

# Lab for Week 12

Data Analysis with Python

Chaufournier & Wood  
CSCI 2541

# Jupyter Notebooks

- Allow you to run python code interactively through cells
- You can treat your code as modular and run cells in any order
  - Be careful, if you change earlier cells to refer to data defined in later cells you can run into runtime errors later on.
- Notebooks are just a giant JSON file with an object per Cell.
- Notebooks can be uploaded and rendered directly on Github
  - This makes them great for sharing and running data analysis with graphs

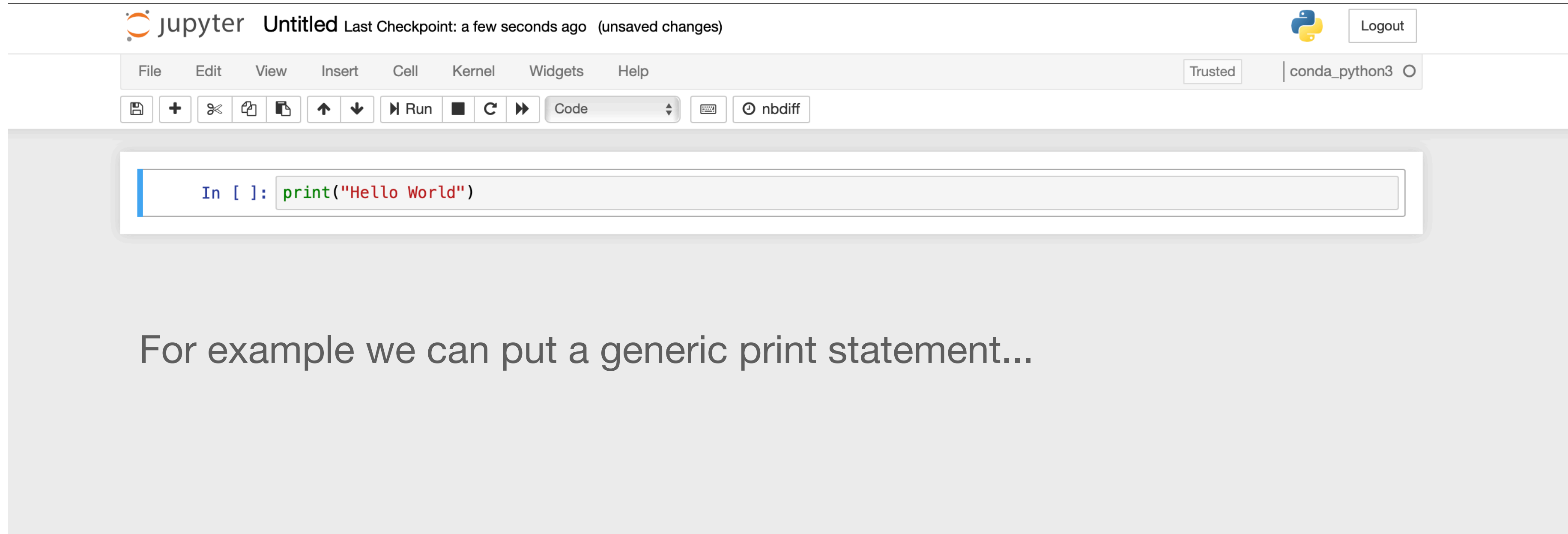
# Notebooks are made of cells



The screenshot shows the Jupyter Notebook interface. At the top, the text "jupyter Untitled" is displayed, followed by "Last Checkpoint: a few seconds ago (unsaved changes)". On the right side, there is a Python logo and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar, there is a status bar showing "Kernel starting, please wait...", "Trusted", and "conda\_python3". Below the menu bar is a toolbar with icons for file operations (save, new, copy, paste), navigation (up, down), execution (run, stop, refresh), and other functions (Code, nbdiff).

