Learn Data Science in Python

Introduction and Basics

Feb 4, 2017

Welcome!

The objectives of this workshop:

Get involved with Data Science

Learn the basics through full process

Gain flexibility in your own career

Who Am I?

Jimmy Lin is …

A Graduate student at Business Analytics

A Taiwanese Graduated from Shanghai JTU

A New Hire at KPMG Cyber in 2017

Why Do We Code?

Why Do We Code?

“Build a streamline to get the job done repeatedly.”

Why Do We Code?

"Build a $\underbrace{\text{streamline}}_{\text{scalable}}$ to get the job done $\underbrace{\text{repeatedly}}_{\text{reproducible}}$."

Why Do We Code?

"Build a streamline to get the job done repeatedly."

scalable reproducible

$$\begin{tikzcd}\boxed{\text{Import Data}} \arrow[r] & \boxed{\text{Process Data}} \arrow[r] & \boxed{\text{Export Info}}\end{tikzcd}$$

Why Do We Code?

"Build a <u>streamline</u> to get the job done <u>repeatedly</u>."

scalable reproducible

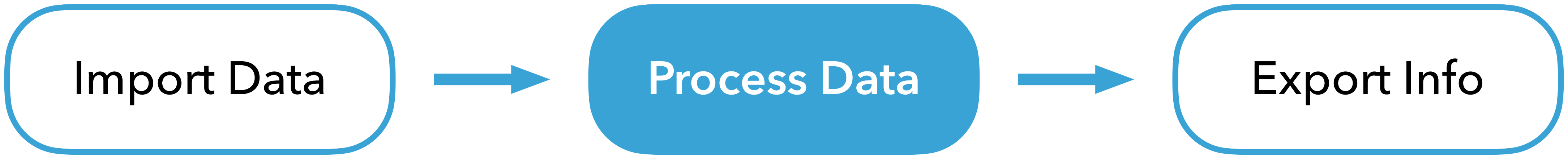
$$\begin{tikzcd}\boxed{\text{Import Data}} \arrow[r] & \boxed{\text{Process Data}} \arrow[r] & \boxed{\text{Export Info}}\end{tikzcd}$$

- Basics
- Importing
- Speculation

Why Do We Code?

"Build a streamline to get the job done repeatedly."

scalable reproducible



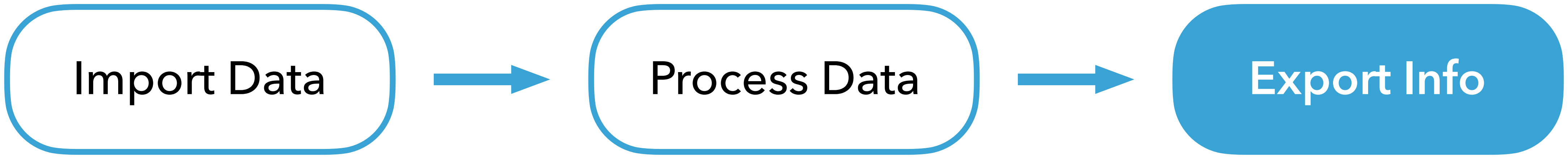
- Basics
- Importing
- Speculation

- Manipulation
- Loop & function
- Basic analysis

Why Do We Code?

"Build a streamline to get the job done repeatedly."

scalable — reproducible



- Basics
- Importing
- Speculation

- Manipulation
- Loop & function
- Basic analysis

- Visualization
- Useful tools
- Sharing

Why Do We Code?

“Build a $\underset{\text{scalable}}{\underline{\text{streamline}}}$ to get the job done $\underset{\text{reproducible}}{\underline{\text{repeatedly}}}$.”

$$\begin{tikzcd}
\boxed{\text{Import Data}} \arrow[r] & \boxed{\text{Process Data}} \arrow[r] & \boxed{\text{Export Info}}
\end{tikzcd}$$

• Basics	• Manipulation	• Visualization
• Importing	• Loop & function	• Useful tools
• Speculation	• Basic analysis	• Sharing

With labs, applications and online resources!

Setup & Basics

Set up the environment for Python and install necessary packages for the future courses.

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)
2. Install **Anaconda** for:
   - IPython
   - Numpy
   - Pandas
   - Matplotlib
   - Scikit-learn

Python | **Anaconda**

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)
2. Install **Anaconda** for:
   - IPython
   - Numpy
   - Pandas
   - Matplotlib
   - Scikit-learn

Python	Anaconda

+ Friendly interface

+ Powerful features

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)
2. Install **Anaconda** for:
   - IPython
   - Numpy
   - Pandas
   - Matplotlib
   - Scikit-learn
3. Test whether everything works well

Python | **Anaconda**

+ Friendly interface
+ Powerful features

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)
2. Install **Anaconda** for:
   - IPython
   - Numpy
   - Pandas
   - Matplotlib
   - Scikit-learn
3. Test whether everything works well
   - In the **terminal**:

```
ipython
```

```
jupyter notebook
```

