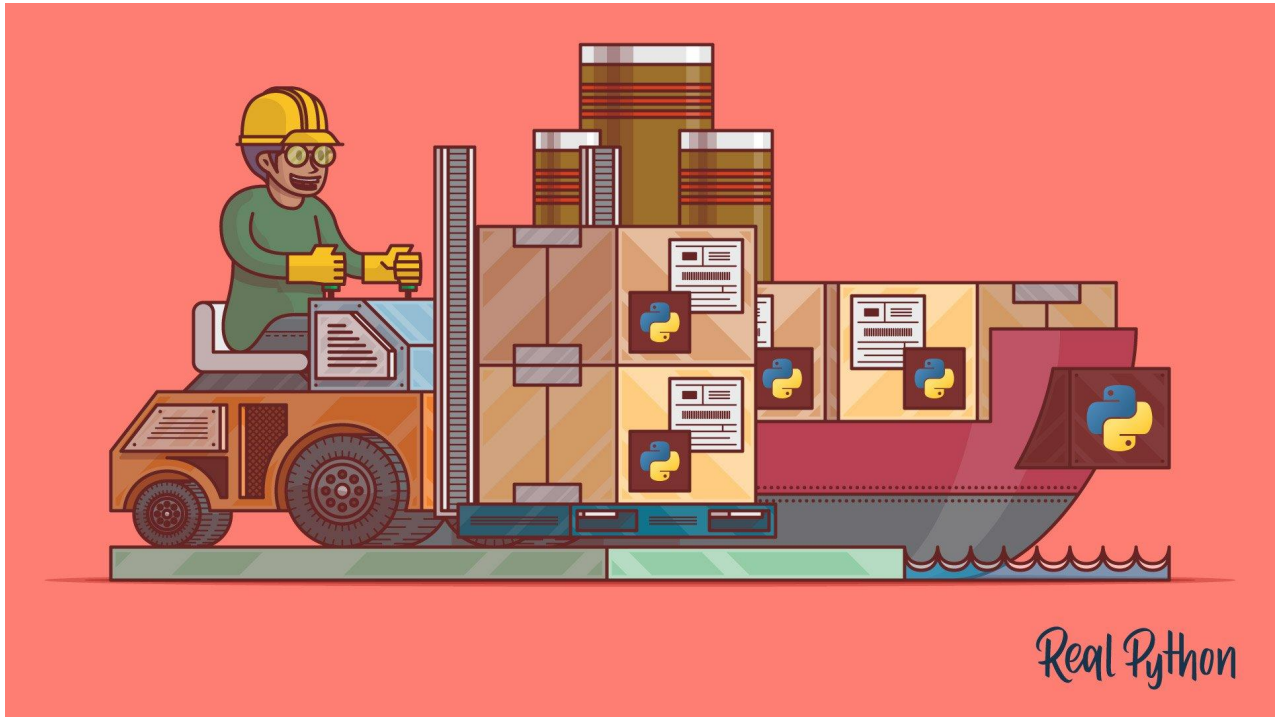


Image Source: https://realpython.com/python-import/

How to use Modules and Packages

1. import all the module (all functions of the module)
   a. **import** math
2. import only some functions of the module
   a. **from** math **import** pi, cos
3. look at which functions are in each module and find out what they do
   a. **dir(math) , help(math)**