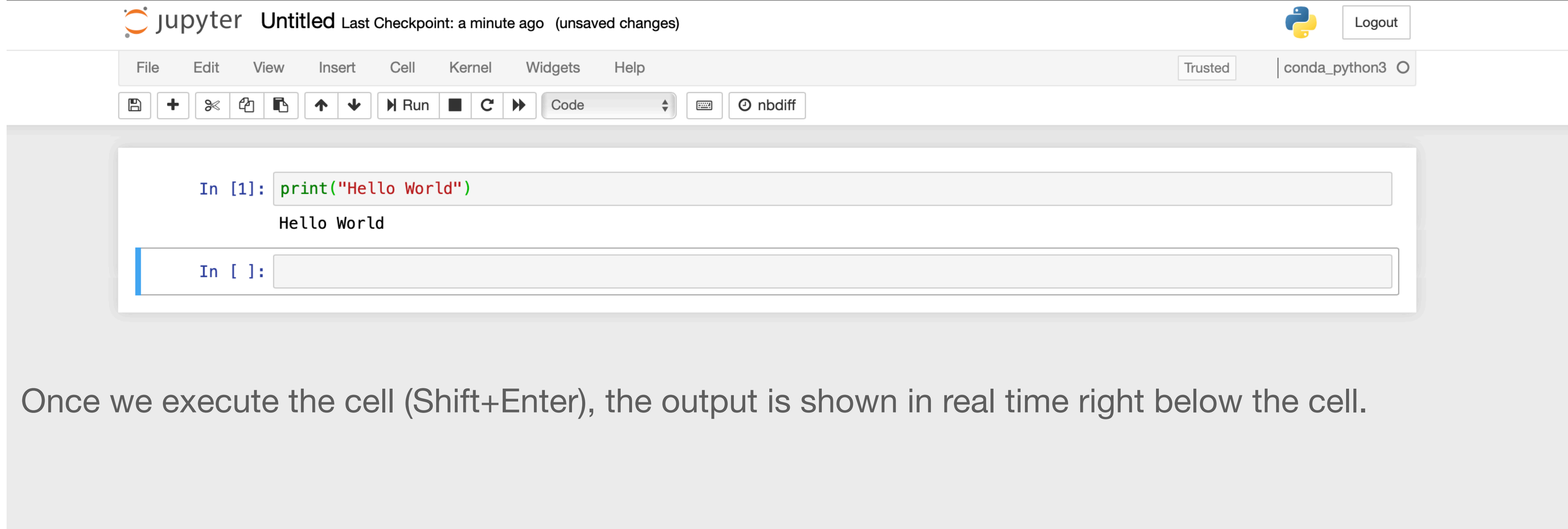
Cells allow you to write python code and run it in a real time and modular fashion. You can put any python code in a cell and then hit Shift+Enter to run the code.

# Programming Cells



The image shows a Jupyter Notebook interface. At the top, the header includes the Jupyter logo, the text "jupyter Untitled", and "Last Checkpoint: a few seconds ago (unsaved changes)". On the right side of the header, there is a Python logo and a "Logout" button. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar, it says "Trusted" and "conda\_python3". Below the menu bar is a toolbar with icons for saving, adding, undo, redo, up/down arrows, Run, Stop, Refresh, and nbdiff. The main area of the notebook contains a single code cell with the text "In [ ]: print('Hello World')". Below the code cell, there is a large grey rectangular area containing the text "For example we can put a generic print statement..."

