EFM Fall 2014, Week 3: More Python

Jason Phang, Allan Zhang

October 22, 2014

Table of Contents

1. IPython Notebooks
2. Assignment 1: Discussion
3. Basic Statistics
4. Functions
5. Libraries

Table of Contents

1 IPython Notebooks

2 Assignment 1: Discussion

3 Basic Statistics

4 Functions

5 Libraries

Starting IPython Notebooks

Another way of running Python code:

Run this in Terminal/Command Prompt

```
$ ipython notebook --pylab=inline
```

Starting IPython Notebooks

Another way of running Python code:

Run this in Terminal/Command Prompt

```
$ ipython notebook --pylab=inline
```

- `ipython` is program you're running

Starting IPython Notebooks

Another way of running Python code:

Run this in Terminal/Command Prompt

```
$ ipython notebook --pylab=inline
```

- `ipython` is program you're running
- `notebook` is the subcommand, telling it to start the IPython notebook server

Starting IPython Notebooks

Another way of running Python code:

Run this in Terminal/Command Prompt

```
$ ipython notebook --pylab=inline
```

- `ipython` is program you're running
- `notebook` is the subcommand, telling it to start the IPython notebook server
- `pylab=inline` is an option, telling IPython that you want to be able to plot in your notebooks

What are IPython Notebooks?

An IPython Notebook

What are IPython Notebooks?

An IPython Notebook

- `.ipynb` files contain a combination of Python code, formatting information, text, and possibly images

What are IPython Notebooks?

An IPython Notebook

- `.ipynb` files contain a combination of Python code, formatting information, text, and possibly images
- Used to present and document code, add in $\LaTeX$

What are IPython Notebooks?

An IPython Notebook

- `.ipynb` files contain a combination of Python code, formatting information, text, and possibly images
- Used to present and document code, add in $\LaTeX$
- Extremely useful for exploratory data analysis

What are IPython Notebooks?

An IPython Notebook

- `.ipynb` files contain a combination of Python code, formatting information, text, and possibly images
- Used to present and document code, add in $\LaTeX$
- Extremely useful for exploratory data analysis
- Each notebook contains a series of **cells**.

What are IPython Notebooks?

An IPython Notebook

- `.ipynb` files contain a combination of Python code, formatting information, text, and possibly images
- Used to present and document code, add in $\LaTeX$
- Extremely useful for exploratory data analysis
- Each notebook contains a series of **cells**.
- Cells can contain Code, Markdown (formatted text) or headings.

What are IPython Notebooks?

How does it work?

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer
    - You'll see it running in the terminal you ran the command in

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer
  - You'll see it running in the terminal you ran the command in
  - If you close it, the server terminates too

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer
  - You'll see it running in the terminal you ran the command in
  - If you close it, the server terminates too
  - Press `Ctrl-C` to close

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer
  - You'll see it running in the terminal you ran the command in
  - If you close it, the server terminates too
  - Press `Ctrl-C` to close
- The server runs sessions of Python in the background, and feeds the output to your notebook

What are IPython Notebooks?

How does it work?

- IPython starts a server process on your computer
  - You'll see it running in the terminal you ran the command in
  - If you close it, the server terminates too
  - Press `Ctrl-C` to close
- The server runs sessions of Python in the background, and feeds the output to your notebook
- Each notebook corresponds to one session - and you can close the notebook and re-open it to continue the same session

IPython Notebook Actions

Basic Operations

IPython Notebook Actions

Basic Operations

- `Ctrl-S` to save (or click the floppy disk)

IPython Notebook Actions

Basic Operations

- `Ctrl-S` to save (or click the floppy disk)
- `Shift-Enter`/`Cmd-Enter` to run a cell

IPython Notebook Actions

Basic Operations

- `Ctrl-S` to save (or click the floppy disk)
- `Shift-Enter`/`Cmd-Enter` to run a cell
  - Equivalent to `%run`ing some code from a file