Module Maths

cos

pi

exp

degrees

log

Explore Packages - Access data

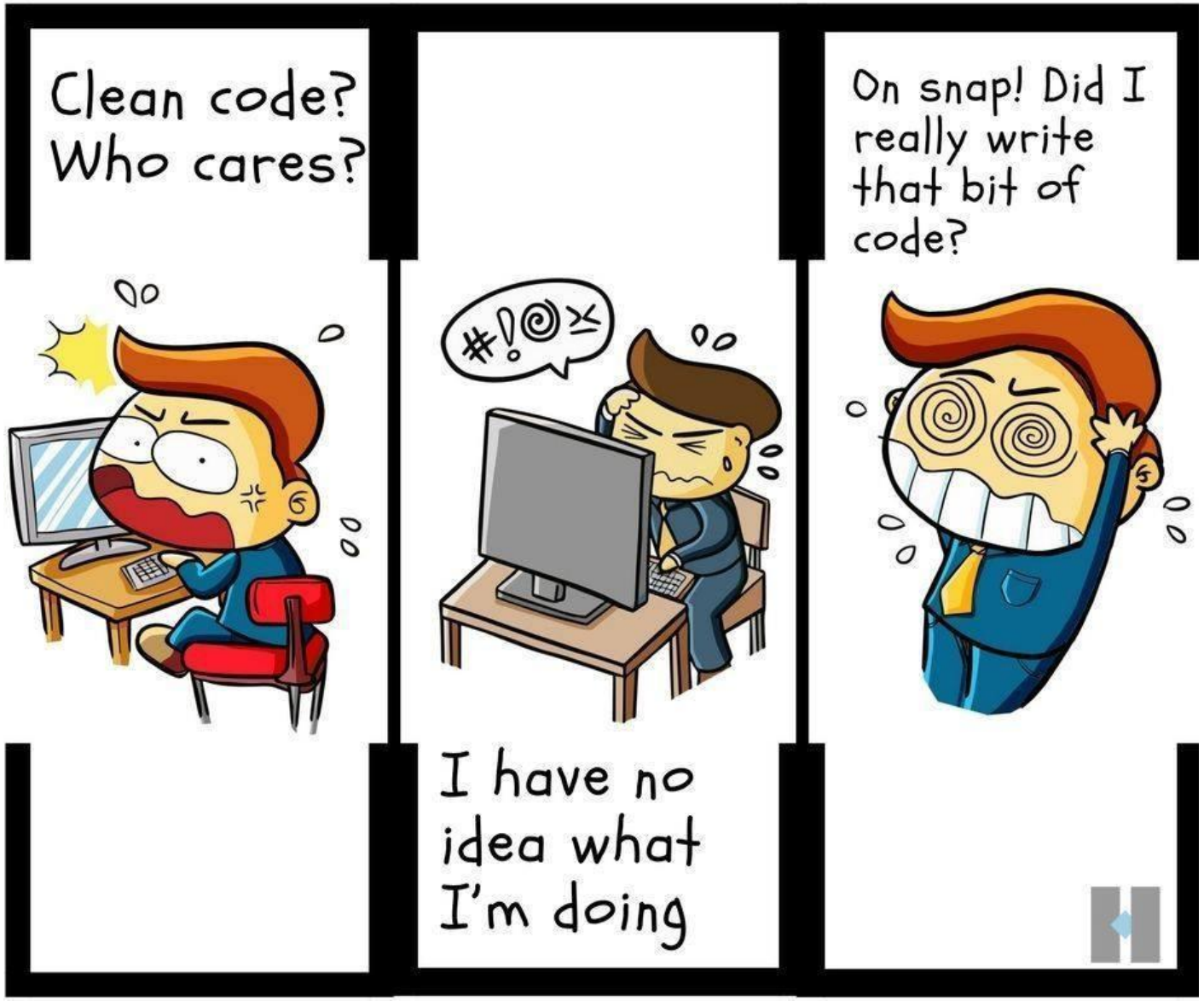
1. **urllib.request**
   a. library to open URLs
2. **csv - comma separated values**
   a. module to read and write tabular data in csv format.
3. **pandas**
   a. library for manipulating DataFrame
4. **numpy**
   a. library for operating with multi-dimensional array and matrices

The Zen of Python

Time to practice !

References

- *"Python Modules & Packages - An Introduction"* by John Sturtz, Link: https://realpython.com/python-modules-packages/
- *"Python Humor - The Zen of Python"* by The Python Software Foundation, Link: https://www.python.org/doc/humor/#the-zen-of-python


Clean code? Who cares?
#!@%
On snap! Did I really write that bit of code?
I have no idea what I'm doing