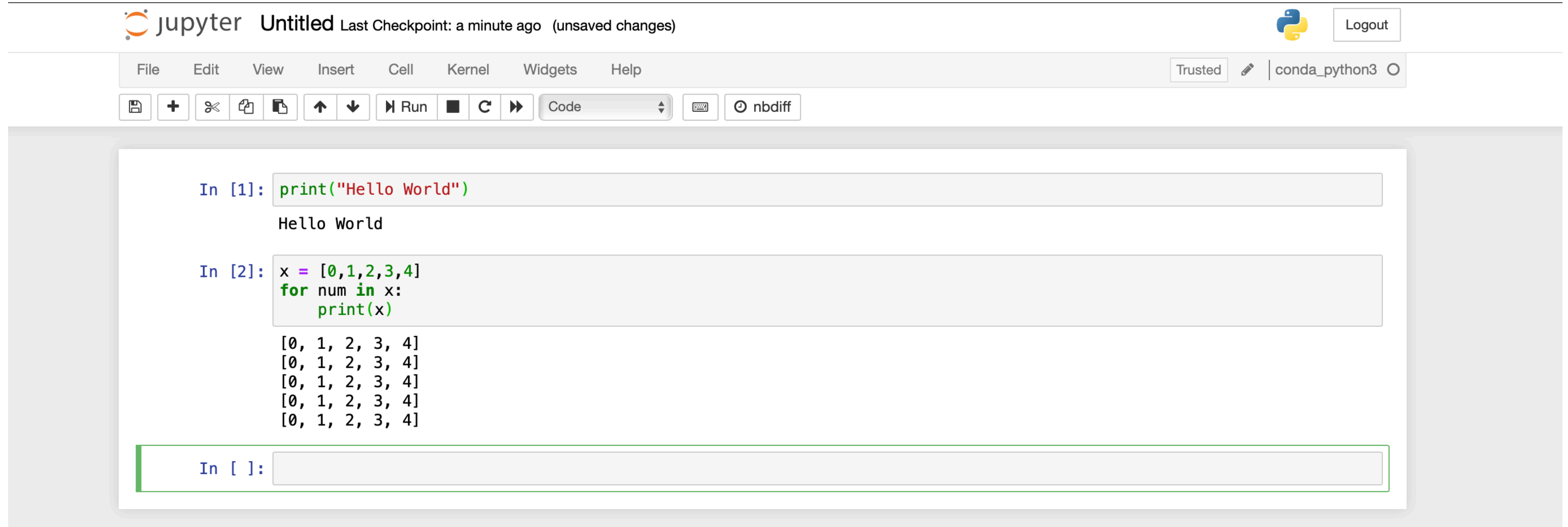
# Executing cells and viewing output



The screenshot displays the Jupyter Notebook interface. At the top, the header shows the Jupyter logo, the text "jupyter Untitled", and "Last Checkpoint: a minute ago (unsaved changes)". On the right side of the header, there is a Python logo and a "Logout" button. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar, there is a "Trusted" status indicator and a "conda\_python3" environment selector. Below the menu bar is a toolbar with various icons for file operations (save, new, copy, paste), navigation (up, down), execution (Run, Stop, Refresh), and other functions (Code, Keyboard shortcuts, nbdiff). The main area of the notebook contains two code cells. The first cell, labeled "In [1]:", contains the code `print("Hello World")` and has the output "Hello World" displayed below it. The second cell, labeled "In [ ]:", is currently empty and has a blue cursor at the beginning of the input line.

Once we execute the cell (Shift+Enter), the output is shown in real time right below the cell.

# Cells support multiple lines of logic



The screenshot shows a Jupyter Notebook interface. At the top, the title bar reads "jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)". On the right, there is a Python logo and a "Logout" button. Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. To the right of the menu bar, it says "Trusted" and "conda\_python3". Below the menu bar is a toolbar with icons for file operations (save, new, copy, paste, undo, redo), a "Run" button, a "Code" dropdown menu, and an "nbdiff" button. The main area contains three code cells. The first cell, labeled "In [1]:", contains the code `print("Hello World")` and has the output "Hello World". The second cell, labeled "In [2]:", contains the code `x = [0,1,2,3,4]` followed by a `for` loop: `for num in x:` and `print(x)`. The output of this cell is five lines, each containing the list `[0, 1, 2, 3, 4]`. The third cell, labeled "In [ ]:", is empty and has a green border.

We can also put multiple lines in a single cell and add complex logic.  
All of the lines in the cell will be executed in order.

# Cells can be referenced by other cells

```
In [1]: print("Hello World")
```

```
Hello World
```

```
In [2]: x = [0,1,2,3,4]
        for num in x:
            print(x)
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

```
In [3]: def sum(x,y):
        return x+y
```

```
In [5]: sum(10,56)
```

```
Out[5]: 66
```

```
In [ ]:
```

You can also implement code in one cell and reference it from another cell

Make sure you execute the code in the cell before you try to reference it from a separate cell.

You need to run cells to have them visible in the runtime.

# Cells will display errors as part of their outputs

```
In [3]: def sum(x,y):  
        return x+y
```

```
In [5]: sum(10,56)
```

```
Out[5]: 66
```

```
In [6]: sum(10)
```

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-6-e86a74b26fb4> in <module>  
----> 1 sum(10)  
  
TypeError: sum() missing 1 required positional argument: 'y'
```

```
In [ ]:
```

Cells will also show you errors in real time. This allows you to easily debug your code



# Pandas and Data Analysis

# Pandas

## Data Analysis Made Easy

- "pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language"
- Pandas operates using the concept of DataFrames which are tabular data structures similar to a table.
- You can dump your data into a DataFrame and then gain access to a variety of library functions that operate on your data
- These functions allow you to clean, reshape, and interpolate your data; describe summary statistics; graph results; and even run machine learning models.