IPython Notebook Actions

Basic Operations

- `Ctrl-S` to save (or click the floppy disk)
- `Shift-Enter/Cmd-Enter` to run a cell
  - Equivalent to `%run`ing some code from a file
- **kernel** → **Interrupt**/**Restart** to kill/restart a Python process

IPython Notebook Pro-tips

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!
- Cells always print the returned value of the last command, even if you don't ask it to

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!
- Cells always print the returned value of the last command, even if you don't ask it to
- Don't over-rely on notebooks

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!
- Cells always print the returned value of the last command, even if you don't ask it to
- Don't over-rely on notebooks
- Save frequently!

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!
- Cells always print the returned value of the last command, even if you don't ask it to
- Don't over-rely on notebooks
- Save frequently!
- Certain code does not work effectively in Notebooks

IPython Notebook Pro-tips

- Keep your code neat! Organize your cells, keep things in order!
- Cells always print the returned value of the last command, even if you don't ask it to
- Don't over-rely on notebooks
- Save frequently!
- Certain code does not work effectively in Notebooks
    - E.g. Certain kinds of plots

Table of Contents

1. IPython Notebooks
2. Assignment 1: Discussion
3. Basic Statistics
4. Functions
5. Libraries

Assignment 1: Discussion

Assignment 1: Discussion

- See: IPython Notebook

Table of Contents

1. IPython Notebooks
2. Assignment 1: Discussion
3. Basic Statistics
4. Functions
5. Libraries

Really Basic Statistics

We start with some very un-rigorous definitions:

Really Basic Statistics

We start with some very un-rigorous definitions:

- Broadly speaking, a distribution or probability process assigns probabilities to specific outcomes:

Really Basic Statistics

We start with some very un-rigorous definitions:

- Broadly speaking, a distribution or probability process assigns probabilities to specific outcomes:
    - Outcomes can be discrete or continuous, finite or continuous

Really Basic Statistics

We start with some very un-rigorous definitions:

- Broadly speaking, a distribution or probability process assigns probabilities to specific outcomes:
    - Outcomes can be discrete or continuous, finite or continuous
    - E.g. A "coin flip" assigns probability $\frac{1}{2}$ to "Heads" and $\frac{1}{2}$ to tails

Really Basic Statistics

We start with some very un-rigorous definitions:

- Broadly speaking, a distribution or probability process assigns probabilities to specific outcomes:
    - Outcomes can be discrete or continuous, finite or continuous
    - E.g. A "coin flip" assigns probability $\frac{1}{2}$ to "Heads" and $\frac{1}{2}$ to tails
    - E.g. A "height distribution" assigns probabilities of a person having a given height, presumably a higher probability for something like 5"10, compared to 8"10.

Really Basic Statistics

We start with some very un-rigorous definitions:

- Broadly speaking, a distribution or probability process assigns probabilities to specific outcomes:
  - Outcomes can be discrete or continuous, finite or continuous
  - E.g. A "coin flip" assigns probability $\frac{1}{2}$ to "Heads" and $\frac{1}{2}$ to tails
  - E.g. A "height distribution" assigns probabilities of a person having a given height, presumably a higher probability for something like 5"10, compared to 8"10.
- To learn more, take STAT 23400 or STAT 24400

Really Basic Statistics

Really Basic Statistics

- Distributions can be characterized by a probability mass function (discrete outcomes) or probability density function (continuous outcomes)

Really Basic Statistics

- Distributions can be characterized by a probability mass function (discrete outcomes) or probability density function (continuous outcomes)
  - E.g. A Coin Flip (with H=1, T=0):

$$f(x) = \begin{cases} \frac{1}{2}, & \text{if } x = 1. \\ \frac{1}{2}, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Really Basic Statistics

- Distributions can be characterized by a probability mass function (discrete outcomes) or probability density function (continuous outcomes)
    - E.g. A Coin Flip (with H=1, T=0):

$$f(x) = \begin{cases} \frac{1}{2}, & \text{if } x = 1. \\ \frac{1}{2}, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

