Section 1 Introduction to Python

Why Python?

Python is Popular

How to measure popularity? It is indeed a data science problem!

- TIOBE: Based on google search results
- PYPL PopularitY: Based on google trends
- GitHut 2.0: Based on Github
- Redmonk: Based on Github+Stack Overflow

Whay about in data science/machine learning community?

A survey conducted by Kaggle

Python is Good

- Stable Learning Curves

An entertaning cartoon from Tobias Hermann

- Scalability of Computation (with the help with other packages)

benchmarking of scientific computation problems

comparison between Numpy and Matlab

- Useful Packages
  - Numpy: Scientific Computing
  - Pandas: Data Analysis and Manipulation
  - Scikit-Learn: Machine Learning
  - Matplotlib: Visualizing Functions/Datasets
  - Seaborn: Visualizing Statistical Data

What is Python

- Official Definition: Python is an **interpreted**, **object-oriented**, high-level programming language with **dynamic semantics**.
- Father of Python: Guido van Rossum, see also the history of python.
- The *ZEN* of python and to be *pythonic*.

In [1]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
```

```
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Comparison between Python and Matlab

A very good introduction on the comparison.

- **Python index starts from 0.** The plausible explanation

In [2]:

```
array = [1,2,3] # create a list
print(array[0]) # First element, in matlab we use array(1)
print(array[-1])# Last element, in matlab we use array(end)
print(list(range(3,9)))
```

```
1
3
[3, 4, 5, 6, 7, 8]
```

- **Indentation is important in Python**. The explanation

In [3]:

```
if 5 > 2:
print("Five is greater than two!") # in matlab we use if-end, in C or R we use curly brackets {}
```

```
  File "<ipython-input-3-0b58fb6e072f>", line 2
    print("Five is greater than two!") # in matlab we use if-end, in C or R we use curly brackets
{}
        ^
IndentationError: expected an indented block
```

In [4]:

```
if 5 > 2:
    print("Five is greater than two!")
```

```
Five is greater than two!
```

How to learn and use Python well?

- Anaconda
- Jupyter Notebook
- Brief guide for Python progamming language: A Byte of Python
- Tutorials on Python Machine Learning: Python Data Science Handbook
- UCI datasets and Kaggle

A Motivating (or more confusing?) Example

please keep this example in mind when we start learning the Python programming systematically from the next lecture.

In [5]:

```
a = 1000
b = a
```

```
b = 1
print(a)
```

```
1000
```

In [9]:

```
a = [1000,1]
b = a
b = [1,1]
print(a)
```

```
[1000, 1]
```

In [10]:

```
a = [1000,1]
b = a
b[0] = 1
print(a)
```

```
[1, 1]
```