# Creating a DataFrame

```
In [8]: import pandas as pd
```

```
In [31]: d = [  
    {"name": "john", "age": 26, "score": 100},  
    {"name": "bill", "age": 57, "score": 20},  
    {"name": "jane", "age": 32, "score": 125},  
    {"name": "dane", "age": 26, "score": 44},  
    {"name": "roderick", "age": 32, "score": 223},  
    {"name": "jaime", "age": 57, "score": 86},  
    {"name": "helen", "age": 32, "score": 45},  
    {"name": "judith", "age": 32, "score": 55},  
    {"name": "alice", "age": 57, "score": 90}  
]
```

```
In [32]: df = pd.DataFrame(d)
```

```
In [33]: print(df)
```

	name	age	score
0	john	26	100
1	bill	57	20
2	jane	32	125
3	dane	26	44
4	roderick	32	223
5	jaime	57	86
6	helen	32	45
7	judith	32	55
8	alice	57	90

```
In [ ]:
```

- A variety of data structures can be converted to pandas DataFrames.
- The easiest and most useful to us is a list of dictionaries.
- Simply cast the list to a pandas DataFrame and you are good to go.
- This should look familiar to you. (Think back to last weeks lab)

# Pandas Describe

```
In [32]: df = pd.DataFrame(d)
```

```
In [33]: print(df)
```

```
   name  age  score
0  john   26   100
1  bill   57    20
2  jane   32   125
3  dane   26    44
4  roderick 32   223
5  jaime   57    86
6  helen   32    45
7  judith  32    55
8  alice   57    90
```

```
In [35]: df.describe()
```

Out [35]:

	age	score
<b>count</b>	9.000000	9.000000
<b>mean</b>	39.000000	87.555556
<b>std</b>	13.720423	60.458893
<b>min</b>	26.000000	20.000000
<b>25%</b>	32.000000	45.000000
<b>50%</b>	32.000000	86.000000
<b>75%</b>	57.000000	100.000000
<b>max</b>	57.000000	223.000000

The builtin describe function on a DataFrame allows you to see summary statistics about your data.

This can be useful for understanding how your data is distributed and can enable you to make better decisions.

# Pandas Groupby

```
{ "name": "dane", "age": 26, "score": 44 },  
{"name": "roderick", "age": 32, "score": 223},  
{"name": "jaime", "age": 57, "score": 86},  
{"name": "helen", "age": 32, "score": 45},  
{"name": "judith", "age": 32, "score": 55},  
{"name": "alice", "age": 57, "score": 90}  
]
```

```
In [32]: df = pd.DataFrame(d)
```

```
In [33]: print(df)
```

```
   name  age  score  
0  john   26    100  
1  bill   57     20  
2  jane   32    125  
3  dane   26     44  
4  roderick 32    223  
5  jaime   57     86  
6  helen   32     45  
7  judith  32     55  
8  alice   57     90
```

```
In [36]: df[["age", "score"]].groupby("age").mean()
```

```
Out[36]:
```

	score
age	
26	72.000000
32	112.000000
57	65.333333

Dataframes also support querying and grouping your data like you would in SQL so you can gather insights into your data.