    - E.g. Two Coin Flips (with outcome of flips being x_1,x_2):

$$f(x_1, x_2) = \begin{cases} \frac{1}{4}, & \text{if } x_1 = 1, x_2 = 1. \\ \frac{1}{4}, & \text{if } x_1 = 1, x_2 = 0. \\ \frac{1}{4}, & \text{if } x_1 = 0, x_2 = 1. \\ \frac{1}{4}, & \text{if } x_1 = 0, x_2 = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Really Basic Statistics

Really Basic Statistics

- E.g. An unfair coin:

$$f(x) = \begin{cases} \frac{1}{4}, & \text{if } x = 1. \\ \frac{3}{4}, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Really Basic Statistics

- E.g. An unfair coin:

$$f(x) = \begin{cases} \frac{1}{4}, & \text{if } x = 1. \\ \frac{3}{4}, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

- E.g. Uniform distribution (for outcome between 0 and 10):

$$f(x) = \begin{cases} \frac{1}{10}, & \text{if } 0 \leq x \leq 10\ . \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Really Basic Statistics

- E.g. An unfair coin:

$$f(x) = \begin{cases} \frac{1}{4}, & \text{if } x = 1. \\ \frac{3}{4}, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

- E.g. Uniform distribution (for outcome between 0 and 10):

$$f(x) = \begin{cases} \frac{1}{10}, & \text{if } 0 \leq x \leq 10\ . \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

  - Note that for continuous distributions, the value of the function does not really correspond to a "probability", but rather a "probability density"

Really Basic Statistics

Really Basic Statistics

- Standard Normal distribution (all real numbers):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} \tag{5}$$

Really Basic Statistics

- Standard Normal distribution (all real numbers):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} \tag{5}$$

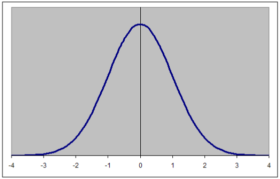
    - This looks horrible, but has really nice properties

Really Basic Statistics

- Standard Normal distribution (all real numbers):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} \tag{5}$$

  - This looks horrible, but has really nice properties

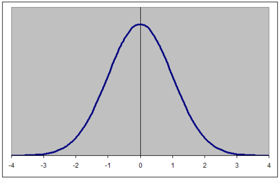


Really Basic Statistics

- Standard Normal distribution (all real numbers):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}} \tag{5}$$

  - This looks horrible, but has really nice properties



- Importantly, PMFs must sum to 1, and PDFs must integrate to 1, so that they can be considered ”probabilities”

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

- A **statistic** is some function of an *empirical* data set (or alternatively, a theoretical distribution)

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

- A **statistic** is some function of an *empirical* data set (or alternatively, a theoretical distribution)
- Mean (or average):

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{6}$$

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

- A **statistic** is some function of an *empirical* data set (or alternatively, a theoretical distribution)
- Mean (or average):

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{6}$$

- Variance:

$$s^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_i - \bar{x})^2 \tag{7}$$

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

- A **statistic** is some function of an *empirical* data set (or alternatively, a theoretical distribution)
- Mean (or average):

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{6}$$

- Variance:

$$s^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_i - \bar{x})^2 \tag{7}$$

  - Standard deviation $= \sqrt{s^2}$

Really Basic Statistics

Suppose we have some data: a list of numbers $[x_1 \cdots x_N]$

- A **statistic** is some function of an *empirical* data set (or alternatively, a theoretical distribution)
- Mean (or average):

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{6}$$

- Variance:

$$s^2 = \frac{1}{N-1} \sum_{n=1}^{N} (x_i - \bar{x})^2 \tag{7}$$

  - Standard deviation $= \sqrt{s^2}$
  - You might sometimes see a similar definition except using σ^2 and having the fraction $\frac{1}{N}$ instead. Don't worry about it - it has to do with the difference between an observed and theoretical data set, and usually the numerical differences are small.