Python | Anaconda

+ Friendly interface
+ Powerful features

Set Up The Environment

1. Install or upgrade to **Python 3** (recommended)
2. Install **Anaconda** for:
   - IPython
   - Numpy
   - Pandas
   - Matplotlib
   - Scikit-learn
3. Test whether everything works well

Python | Anaconda

+ Friendly interface
+ Powerful features

- In the **terminal**:

```
ipython
```

```
jupyter notebook
```

- In **Python** or **IPython**:

```
import numpy as np
```

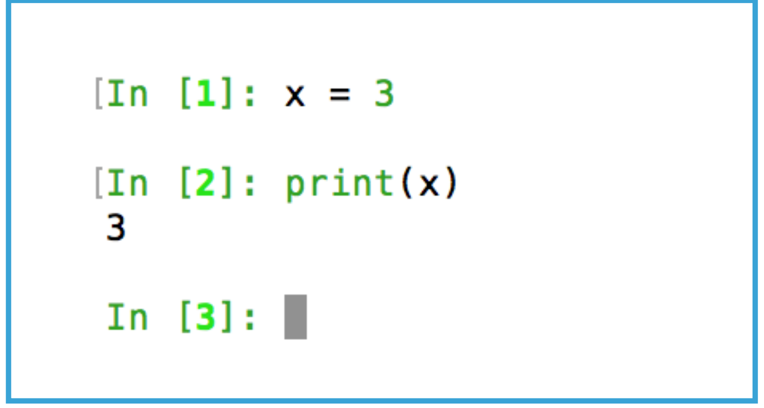
```
import pandas as pd
```

```
import matplotlib
```

```
import sklearn
```

Choices Of Editor

Console

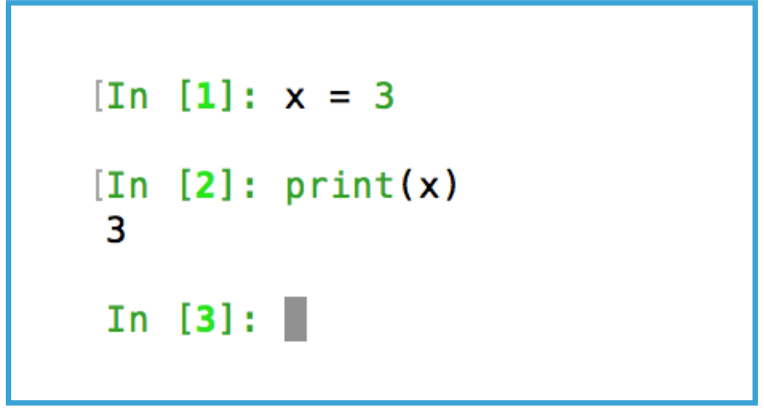


```
[In [1]: x = 3

[In [2]: print(x)
 3

 In [3]: 
```

- Terminal
- Limit in progress saving

Choices Of Editor

Console

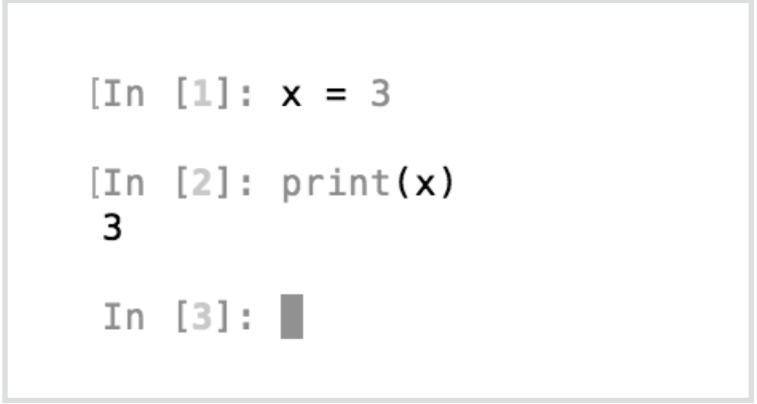


```
[In [1]: x = 3

[In [2]: print(x)
 3

 In [3]: 
```

- Terminal
- Limit in progress saving
- **Simple testing**

Choices Of Editor

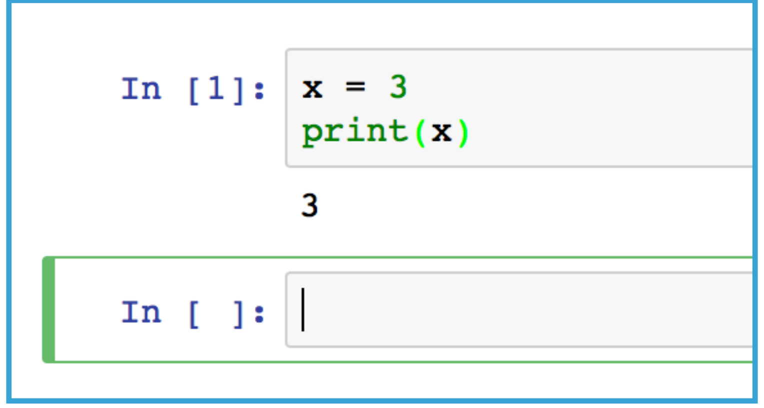
Console



```
[In [1]: x = 3

[In [2]: print(x)
 3

 In [3]: █
```