# Pandas graphing

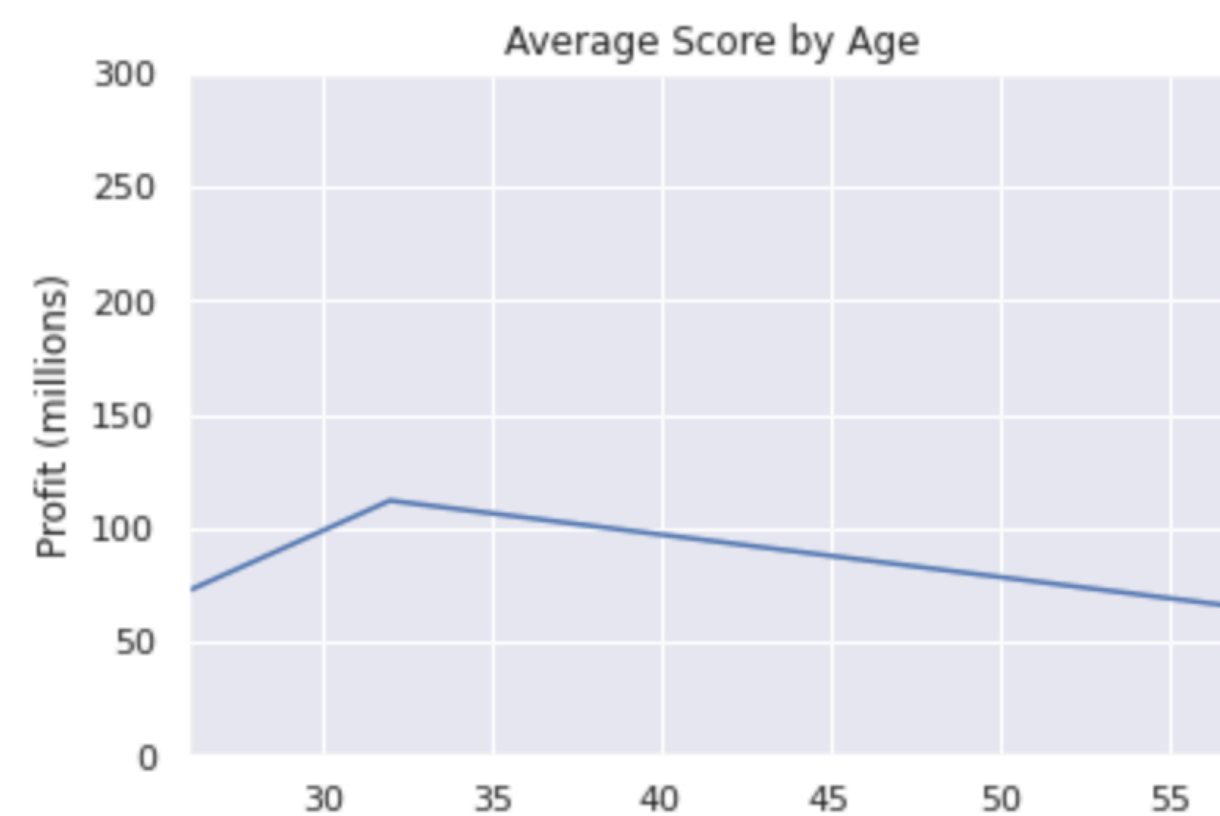
```
In [22]: group_by_age = df.loc[:, ['age', 'score']].groupby('age')

#Select the averages for each ag
avgs = group_by_age.mean()

# X will be the ages
x = avgs.index

# Y will be the increase in mean score by age
y1 = avgs.score
def plot(x, y, ax, title, y_label):
    ax.set_title(title)
    ax.set_ylabel(y_label)
    ax.plot(x, y)
    ax.margins(x=0, y=0)
    ax.set_ylim([0, 300])
```

```
In [23]: fig, ax = plt.subplots()
plot(x, y1, ax, 'Average Score by Age', 'Score')
```