Really Basic Statistics

Really Basic Statistics

Okay, why the math?

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution
- Understanding the math behind those distributions can give us a lot of information

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution
- Understanding the math behind those distributions can give us a lot of information
- E.g. The Normal Distribution is a family of distributions

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution
- Understanding the math behind those distributions can give us a lot of information
- E.g. The Normal Distribution is a family of distributions
  - Parameterized by two parameters μ and σ^2

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution
- Understanding the math behind those distributions can give us a lot of information
- E.g. The Normal Distribution is a family of distributions
  - Parameterized by two parameters μ and σ^2

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

Really Basic Statistics

Okay, why the math?

- That was really, dry. Please stop.
- Most models (economic, financial, statistical) assume our data comes from some (unknown) distribution
- Understanding the math behind those distributions can give us a lot of information
- E.g. The Normal Distribution is a family of distributions
  - Parameterized by two parameters μ and σ^2

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

  - The mean is μ and the variance is σ^2

Table of Contents

1. IPython Notebooks
2. Assignment 1: Discussion
3. Basic Statistics
4. Functions
5. Libraries

Functions

Functions

Back to lovely, lovely coding!

Functions

Functions

Back to lovely, lovely coding!

Functions

- A function is a pre-written set of instruction that you will run repeatedly

Functions

Back to lovely, lovely coding!

Functions

- A function is a pre-written set of instruction that you will run repeatedly
- In some ways similar to the mathematical definition of a function, in that it *often* takes in some input (known as **arguments**), and it *often* gives an output (known as **return**ing a value)

Functions

Back to lovely, lovely coding!

Functions

- A function is a pre-written set of instruction that you will run repeatedly
- In some ways similar to the mathematical definition of a function, in that it *often* takes in some input (known as **arguments**), and it *often* gives an output (known as **return**ing a value)
- Functions *will* be the most important thing you learn in programming

Functions

Functions

Functions

Functions

- Start a definition using **def** and some parenthesis

Functions

Functions

- Start a definition using **def** and some parenthesis
- Indent function code

Functions

Functions

- Start a definition using **def** and some parenthesis
- Indent function code
- Call a function via the function name, and paranthesis

Functions

Functions

- Start a definition using **def** and some parenthesis
- Indent function code
- Call a function via the function name, and paranthesis

Simple function

```
def greet_user():
    print "Hello, friend!"

greet_user()
```

Functions

Functions

- Start a definition using **def** and some parenthesis
- Indent function code
- Call a function via the function name, and paranthesis

Simple function

```
def greet_user():
    print "Hello, friend!"

greet_user()
```

Output

```
Hello, friend!
```

Functions

Functions

Functions

Functions

- Here's a function with an argument, and a return value

Functions

Functions

- Here's a function with an argument, and a return value

Simple function

```
def square_this_number(x):
    return x*x

print square_this_number(10):
```

Functions

Functions

- Here's a function with an argument, and a return value

Simple function

```
def square_this_number(x):
    return x*x


print square_this_number(10):
```

Output

```
100
```

Functions

Functions - Some Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user()
```

Functions

Functions - Some Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user()
```

Output

```
Hello, friend!
None
```

Functions

Functions - Some Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user()
```

Output

```
Hello, friend!
None
```

- Function gets called and prints ”Hello, friend!”

Functions

Functions - Some Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user()
```

Output

```
Hello, friend!
None
```

- Function gets called and prints ”Hello, friend!”
- Then Python tries to print the output value of the function, but there is none!

Functions

Functions - Some Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user()
```

Output

```
Hello, friend!
None
```

- Function gets called and prints ”Hello, friend!”
- Then Python tries to print the output value of the function, but there is none!
  - Yes, Python has an *object* called None, which is the default returned value

Functions

Functions - More Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user
```

Functions

Functions - More Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user
```

Output

```
<function greet_user at 0x10bbabc08>
```

Functions

Functions - More Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user
```

Output

```
<function greet_user at 0x10bbabc08>
```

- You're not calling the function, you're printing it!

