Programming with Python

Getting Started with Python.

Kuo, Yao-Jen yaojenkuo@datainpoint.com (mailto:yaojenkuo@datainpoint.com) from DATAINPOINT (https://www.datainpoint.com)

TL; DR

In this lecture, we will talk about the what and why, setting up a Python development environment, and functions.

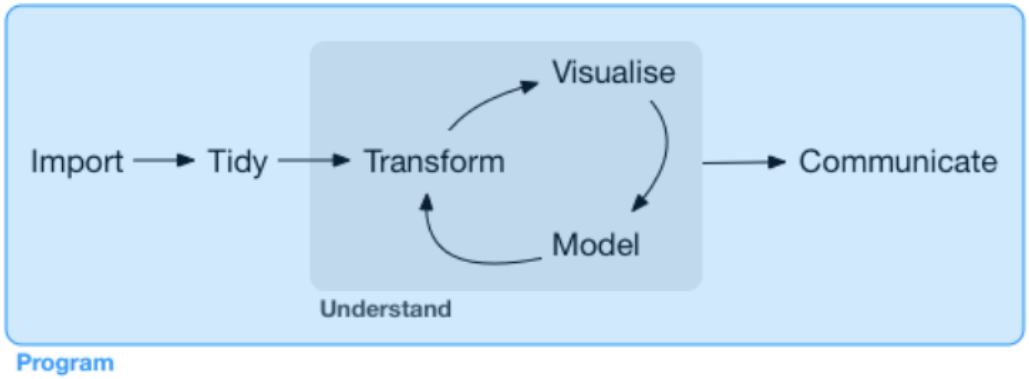
The What and Why

Why do we want to learn Python?

Simply put, to tackle the modern data science applications.

What is modern data science?

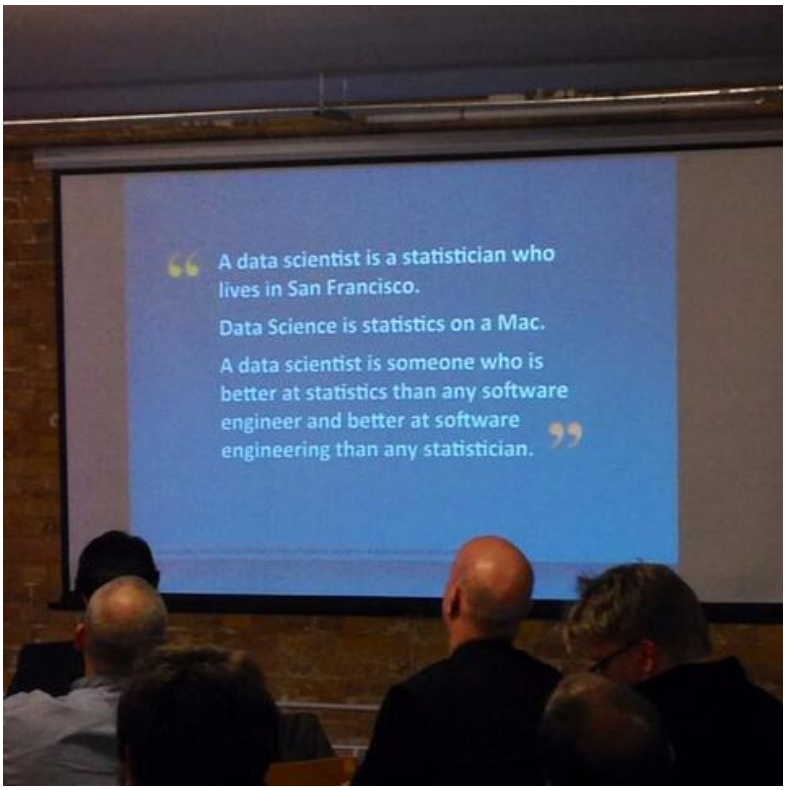
Modern data science is a huge field, it invovles applications and tools like importing, tidying, transformation, visualization, modeling, and communication. Surrounding all these is programming.