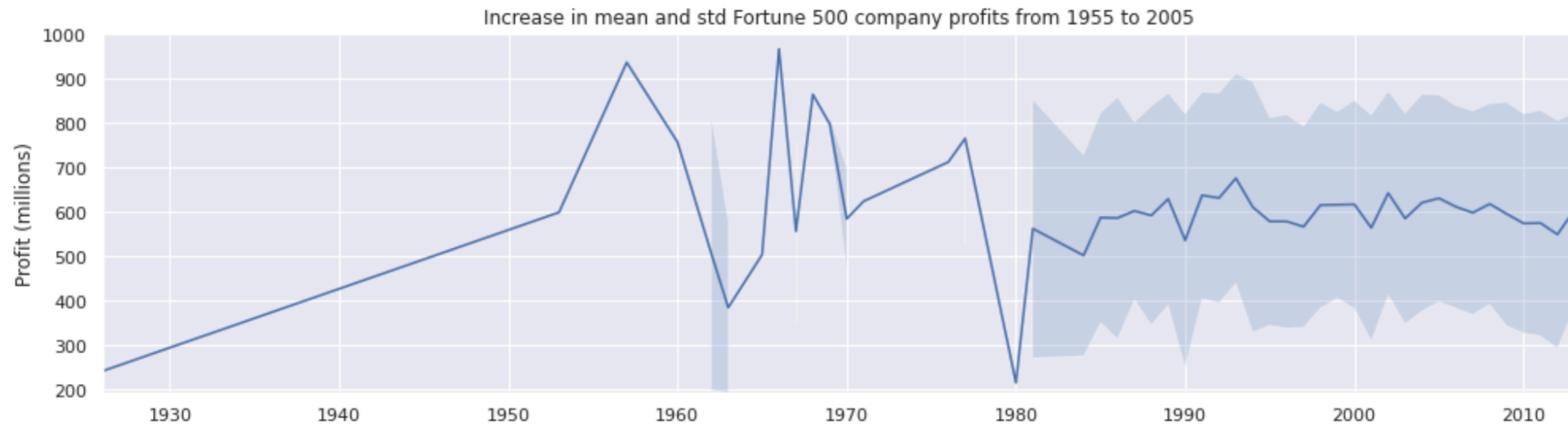


```
In [ ]:
```

Dataframes also allow for easy graphing of your data. You can easily use libraries such as Matplotlib to construct graphs of your data.

# More Complex Graphs

```
In [70]: def plot_with_std(x, y, stds, ax, title, y_label):  
         ax.fill_between(x, y - stds, y + stds, alpha=0.2)  
         plot(x, y, ax, title, y_label)  
fig, ax1 = plt.subplots()  
title = 'Increase in mean and std Fortune 500 company %s from 1955 to 2005'  
stds1 = group_by_year.std().profit.values  
plot_with_std(x, y1.values, stds1, ax1, title % 'profits', 'Profit (millions)')  
fig.set_size_inches(14, 4)  
fig.tight_layout()
```



# Various other dataframe functions

- `df.columnName.min()` -- The minimum value for a given column.
- `df.columnName.median()` -- The median value for a given column
- `df.columnName.max()` -- The max value for a given column
- `df.columnName.mode()` -- The mode for a given column
- `df.columnName.std()` -- The standard deviation for a column

Here you should replace `columnName` with one of the column titles for your dataset



How do we run our own  
notebooks?

**Amazon Sagemaker!**

# Navigate to Amazon SageMaker in the Console

The screenshot shows the AWS Management Console interface. At the top, the AWS logo and 'Services' dropdown are visible. A search bar contains the text 'sagemaker'. The main content area displays search results for 'sagemaker', categorized into Services, Features, and Documentation. The 'Services' section highlights 'Amazon SageMaker' with a blue border, described as 'Build, Train, and Deploy Machine Learning Models'. Below it is 'AWS Glue DataBrew', described as a 'Visual data preparation tool to clean and normalize data for analytics and machine learning'. The 'Features' section lists 'Notebooks' with an 'IoT Analytics feature'. The 'Documentation' section includes links to 'SageMaker Roles - Amazon SageMaker', 'What is a SageMaker Project? - Amazon SageMaker', and 'Automate MLOps with SageMaker Projects - Amazon SageMaker'. On the right side, there are promotional banners for staying connected via mobile app, exploring AWS resources, and trying AWS Graviton2 based EC2 instances.

aws Services

Search results for 'sagemaker'

Services

- Amazon SageMaker**  
Build, Train, and Deploy Machine Learning Models
- AWS Glue DataBrew**  
Visual data preparation tool to clean and normalize data for analytics and machine learning...

Features

- Notebooks**  
IoT Analytics feature

Documentation [See all 38,976 results in Documentation](#)

- SageMaker Roles - Amazon SageMaker**  
Developer Guide
- What is a SageMaker Project? - Amazon SageMaker**  
Developer Guide
- Automate MLOps with SageMaker Projects - Amazon SageMaker**  
Developer Guide

Stay connected to your AWS on-the-go

AWS Console Mobile App additional regions. Download Mobile App to your iOS or device. [Learn more](#)