Functions

Functions - More Subtlety

Simple function

```
def greet_user():
    print "Hello, friend!"

print greet_user
```

Output

```
<function greet_user at 0x10bbabc08>
```

- You're not calling the function, you're printing it!
- Python goes and prints the function as an object

Functions

Functions - Multiple Arguments

Functions

Functions - Multiple Arguments

- You can supply multiple arguments

Simple function

```
def add_numbers(x,y):
    return x+y

print add_numbers(5,10)
```

Functions

Functions - Multiple Arguments

- You can supply multiple arguments

Simple function

```
def add_numbers(x,y):
    return x+y

print add_numbers(5,10)
```

Output

```
15
```

Functions

Function - A more complex example

Simple function

```
def find_maximum(ls):
    current_max = ls[0]
    for i in ls[1:]:
     if i > current_max:
         current_max = i
    return current_max

print find_maximum([5,3,1,2,3,1,4])
```

Functions

Function - A more complex example

Simple function

```
def find_maximum(ls):
    current_max = ls[0]
    for i in ls[1:]:
     if i > current_max:
         current_max = i
    return current_max

print find_maximum([5,3,1,2,3,1,4])
```

Output

```
5
```

Functions

Function - They are not type-safe

Simple function

```
def find_maximum(ls):
    current_max = ls[0]
    for i in ls[1:]:
     if i > current_max:
         current_max = i
    return current_max

print find_maximum(3)
```

Functions

Function - They are not type-safe

Simple function

```
def find_maximum(ls):
    current_max = ls[0]
    for i in ls[1:]:
     if i > current_max:
         current_max = i
    return current_max

print find_maximum(3)
```

Output

```
Some long error
```

Functions

Function - They are not type-safe

Simple function

```
def find_maximum(ls):
    current_max = ls[0]
    for i in ls[1:]:
     if i > current_max:
         current_max = i
    return current_max

print find_maximum(3)
```

Output

```
Some long error
```

- Functions don't check for the type of your input (unlike languages like C or Java), YOU (or your code) have to do that!

Functions

A function that literally does nothing

Simple function

```
def do_nothing():
    pass

do_nothing()
```

Functions

A function that literally does nothing

Simple function

```
def do_nothing():
    pass

do_nothing()
```

Output

Functions

A function as an object

Functions

A function as an object

- You can also pass a function as an argument!

Simple function

```
def run_function_on_number(n,f):
    return f(n)

run_function_on_number(2.0,sqrt)
```

Functions

A function as an object

- You can also pass a function as an argument!

Simple function

```
def run_function_on_number(n,f):
    return f(n)


run_function_on_number(2.0,sqrt)
```

Output

```
1.4142135623730951
```

Functions

Functions you already encountered

Functions

Functions you already encountered

- `sum`

Functions

Functions you already encountered

- `sum`
- `range` - This could take either 1 or 2 arguments, isn't that funny?

Functions

Functions you already encountered

- `sum`
- `range` - This could take either 1 or 2 arguments, isn't that funny?
- `len`

Functions

Functions you already encountered

- `sum`
- `range` - This could take either 1 or 2 arguments, isn't that funny?
- `len`
- `open`

Table of Contents

1. IPython Notebooks

2. Assignment 1: Discussion

3. Basic Statistics

4. Functions

5. Libraries

Libraries

Libraries

Libraries

Libraries

- Libraries are prepackaged code written by other people!

Libraries

Libraries

- Libraries are prepackaged code written by other people!
- You can install libraries via `pip`, as we did on the first day

Libraries

Libraries

- Libraries are prepackaged code written by other people!
- You can install libraries via `pip`, as we did on the first day
- Python comes with some of its own built-in libraries. Canopy comes with TONS of libraries pre-installed.