- Terminal
- Limit in progress saving
- **Simple testing**

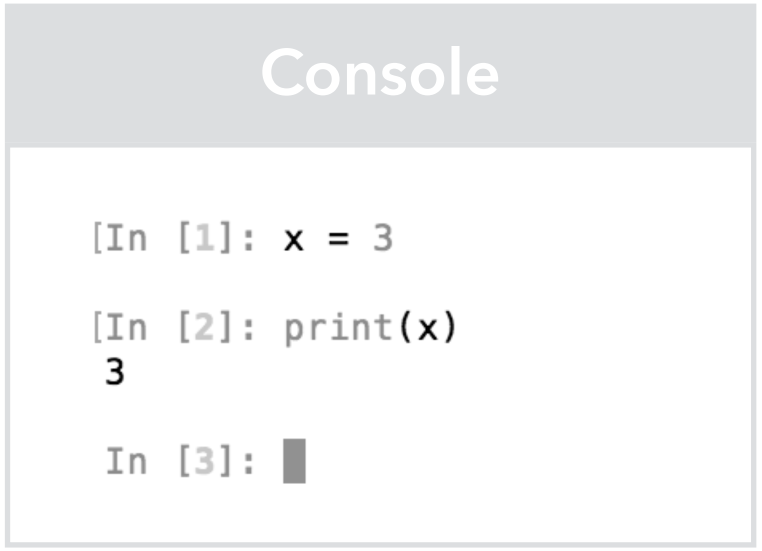
Notebook



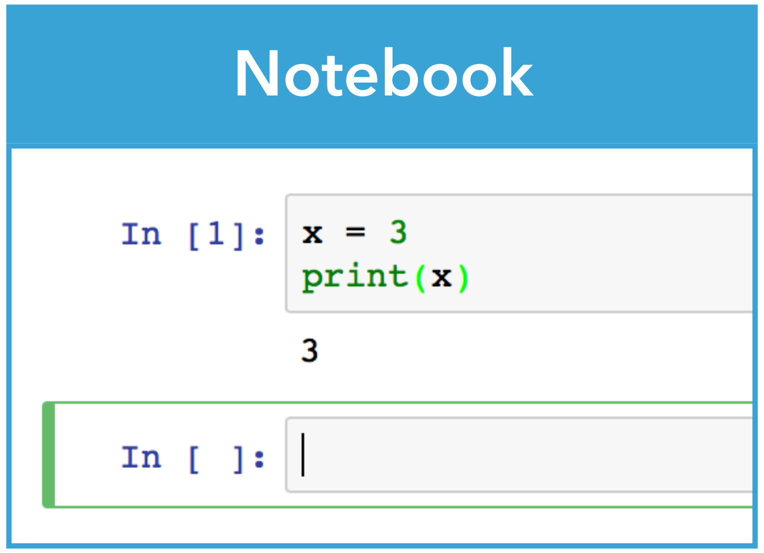
```
In [1]: x = 3
        print(x)

        3

In [ ]: |
```

- Web browser
- Clear input and result for each step

Choices Of Editor


Console

```
[In [1]: x = 3

[In [2]: print(x)
 3

 In [3]: █
```


Notebook

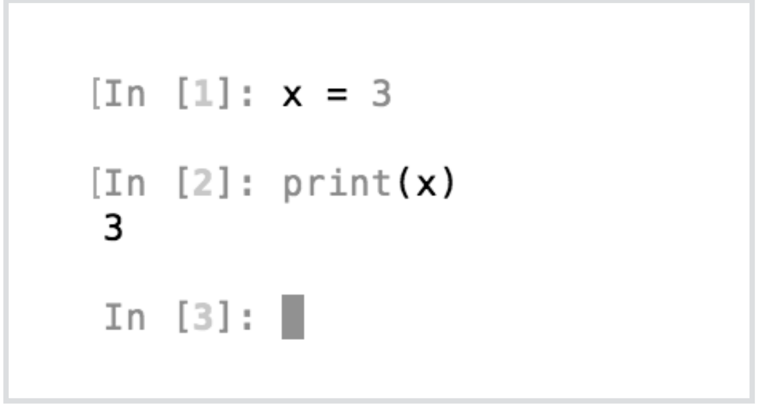
```
In [1]: x = 3
        print(x)

        3

In [ ]: |
```