Explore AWS

**Amazon SageMaker Resources**  
Explore features, use cases, and tutorials for developers. [Learn more](#)

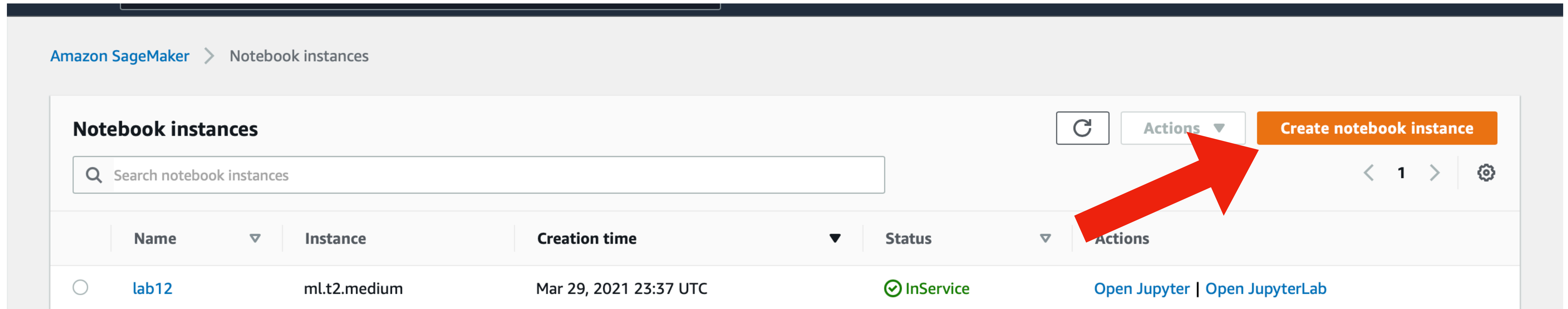
**Try AWS Graviton2 Based EC2 Instances Free**  
See how running your workloads on AWS Graviton2 based EC2 instances can provide the best price performance with 750 hours free until June 2021. [Learn more](#)

**Automate Document Data Processing**  
Extract text and understand the content of documents using Amazon SageMaker and Amazon Comprehend.

# Click on Notebook instances on the left

The screenshot shows the AWS SageMaker console interface. At the top, there is a search bar with the text "Search for services, features, marketplace products, and docs" and a user profile "vocstartsoft/user=admin @ 439". The left sidebar contains a navigation menu for "Amazon SageMaker" with the following items: Amazon SageMaker Studio, Dashboard, Search, Images, Ground Truth, Notebook (expanded), Notebook instances (highlighted with a red arrow), Lifecycle configurations, Git repositories, Processing, Training, Inference, Edge Manager, Augmented AI, and AWS Marketplace. The main content area displays the "MACHINE LEARNING" header and the "Amazon SageMaker" title, followed by the subtitle "Build, train, and deploy machine learning models at scale" and the tagline "The quickest and easiest way to get ML models from idea to production." Below this is a "How it works" section with three steps: Label (Set up and manage labeling jobs for highly accurate training datasets), Build (Connect to other AWS services and transform data in Amazon), and Train (Use Amazon SageMaker's algorithms and frameworks, or bring your own). On the right side, there are sections for "Get started" (with a "SageMaker Studio" button), "Pricing (US)" (with a "Learn more" link), and "Related services" (listing "AWS Glue" and "Amazon EC2").

# Click "Create notebook instance"



The screenshot shows the Amazon SageMaker Notebook instances management console. At the top left, the breadcrumb navigation reads "Amazon SageMaker > Notebook instances". Below this is a header section titled "Notebook instances" which includes a search bar, a refresh button, an "Actions" dropdown menu, and a prominent orange "Create notebook instance" button. A large red arrow points from the "Actions" dropdown towards the "Create notebook instance" button. Below the header is a table with the following columns: Name, Instance, Creation time, Status, and Actions. One instance is listed with the name "lab12", instance type "ml.t2.medium", creation time "Mar 29, 2021 23:37 UTC", and status "InService". The Actions column for this instance contains links for "Open Jupyter" and "Open JupyterLab".

