
IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN

Design of experiments

and the Repeatability Problem

Formal Definition

- The **design of experiments** (**DOE**, **DOX**, or **experimental design**) is the design of any task that aims to describe and explain the variation of information under conditions that are hypothesized to reflect the variation. The term is generally associated with experiments in which the design introduces conditions that directly affect the variation, but may also refer to the design of quasi-experiments, in which natural conditions that influence the variation are selected for observation.
- In its simplest form, an experiment aims at predicting the outcome by introducing a change of the preconditions, which is represented by one or more independent variables, also referred to as "input variables" or "predictor variables." The change in one or more independent variables is generally hypothesized to result in a change in one or more dependent variables, also referred to as "output variables" or "response variables."
- Main concerns in experimental design include the establishment of validity, reliability, and replicability

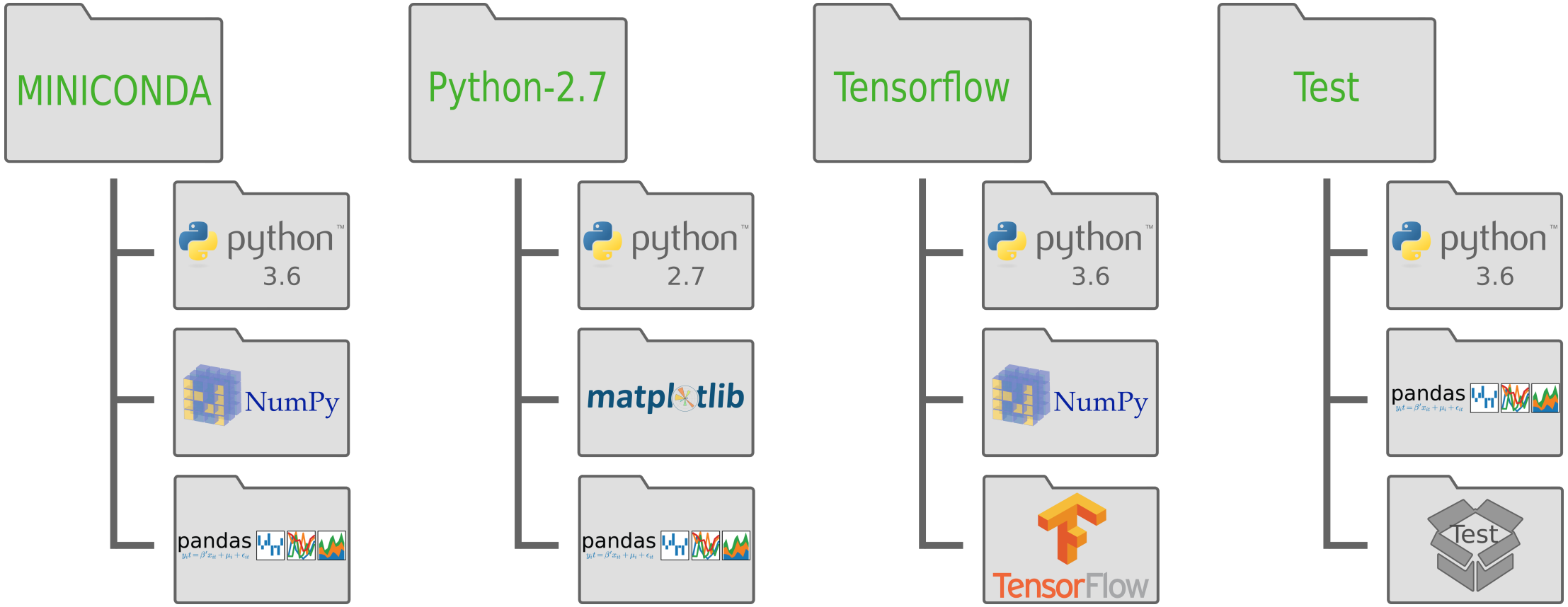
https://en.wikipedia.org/wiki/Design_of_experiments

In Data Mining

- Independent variables include:
  - Features
  - Model
  - Hyperparameters
  - …
  - Computation Environment!

Python Virtual Environments

- **What is a virtual environment?**
- A virtual environment is a self-contained version of Python and specified packages. When you switch to a different virtual environment conda points to that python installation and installed packages.



https://python-for-scientists.readthedocs.io/en/latest/_pages/environments.html

Controlling the Virtual Environment – Package installation

- Installing additional packages not included with the standard Python distribution
- Finding packages published to the Python Package Index (PyPI)
- Managing requirements for your scripts and applications
- Uninstalling packages and their dependencies

  - `pip help`
  - `pip install <package>`
  - `pip list`
  - `pip show <package>`
  - `pip install –r requirements.txt`

https://realpython.com/what-is-pip/

Using Requirements Files

- The pip install command always installs the latest published version of a package, but sometimes, you may want to install a specific version that you know works with your code.