- Terminal
- Limit in progress saving
- **Simple testing**

- Web browser
- Clear input and result for each step
- **Demonstration**

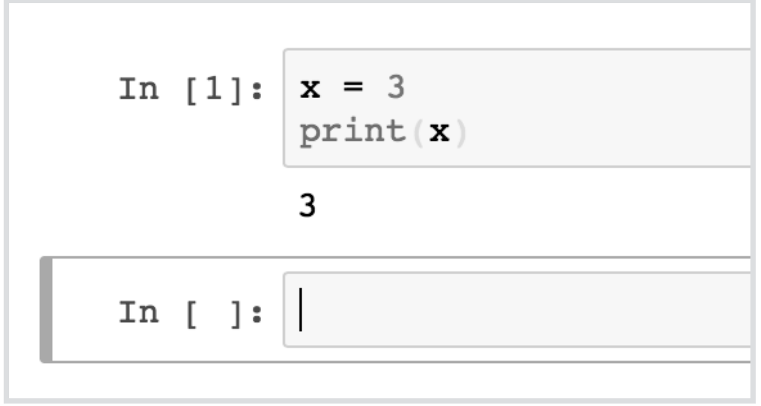
Choices Of Editor

Console



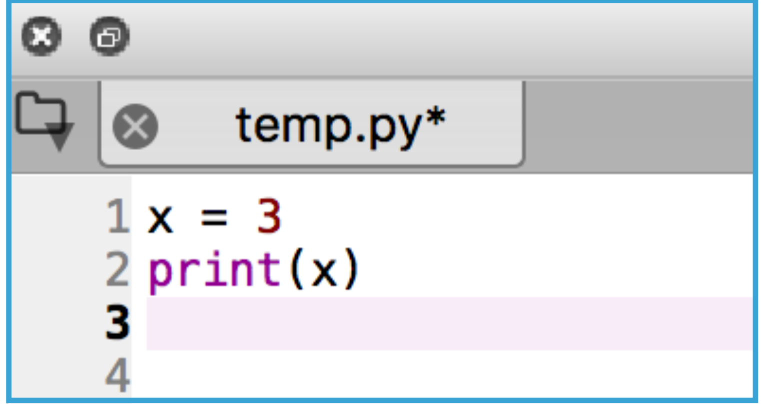
```
In [1]: x = 3

In [2]: print(x)
3

In [3]:
```

Notebook



```
In [1]: x = 3
        print(x)

        3

In [ ]:
```

Spyder, etc.



```
x = 3
print(x)
```

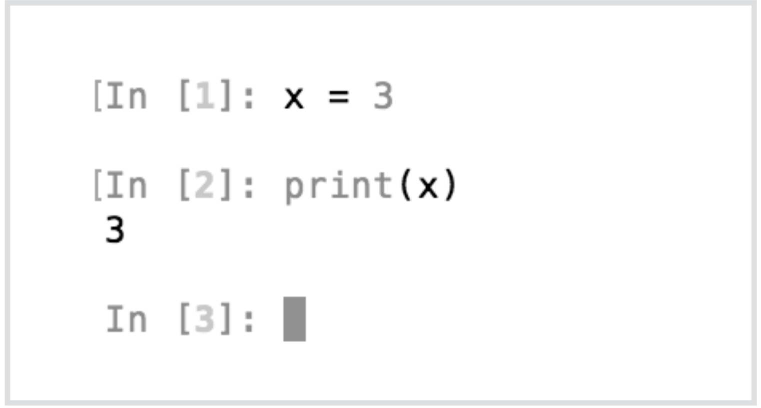
- Terminal
- Limit in progress saving
- **Simple testing**

- Web browser
- Clear input and result for each step
- **Demonstration**

- Standalone App
- Convenient panels for multiple purposes

Choices Of Editor

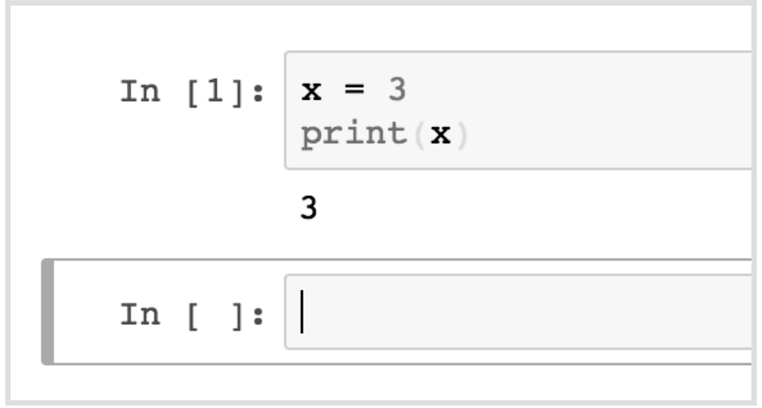
Console



```
In [1]: x = 3

In [2]: print(x)
3

In [3]:
```

- Terminal
- Limit in progress saving
- **Simple testing**

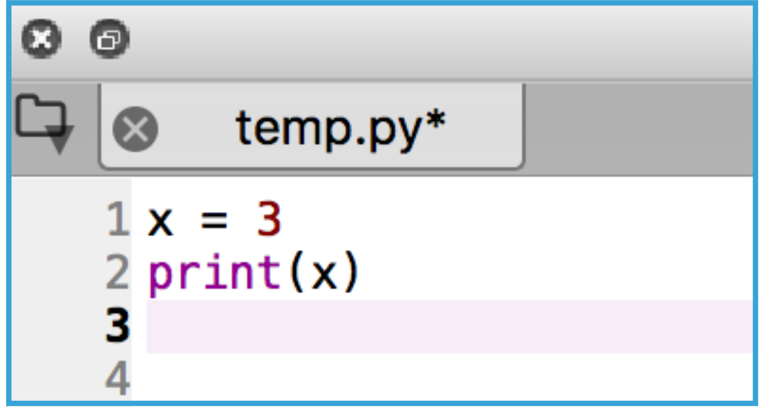
Notebook



```
In [1]: x = 3
        print(x)

        3

In [ ]:
```

- Web browser
- Clear input and result for each step
- **Demonstration**

Spyder, etc.