Source: [R for Data Science (https://r4ds.had.co.nz/)](https://r4ds.had.co.nz/)

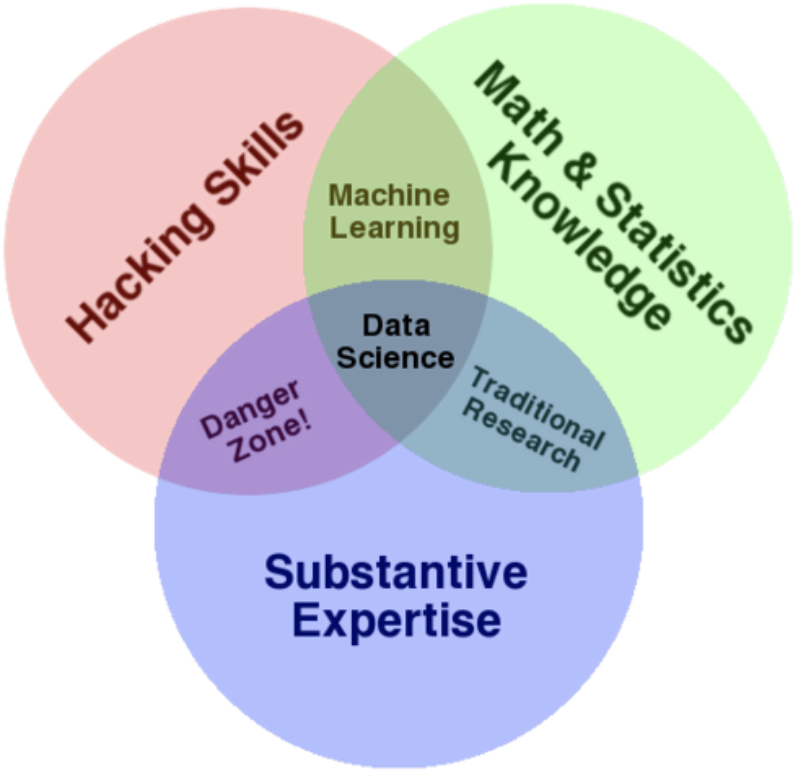
Apparently, Python is gonna be our choice of "Program".

The sarcastic ones



Source: https://twitter.com/cdixon/status/428914681911070720/photo/1 (https://twitter.com/cdixon/status/428914681911070720/photo/1)

The serious ones



Source: [http://drewconway.com/ (http://drewconway.com/)](http://drewconway.com/)

What is Python? the Monty Python British surreal comedy troupe

Source: https://giphy.com/ (https://giphy.com/)

Python: the programming language

> *Python is a clear and powerful object-oriented programming language.*

Source: https://www.python.org/ (https://www.python.org/)

Python: the gigantic snake and its relatives

- Python: the programming language itself
- Anaconda: the data science toolkit solution
- Reticulate: the Python and R interface package

Python is a general-purposed language that is extremely powerful and popular among many applications

- Automation
- Databases
- Analytics
- Graphical user interfaces
- Machine learning
- Web frameworks
- Web scraping
- ...etc.

How does Python do that?

Simply put: third party libraries.

What is a third party library?

A third party library refers to any library where the latest version of the code is not maintained and hosted by neither ourselves nor the official organization, say Python.org (https://www.python.org/).

So which third party library makes Python the go-to choice for data science?

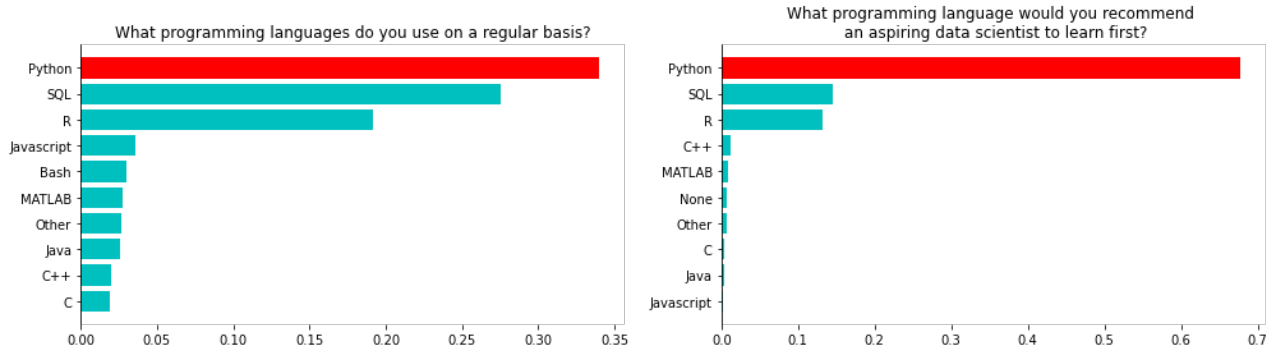
- NumPy
- Pandas
- Matplotlib
- Scikit-Learn
- TensorFlow
- PyTorch
- ...etc.