Libraries

Libraries

- Libraries are prepackaged code written by other people!
- You can install libraries via `pip`, as we did on the first day
- Python comes with some of its own built-in libraries. Canopy comes with TONS of libraries pre-installed.
- To use code from a library, you have to **import** it.

Libraries

Importing Libraries

Libraries

Importing Libraries

- To import a library, literally type `import LIBRARY`

Libraries

Importing Libraries

- To import a library, literally type `import LIBRARY`

Import

```
import math
print math.e
```

Libraries

Importing Libraries

- To import a library, literally type `import LIBRARY`

Import

```
import math
print math.e
```

Output

```
2.71828182846
```

Libraries

Importing Libraries

Libraries

Importing Libraries

- There are multiple ways to import a library

Libraries

Importing Libraries

- There are multiple ways to import a library
- Which method depends on how you want to name and refer to things

Libraries

Importing Libraries

- There are multiple ways to import a library
- Which method depends on how you want to name and refer to things

Direct Import

```
import math
print math.e
```

Libraries

Importing Libraries

- There are multiple ways to import a library
- Which method depends on how you want to name and refer to things

Direct Import

```
import math
print math.e
```

Import as

```
import math as my_mathematical_library
print my_mathematical_library.e
```

Libraries

Importing Libraries

- There are multiple ways to import a library
- Which method depends on how you want to name and refer to things

Direct Import

```
import math
print math.e
```

Import as

```
import math as my_mathematical_library
print my_mathematical_library.e
```

From .. Import

```
from math import e
print e
```

Libraries

Library - Random

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Random

```
import random

## Returns a random integer between a low and high number
random.randint(0,10)
```

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Random

```
import random

## Returns a random integer between a low and high number
random.randint(0,10)
```

Output

```
2
```

Libraries

Library - Random

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Random

```
import random

## Returns a random number between 0 and 1
random.random()
```

Libraries

Library - Random

- The **random** library provides functions that generate random numbers

Random

```
import random

## Returns a random number between 0 and 1
random.random()
```

Output

```
0.5221620691621208
```

Libraries

Library - Random

Libraries

Library - Random

- Computers usually can't generate truly random numbers, instead they generate "pseudo-random" numbers, based on some chaotic (close to unpredictable) process

Libraries

Library - Random

- Computers usually can't generate truly random numbers, instead they generate "pseudo-random" numbers, based on some chaotic (close to unpredictable) process
- Sometimes, for the purpose of reproducability, we want to obtain the same set of random numbers every time we run our code

Libraries

Library - Random

- Computers usually can't generate truly random numbers, instead they generate ”pseudo-random” numbers, based on some chaotic (close to unpredictable) process
- Sometimes, for the purpose of reproducability, we want to obtain the same set of random numbers every time we run our code

Random

```
import random

random.seed(123)
random.random()
```

Libraries

Library - Random

- Computers usually can't generate truly random numbers, instead they generate "pseudo-random" numbers, based on some chaotic (close to unpredictable) process
- Sometimes, for the purpose of reproducability, we want to obtain the same set of random numbers every time we run our code

Random

```
import random

random.seed(123)
random.random()
```

Output

```
0.052363598850944326
```

Libraries

Library - Math

Libraries

Library - Math

- Contains most common math functions

Libraries

Library - Math

- Contains most common math functions

Random

```
import math

## Returns cosine of am angle (in radians)
math.sin(math.pi/3)
```

Libraries

Library - Math

- Contains most common math functions

Random

```
import math

## Returns cosine of am angle (in radians)
math.sin(math.pi/3)
```

Output

```
0.8660254037844386
```

Libraries

Extra notes on Libraries

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
    - Google!