Example Requirements File

Use `pip install -r example-requirements.txt` to install:

```
#
####### example-requirements.txt #######
#
###### Requirements without Version Specifiers ######
nose
nose-cov
beautifulsoup4
#
###### Requirements with Version Specifiers ######
#   See https://www.python.org/dev/peps/pep-0440/#version-specifiers
docopt == 0.6.1             # Version Matching. Must be version 0.6.1
keyring >= 4.1.1            # Minimum version 4.1.1
coverage != 3.5             # Version Exclusion. Anything except version 3.5
Mopidy-Dirble ~= 1.1        # Compatible release. Same as >= 1.1, == 1.*
#
###### Refer to other requirements files ######
-r other-requirements.txt
#
#
###### A particular file ######
./downloads/numpy-1.9.2-cp34-none-win32.whl
http://wxpython.org/Phoenix/snapshot-builds/wxPython_Phoenix-3.0.3.dev1820+49a8884-cp3
#
###### Additional Requirements without Version Specifiers ######
#   Same as 1st section, just here to show that you can put things in any order.
rejected
green
#
```

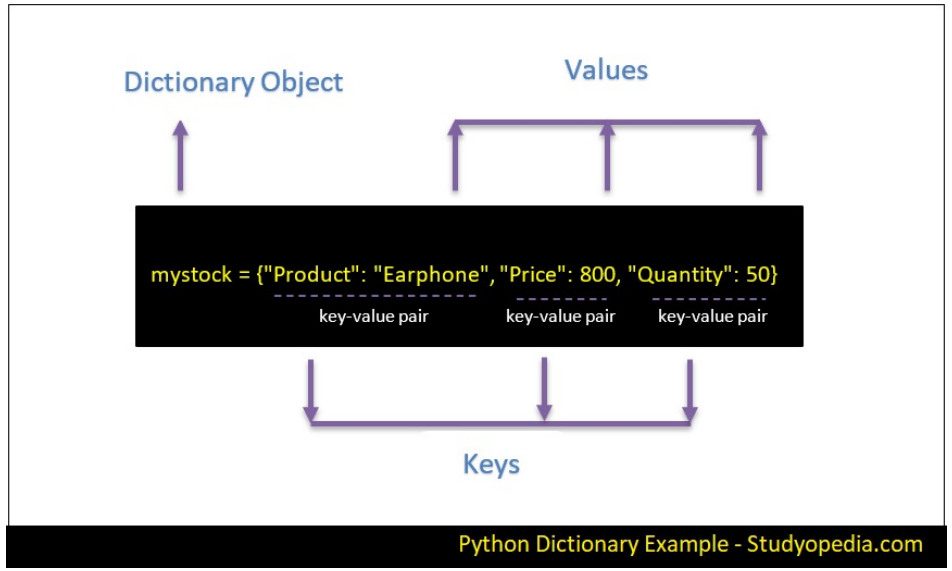
Requirement Specifiers

https://pip.pypa.io/en/stable/reference/pip_install/#requirements-file-format

Python Data Structure Review

Useful built-in Data Types

- Pandas and numpy work great, but sometimes nothing works better than the built-in types!
  - Dictionary
  - Tuple
  - List
- Remember, python loves its iterators, so use those whenever possible



Iterators in Pandas

The two big ones are:

- iterrows (for a dataframe)
- iteritems (for a series)

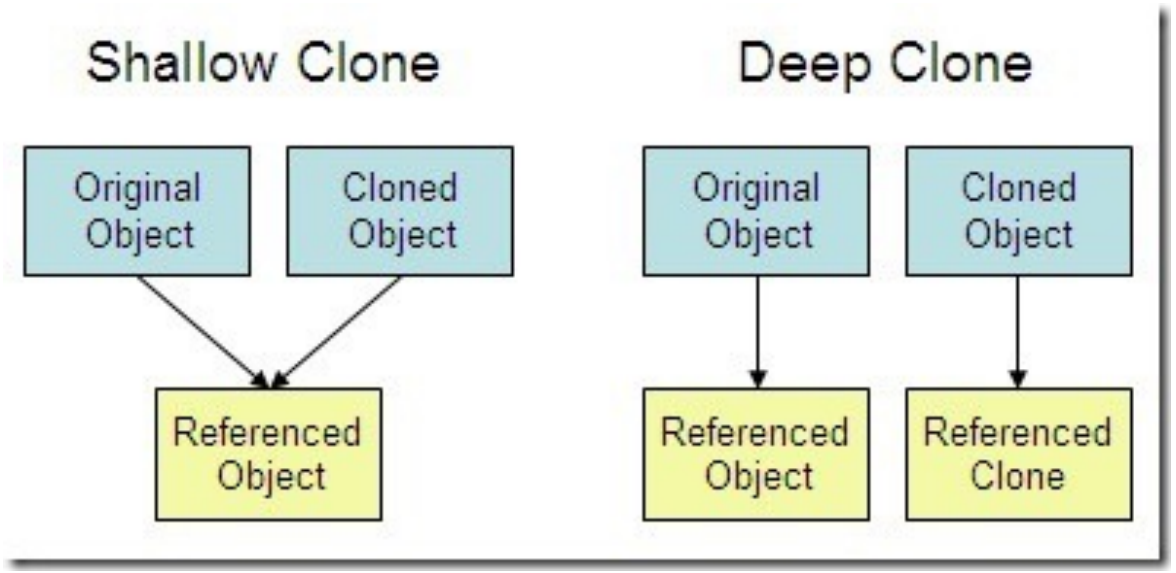
```
import pandas as pd

#create dataframe
df_marks = pd.DataFrame({
    'name': ['apple', 'banana', 'orange', 'mango'],
    'calories': [68, 74, 77, 78]})

#iterate through each row of dataframe
for index, row in df_marks.iterrows():
    print(index, ': ', row['name'], 'has', row['calories'], 'calories.')
```

https://pythonexamples.org/pandas-dataframe-iterate-rows-iterrows/

Shallow vs Deep Copies



In [ ]:
```
!pip install matplotlib
```

In [12]:
```
import matplotlib.pyplot as plt
```

In [16]:
```
plt.plot( [1,2,3],[1,4,9])
plt.xlabel('x')
plt.ylabel('y')
plt.title('This is my first plot')
plt.xlim([0,3.5])
```

Out[16]: (0.0, 3.5)