Why Python?

Let's review 2 questions in [2019 Kaggle ML & DS Survey (https://www.kaggle.com/c/kaggle-survey-2019)]

- Q18: What programming languages do you use on a regular basis?
- Q19: What programming language would you recommend an aspiring data scientist to learn first?

In [4]:
```
# What programming languages do you use on a regular basis?
# What programming language would you recommend an aspiring data scientist to lear
n first?
plot_ans_18_19(ans_18, ans_19)
```



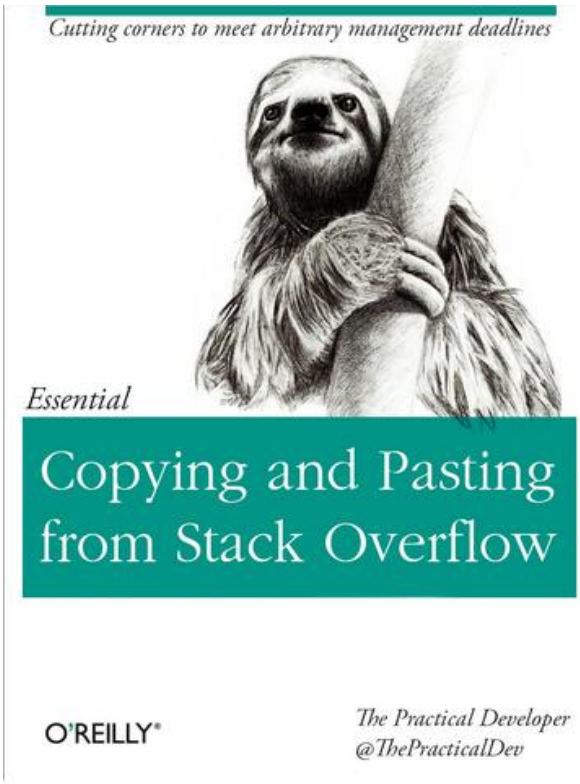
Let's not only explore a snapshot, but also the trend on Stack Overflow (https://stackoverflow.com/)

What is Stack Overflow (https://stackoverflow.com/)?

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.

Trust me, you will love Stack Overflow (https://stackoverflow.com/) so much in the future, short and long-term

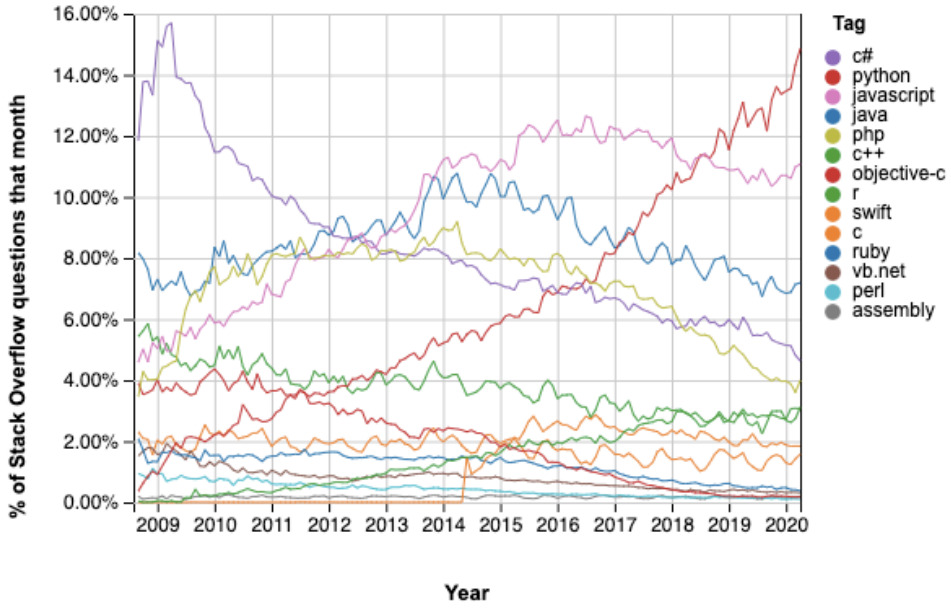
We love it SO MUCH, someone even writes a book for it!