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
  - Google!
  - After you've imported it, type `dir(LIBRARY)`

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
  - Google!
  - After you've imported it, type `dir(LIBRARY)`
  - (In IPython) After you've imported it, type LIBRARY. and press TAB

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
    - Google!
    - After you've imported it, type `dir(LIBRARY)`
    - (In IPython) After you've imported it, type LIBRARY. and press TAB
- **IPython pre-imports a whole bunch of libraries**

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
  - Google!
  - After you've imported it, type `dir(LIBRARY)`
  - (In IPython) After you've imported it, type LIBRARY. and press TAB
- **IPython pre-imports a whole bunch of libraries**
  - E.g. `math`, `numpy`, `pylab` etc

Libraries

Extra notes on Libraries

- Python's greatest strength is that it has a wealth of powerful and easy-to-use libraries
- To find out what's in a library
    - Google!
    - After you've imported it, type `dir(LIBRARY)`
    - (In IPython) After you've imported it, type LIBRARY. and press TAB
- **IPython pre-imports a whole bunch of libraries**
    - E.g. `math`, `numpy`, `pylab` etc
    - Sometimes your code written in IPython won't run elsewhere, because you need to explicitly import the libraries yourself

Plotting

Plotting

Plotting

Plotting

- IPython pre-imports **matplotlib**, a plotting library for Python

Plotting

Plotting

- IPython pre-imports **matplotlib**, a plotting library for Python
- As such, you can use the **plot** function without importing anything, even though normally you would need to type
  `from matplotlib.pyplot import plot`

Plotting

Plotting

- IPython pre-imports **matplotlib**, a plotting library for Python
- As such, you can use the **plot** function without importing anything, even though normally you would need to type `from matplotlib.pyplot import plot`
- **plot** creates a lineplot, and can take either a list of *y*-values, or a list of *x*- and a list of *y*-values

Plotting

Plotting

- IPython pre-imports **matplotlib**, a plotting library for Python
- As such, you can use the **plot** function without importing anything, even though normally you would need to type `from matplotlib.pyplot import plot`
- **plot** creates a lineplot, and can take either a list of *y*-values, or a list of *x*- and a list of *y*-values

Random

```
from matplotlib.pyplot import plot

plot(range(10))
plot(range(10),[1,-1,2,-2,3,-3,4,-4,5,-5])
```

Plotting

Plotting

Plotting

Plotting

- The **plot** function has a whole bunch of options (optional arguments). Try some of the following!

Plotting

Plotting

- The **plot** function has a whole bunch of options (optional arguments). Try some of the following!
- If you run multiple **plot** function in the same IPython cell, they will plot on the same graph

Plotting

Plotting

- The **plot** function has a whole bunch of options (optional arguments). Try some of the following!
- If you run multiple **plot** function in the same IPython cell, they will plot on the same graph

Random

```
from matplotlib.pyplot import plot, title, xlabel, ylabel

ls1 = range(10)
ls2 = [1,-1,2,-2,3,-3,4,-4,5,-5]

plot(ls,"xk")
plot(ls1,ls2,"--o")
xlabel("X-axis!")
xlabel("Y-axis!")
plt.title("Wow a title!")
```

Plotting

Plotting

Plotting

Plotting

- You can also do a histogram plot

Plotting

Plotting

- You can also do a histogram plot
- You can simply supply a list of elements, and matplotlib and numpy will automatically bin the elements for you

Plotting

Plotting

- You can also do a histogram plot
- You can simply supply a list of elements, and matplotlib and numpy will automatically bin the elements for you

Random

```
from matplotlib.pyplot import plot, title, xlabel, ylabel, his
import random

normal_vals = [random.normalvariate(0,1) for i in range(10000)
hist(normal_vals)
```

Bye!

To-do!

Bye!

To-do!

- New assignment for the week!

Bye!

To-do!

- New assignment for the week!
- Questions are short - it's week 4 after all :(

Bye!

To-do!

- New assignment for the week!
- Questions are short - it's week 4 after all :(

Next Week

Bye!

To-do!

- New assignment for the week!
- Questions are short - it's week 4 after all :(

Next Week

- Portfolio Theory!