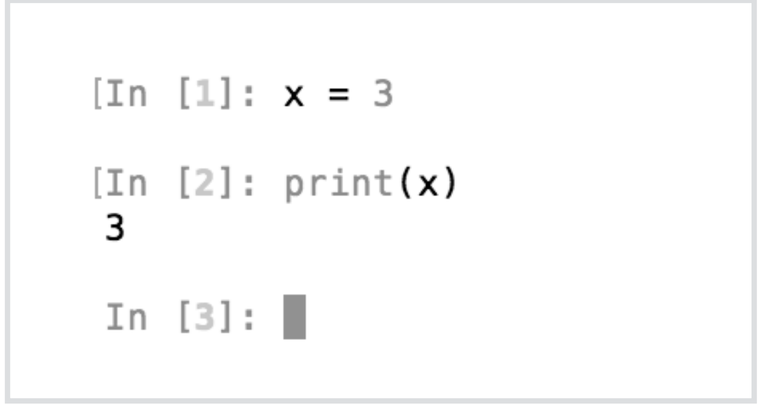


```
x = 3
print(x)
```

- Standalone App
- Convenient panels for multiple purposes
- **Development**

Choices Of Editor

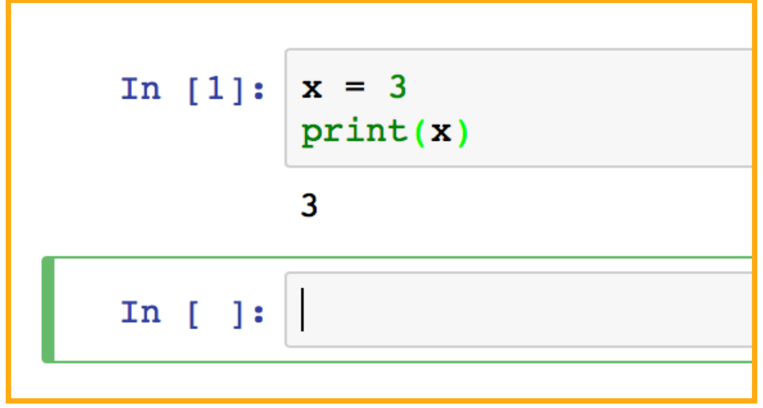
Console



```
In [1]: x = 3

In [2]: print(x)
3

In [3]:
```

- Terminal
- Limit in progress saving
- **Simple testing**

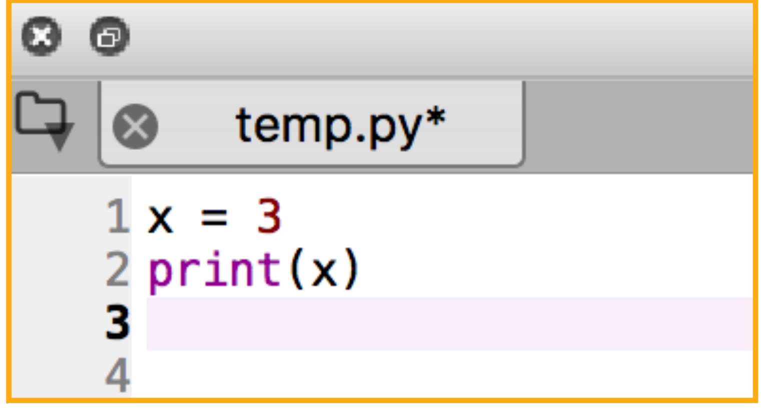
Notebook



```
In [1]: x = 3
        print(x)

        3

In [ ]:
```

- Web browser
- Clear input and result for each step
- **Demonstration**

Spyder, etc.



```
x = 3
print(x)
```

- Standalone App
- Convenient panels for multiple purposes
- **Development**

An Useful Calculator

1. Arithmetic Operations

```
3 + 5
3 - 5
3 * 5
3 / 5 # results in float
```

An Useful Calculator

1. Arithmetic Operations

```python
3 + 5
3 - 5
3 * 5
3 / 5 # results in float
```

2. Truncated Division and Modulo

```python
3 // 5 # returns 0
5 // 3 # returns 1
-5 // 3 # returns -2
5 % 3 # returns 2
```

An Useful Calculator

1. Arithmetic Operations

```python
3 + 5
3 - 5
3 * 5
3 / 5 # results in float
```

2. Truncated Division and Modulo

```python
3 // 5 # returns 0
5 // 3 # returns 1
-5 // 3 # returns -2
5 % 3 # returns 2
```