Source: Google Search

Python is growing so fast in the developer community in the last 10 years

Stack Overflow Trends (https://insights.stackoverflow.com/trends?tags=java%2Cc%2Cc%2B%2B%2Cpython%2Cc%23%2Cvb.net%2Cjavascript%2Cassembly%2Cphp%2Cperl%2Cruby%2Cswift%2Cr%2Cobjective-c)



How to utilize Stack Overflow (https://stackoverflow.com/)?

- The first post is question itself
- The second post, if checked "Green", is the answer chose by the initiatior
- The third post, is the answer upvoted by others

Setting up a Python Development Environment

Given Python is a general-purposed programming language, there are variuos ways setting up a Python development environement

It is a mix-and-match challenge

- The operating system
- The Python interpreter
- The Integrated Development Environment
- The package/environment manager

Even among data analysts, everyone has its own favorite flavour

Source: https://giphy.com/ (https://giphy.com/)

The best practice so far for a data analyst is Python interpreter + Jupyter

For novice users who have no prior experience with Python and are willing to install locally

Let's install Python and Jupyter simultaneously: Anaconda (https://www.anaconda.com).

Novice users can also start online without any local installtions

- Google Colab (https://colab.research.google.com/)
- Kaggle Kernel (https://www.kaggle.com/kernels)
- Binder (https://jupyter.org/binder)

Advaned users can make your installation more flexible and customized

- Install a lighter version: Miniconda (https://docs.conda.io/en/latest/miniconda.html).
- Install interpreter from python.org (https://www.python.org/).
- Install jupyter from PyPi (https://pypi.org/).
- Install Visual Studio Code (https://code.visualstudio.com/).

When you are more familiar with Python, learn advanced topics then choose your own flavour

- Path variables
- Library management
- Virtual environemnt
- Deployment

What is a program?

> *A program is a sequence of instructions that specifies how to perform a computation. The computation might be something mathematical, such as solving a system of equations, something symbolic, such as searching for and replacing text in a document, or something graphical, like processing an image.*

Source: Think Julia: How to Think Like a Computer Scientist (https://benlauwens.github.io/ThinkJulia.jl/latest/book.html)

Critical elements of writing/running a Python program

- Text editor
- Python interpreter
- Terminal
- Integrated development environment(IDE)

At the homepage of your Jupyter Notebook Server

- New > Text File
- New > Terminal
- New > Notebook

A few Python programs to try on first

- Hello world!
- Hello John Doe!
- The Zen of Python

Hello world!

In [5]:
```
print("Hello, world!")
```

```
Hello, world!
```

Hello John Doe!

"John Doe" (for males) and "Jane Doe" (for females) are multiple-use names that are used when the true name of a person is unknown or is being intentionally concealed.

Source: https://en.wikipedia.org/wiki/John_Doe (https://en.wikipedia.org/wiki/John_Doe)

In [6]:
```
john_doe = input("Please input your name:")
print("Hello, {}!".format(john_doe))
```

```
Please input your name:Tony
Hello, Tony!
```

Zen of Python

Long time Pythoneer Tim Peters (https://en.wikipedia.org/wiki/Tim_Peters_(software_engineer)) succinctly channels the BDFL's guiding principles for Python's design into 20 aphorisms, only 19 of which have been written down.

In [7]:
```
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!
```

My favorite one would be:

> *Now is better than never.*

How about yours?

Functions

What are `print()` and `input()` in our previous examples?