3. Exponentiation

```python
3 ** 5
3 * 5 ** 5 # returns 9375
(3 * 5) ** 5 # returns 759375
```

An Useful Calculator

4. Boolean

```
True
not True
True + True # returns 2
True * False # returns 0
```

An Useful Calculator

4. Boolean

```
True
not True
True + True # returns 2
True * False # returns 0
```

5. Variable Assignment

```
x = 3
y = 4
x * y # returns 12
```

An Useful Calculator

4. Boolean

```python
True
not True
True + True # returns 2
True * False # returns 0
```

5. Variable Assignment

```python
x = 3
y = 4
x * y # returns 12
```

6. More Data Types

```python
integer = 3
boolean = True
float = 3.5
string = 'I have a pen'
list = ['a', 6, 'c', 0.1, True]
```

Deal With Strings

7. Conversion Among Data Types

```
str(True)
int('3')
int(3.999)
float(3)
bool(3) # only bool() or bool(0) returns False
type(True)
```

Deal With Strings

7. Conversion Among Data Types

```
str(True)
int('3')
int(3.999)
float(3)
bool(3) # only bool() or bool(0) returns False
type(True)
```

8. Concatenate Strings

```
'a' + 'b' # returns 'ab'
'ab' * 2 # returns 'abab'
'abab' / 2 ### error, solve later
```

Deal With Strings

7. Conversion Among Data Types

```python
str(True)
int('3')
int(3.999)
float(3)
bool(3) # only bool() or bool(0) returns False
type(True)
```

8. Concatenate Strings

```python
'a' + 'b' # returns 'ab'
'ab' * 2 # returns 'abab'
'abab' / 2 ### error, solve later
```

9. Print Strings

```python
print('I have' + ' ' + 1 + ' ' + 'pen.') ### error
print('I have' + ' ' + str(1) + ' ' + 'pen.')
print('I have' + ' ' + str(int(True)) + ' ' + 'pen.')
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
```

len(test)

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error

```

len(test)

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

len(test)

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```python
# slice
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```python
# slice
test[1:3]
```

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```
# slice
test[1:3]
test[5:]
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```python
# slice
test[1:3]
test[5:]
test[:5]
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```python
# slice
test[1:3]
test[5:]
test[:5]
test[:-5]
```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

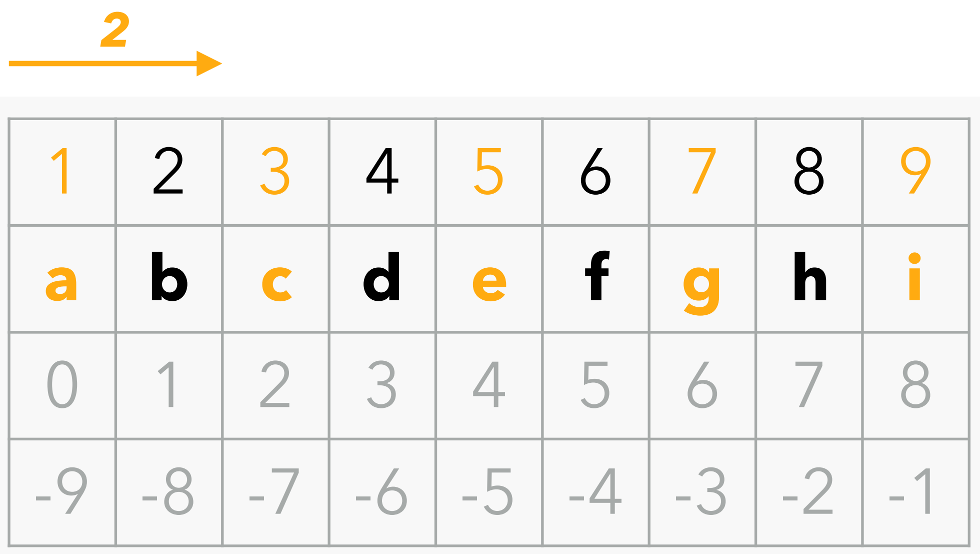
11. Advanced Index

```python
# slice
test[1:3]
test[5:]
test[:5]
test[:-5]
test[:]



```

Deal With Strings

10. Index Strings

```python
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```



Order No.	1	2	3	4	5	6	7	8	9
	a	b	c	d	e	f	g	h	i
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```python
# slice
test[1:3]
test[5:]
test[:5]
test[:-5]
test[:]
# specify stride
test[::2] # returns ‘acegi'

```

Deal With Strings

10. Index Strings

```
test = 'abcdefghi'
test[0] # returns 'a'
test[8] # returns 'i'
test[-1] # returns 'i'
len(test) # length of test
test[len(test)] ### error
test[-len(test)] # returns 'a'
```

$\xleftarrow{-3}$

Order No.	1	2	3	4	5	6	7	8	9
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Pos. Index	0	1	2	3	4	5	6	7	8
Neg. Index	-9	-8	-7	-6	-5	-4	-3	-2	-1

11. Advanced Index

```
# slice
test[1:3]
test[5:]
test[:5]
test[:-5]
test[:]
# specify stride
test[::2] # returns 'acegi'
test[::-3] # returns 'ifc'
```

Index Everything

12. String Division?