`print()` and `input()` are so-called **built-in** functions in Python.

What is a function

A function is a named sequence of statements that performs a computation, either mathematical, symbolic, or graphical. When we define a function, we specify the name and the sequence of statements. Later, we can call the function by name.

How do we analyze a function?

- function name
- inputs and arguments, if any
- sequence of statements
- outputs, if any

Take bubble tea shop for instance

Source: Google Search

What is a built-in function?

A pre-defined function, we can call the function by name without defining it.

How many built-in functions are available for us?

- `print()`
- `input()`
- `help()`
- `type()`
- ...etc.

Source: https://docs.python.org/3/library/functions.html (https://docs.python.org/3/library/functions.html)

Get HELP with `help()`

In [8]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

In [9]:
```
help(input)
```

```
Help on method raw_input in module ipykernel.kernelbase:

raw_input(prompt='') method of ipykernel.ipkernel.IPythonKernel instance
    Forward raw_input to frontends

    Raises
    ------
    StdinNotImplentedError if active frontend doesn't support stdin.
```

In [10]:
```
help(type)
```

```
Help on class type in module builtins:

class type(object)
 |  type(object_or_name, bases, dict)
 |  type(object) -> the object's type
 |  type(name, bases, dict) -> a new type
 |
 |  Methods defined here:
 |
 |  __call__(self, /, *args, **kwargs)
 |      Call self as a function.
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __dir__(...)
 |      __dir__() -> list
 |      specialized __dir__ implementation for types
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __instancecheck__(...)
 |      __instancecheck__() -> bool
 |      check if an object is an instance
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signatur
e.
 |
 |  __prepare__(...)
 |      __prepare__() -> dict
 |      used to create the namespace for the class statement
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __sizeof__(...)
 |      __sizeof__() -> int
 |      return memory consumption of the type object
 |
 |  __subclasscheck__(...)
 |      __subclasscheck__() -> bool
 |      check if a class is a subclass
 |
 |  __subclasses__(...)
 |      __subclasses__() -> list of immediate subclasses
 |
 |  mro(...)
 |      mro() -> list
 |      return a type's method resolution order
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __abstractmethods__
```

```
|
|  __dict__
|
|  __text_signature__
|
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __base__ = <class 'object'>
|      The most base type
|
|  __bases__ = (<class 'object'>,)
|
|  __basicsize__ = 864
|
|  __dictoffset__ = 264
|
|  __flags__ = 2148291584
|
|  __itemsize__ = 40
|
|  __mro__ = (<class 'type'>, <class 'object'>)
|
|  __weakrefoffset__ = 368
```

We can also `help()` on `help()`

In [11]:
```
help(help)
```

```
Help on _Helper in module _sitebuiltins object:

class _Helper(builtins.object)
 |  Define the builtin 'help'.
 |
 |  This is a wrapper around pydoc.help that provides a helpful message
 |  when 'help' is typed at the Python interactive prompt.
 |
 |  Calling help() at the Python prompt starts an interactive help session.
 |  Calling help(thing) prints help for the python object 'thing'.
 |
 |  Methods defined here:
 |
 |  __call__(self, *args, **kwds)
 |      Call self as a function.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

Besides built-in functions or library-powered functions, we sometimes need to self-define our own functions

- `def` the name of our function
- `return` the output of our function

The pseudo code layout of a self-defined function

```
def function_name(INPUTS, ARGUMENTS, ...):
    """
    docstring: print documentation when help() is called
    """
    # sequence of statements
    return OUTPUTS
```

An extremely simple function `add`

In [12]:
```
def add(x, y):
    """
    Equivalent to x + y
    """
    return x + y

help(add)
```

```
Help on function add in module __main__:

add(x, y)
    Equivalent to x + y
```

Call the function by name after defining it

In [13]:
```
add(5, 2)
```

Out[13]:
```
7
```

Another extremely simple function `power`

In [14]:
```
def power(x, y):
    """
    Equivalent to x**y
    """
    return x**y

help(power)
```

```
Help on function power in module __main__:

power(x, y)
    Equivalent to x**y
```

Call the function by name after defining it

In [15]:
```
power(5, 2)
```

Out[15]:
```
25
```