```
'abab' / 2 ### error
```

Order No.	1	2	3	4
	a	**b**	**a**	**b**
Pos. Index	0	1	2	3

Index Everything

12. String Division?

```python
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
```

len(test)

Order No.	1	2	3	4
	a	b	a	b
Pos. Index	0	1	2	3

Index Everything

12. String Division?

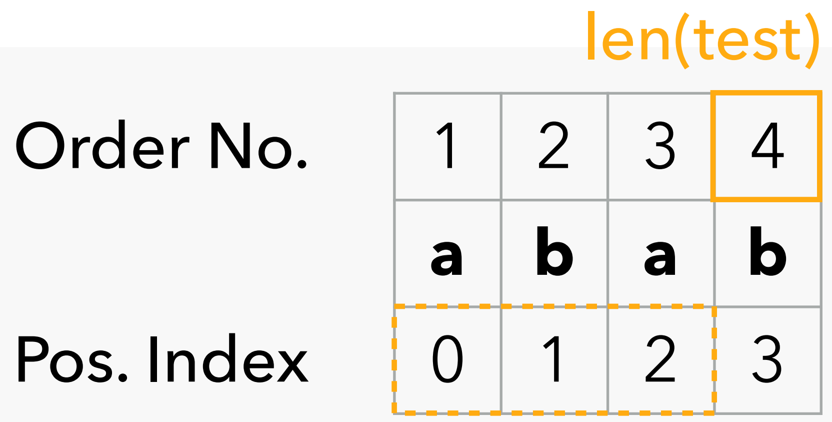
```python
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]


```

Order No.	1	2	3	4
	a	b	a	b
Pos. Index	0	1	2	3

Index Everything

12. String Division?

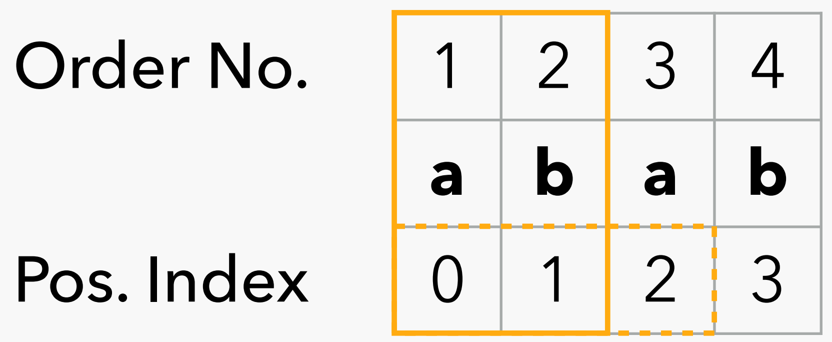
```
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]
test[:len(test)/2] ### error
```



Index Everything

12. String Division?

```python
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]
test[:len(test)/2] ### error
test[:int(len(test)/2)]
```



Order No.	1	2	3	4
	a	**b**	**a**	**b**
Pos. Index	0	1	2	3

Index Everything

12. String Division?

```
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]
test[:len(test)/2] ### error
test[:int(len(test)/2)]
```

Order No.	1	2	3	4
	a	**b**	**a**	**b**
Pos. Index	0	1	2	3

13. Index Lists

```
test = ['a','b','a','b']
test + test
test * 3
```

Index Everything

12. String Division?

```
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]
test[:len(test)/2] ### error
test[:int(len(test)/2)]
```

Order No.	1	2	3	4
	a	**b**	**a**	**b**
Pos. Index	0	1	2	3

13. Index Lists

```
test = ['a','b','a','b']
test + test
test * 3
# nested lists
test = [['a','b'],['a','b']]
test[0][1] # returns 'b'


```

Index Everything

12. String Division?

```python
'abab' / 2 ### error
test = 'abab'
test[:len(test)] # different from test[len(test)]
test[:100]
test[:len(test)/2] ### error
test[:int(len(test)/2)]
```

Order No.	1	2	3	4
	a	**b**	**a**	**b**
Pos. Index	0	1	2	3

13. Index Lists

```python
test = ['a','b','a','b']
test + test
test * 3
# nested lists
test = [['a','b'],['a','b']] # 2D list
test[0][1] # returns 'b'
test = ['a',['b',['c',['d']]]]
test[-1][-1][-1][-1] # returns 'd'
```

List Manipulation

14. Matrix-like List

```
test = [['Los Angeles', 34.0207504, -118.6919233],
        ['San Luis Obispo', 35.2725611, -120.7054056],
        ['San Francisco', 37.757815, -122.5076402]]
```

List Manipulation

14. Matrix-like List

```
test = [['Los Angeles', 34.0207504, -118.6919233],
        ['San Luis Obispo', 35.2725611, -120.7054056],
        ['San Francisco', 37.757815, -122.5076402]]
```

Los Angeles	34.0207504	-118.6919233
San Luis Obispo	35.2725611	-120.7054056
San Francisco	37.757815	-122.5076402

List Manipulation

14. Matrix-like List

```python
test = [['Los Angeles', 34.0207504, -118.6919233],
        ['San Luis Obispo', 35.2725611, -120.7054056],
        ['San Francisco', 37.757815, -122.5076402]]

test + ['San Jose', 37.2972061, -121.9574961] ### Incorrect


```

Los Angeles	34.0207504	-118.6919233
San Luis Obispo	35.2725611	-120.7054056
San Francisco	37.757815	-122.5076402

List Manipulation

14. Matrix-like List

```
test = [['Los Angeles', 34.0207504, -118.6919233],
        ['San Luis Obispo', 35.2725611, -120.7054056],
        ['San Francisco', 37.757815, -122.5076402]]

test + ['San Jose', 37.2972061, -121.9574961] ### Incorrect


```

Los Angeles	34.0207504	-118.6919233
San Luis Obispo	35.2725611	-120.7054056
San Francisco	37.757815	-122.5076402
San Jose		
37.2972061		
-121.9574961		

List Manipulation

14. Matrix-like List

```
test = [['Los Angeles', 34.0207504, -118.6919233],
        ['San Luis Obispo', 35.2725611, -120.7054056],
        ['San Francisco', 37.757815, -122.5076402]]

test + ['San Jose', 37.2972061, -121.9574961] ### Incorrect
test + [['San Jose', 37.2972061, -121.9574961]] # list + list
test.append(['San Diego', 32.8248175, -117.3753547])
```

Los Angeles	34.0207504	-118.6919233
San Luis Obispo	35.2725611	-120.7054056
San Francisco	37.757815	-122.5076402
San Jose	37.2972061	-121.9574961

List Manipulation

14. Matrix-like List (continued)

```







```

Los Angeles	34.0207504	-118.6919233
San Luis Obispo	35.2725611	-120.7054056
San Francisco	37.757815	-122.5076402
San Jose	37.2972061	-121.9574961

List Manipulation

14. Matrix-like List (continued)

```
test.pop(1) # returns and removes SLO
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402
San Jose	37.2972061	-121.9574961

→ SLO

List Manipulation

14. Matrix-like List (continued)

```
test.pop(1) # returns and removes SLO
del test[1] # removes SF
```

Los Angeles	34.0207504	-118.6919233
San Jose	37.2972061	-121.9574961

→ SF

List Manipulation

14. Matrix-like List (continued)

```
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude
```

Los Angeles	34.0207504	-118.6919233
San Jose		-121.9574961

List Manipulation

14. Matrix-like List (continued)

```
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1


```

Los Angeles	34.0207504	-118.6919233
San Jose	37.2972061	-121.9574961

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'

```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.2972061	-121.9574961

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

- Subset columns

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

- Subset columns
- Value manipulation

List Manipulation

14. Matrix-like List (continued)

```python
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

- Subset columns
- Value manipulation
- Headers

AND more challenges…

List Manipulation

14. Matrix-like List (continued)

```
test.pop(1) # returns and removes SLO
del test[1] # removes SF
del test[1][1] # removes SD's latitude

test[1].insert(1, 32.8248175) # inserts back at position 1
test[1][0] = 'San Francisco'
test[1] = ['San Francisco', 37.757815, -122.5076402]
```

Los Angeles	34.0207504	-118.6919233
San Francisco	37.757815	-122.5076402

Array (Matrix)

Data Frame

Lab: Lists

To get you familiar with how number, string, list and other data types work in Python.

To Make A List …

Here's a survey:

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21

1. Assign 4 variables for those observations
2. Combine them together to make a list

To Make A List …

Assign 4 variables for those observations

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21

```python
matt = ['Matt', 'MSBA', True, 23]
erin = ['Erin', 'MBA', True, 27]
jim = ['Jim', 'CS', False, 19]
kyle = ['Kyle', 'IE', True, 21]



```

To Make A List …

Combine them together to make a list

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21

```python
matt = ['Matt', 'MSBA', True, 23]
erin = ['Erin', 'MBA', True, 27]
jim = ['Jim', 'CS', False, 19]
kyle = ['Kyle', 'IE', True, 21]

survey = [matt, erin, jim, kyle]
print(survey)
```

To Make A List …

Manipulate the list:

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21
Patrick	MPH	No	27

3. Append Patrick's data into the list
4. Kyle's major is actually architecture

To Make A List …

Append Patrick's data into the list

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21
Patrick	MPH	No	27

```
patrick = ['Patrick', 'MPH', False, 27]
survey.append(patrick)
print(survey)
```

To Make A List …

Append Patrick's data into the list

Name	Major	CA Resident	Age
Matt	MSBA	Yes	23
Erin	MBA	Yes	27
Jim	CS	No	19
Kyle	IE	Yes	21
Patrick	MPH	No	27

```python
patrick = ['Patrick', 'MPH', False, 27]
survey.append(patrick)
print(survey)

print(survey[3][1]) # returns 'IE'
survey[3][1] = 'Architecture'
print(survey)
```