Name	Instance	Creation time	Status	Actions
lab12	ml.t2.medium	Mar 29, 2021 23:37 UTC	InService	Open Jupyter   Open JupyterLab

We are going to create a new notebook instance which is a server allowing you to create, host, and run your notebooks

# Enter notebook details

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

### Notebook instance settings

Notebook instance name

week12-lab

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t2.medium

Elastic Inference [Learn more](#)

none

▶ Additional configuration

### Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20210329T193756

Create a new role

Enter a custom IAM role ARN

Create a new role

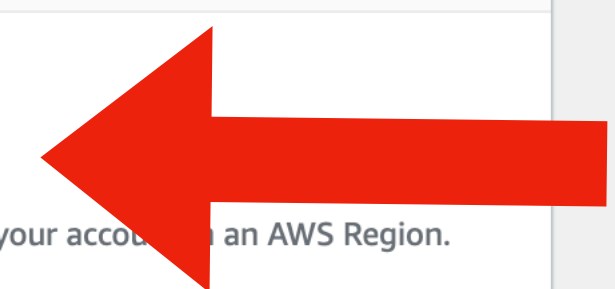
Use existing role

AmazonSageMaker-ExecutionRole-20210329T193756

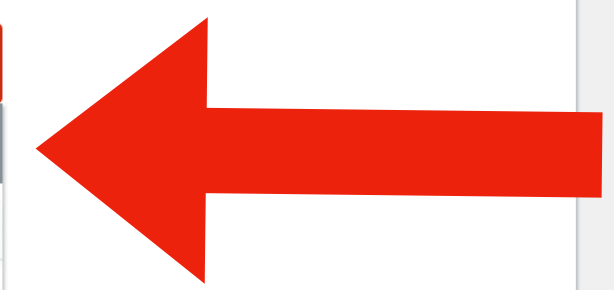
Encryption key - *optional*

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption



Enter week12-lab for the notebook name



Under IAM role select "Create a new role"

# Create a new IAM role

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances provide a secure, managed environment for developing and testing machine learning models. For more information, see [Example code for common model training and hosting exercises](#). [Learn more](#)

### Notebook instance settings

Notebook instance name:

Notebook instance type:

Additional configuration:  [Learn more](#)

### Permissions and encryption

Create a new role

Use provide a valid Arn.

Access - optional

Root access - Give users root access to the notebook instance

Root access - Don't give users root access to the notebook instance (Notebook configurations always have root access)

Encryption key - optional

Choose an existing KMS key or enter a key's ARN.

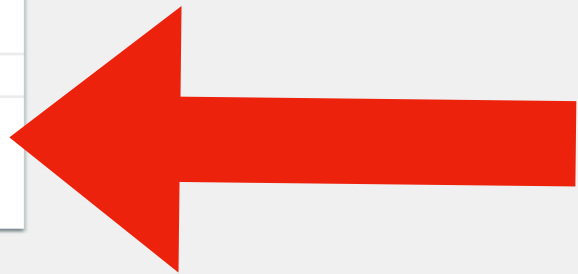
Custom Encryption

### Create an IAM role

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

- S3 buckets you specify - *optional*
  - Any S3 bucket  
Allow users that have access to your notebook instance access to any bucket and its contents in your account.
  - Specific S3 buckets  
  
Comma delimited. ARNs, "\*" and "/" are not supported.
  - None
- Any S3 bucket with "sagemaker" in the name
- Any S3 object with "sagemaker" in the name
- Any S3 object with the tag "sagemaker" and value "true" [See Object tagging](#)
- S3 bucket with a Bucket Policy allowing access to SageMaker [See S3 bucket policies](#)



Hit "Create role" without modifying anything

# Create notebook!

☰

Notebook instance type

ml.t2.medium ▼

Elastic Inference [Learn more](#) [↗](#)

none ▼

▶ **Additional configuration**

---

**Permissions and encryption**

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20210329T193756 ▼

Root access - *optional*

Enable - Give users root access to the notebook

**Disable - Don't give users root access to the notebook**  
Lifecycle configurations always have root access

Encryption key - *optional*

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption ▼

---

▶ **Network - optional**

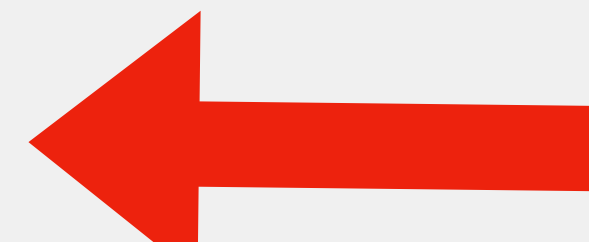
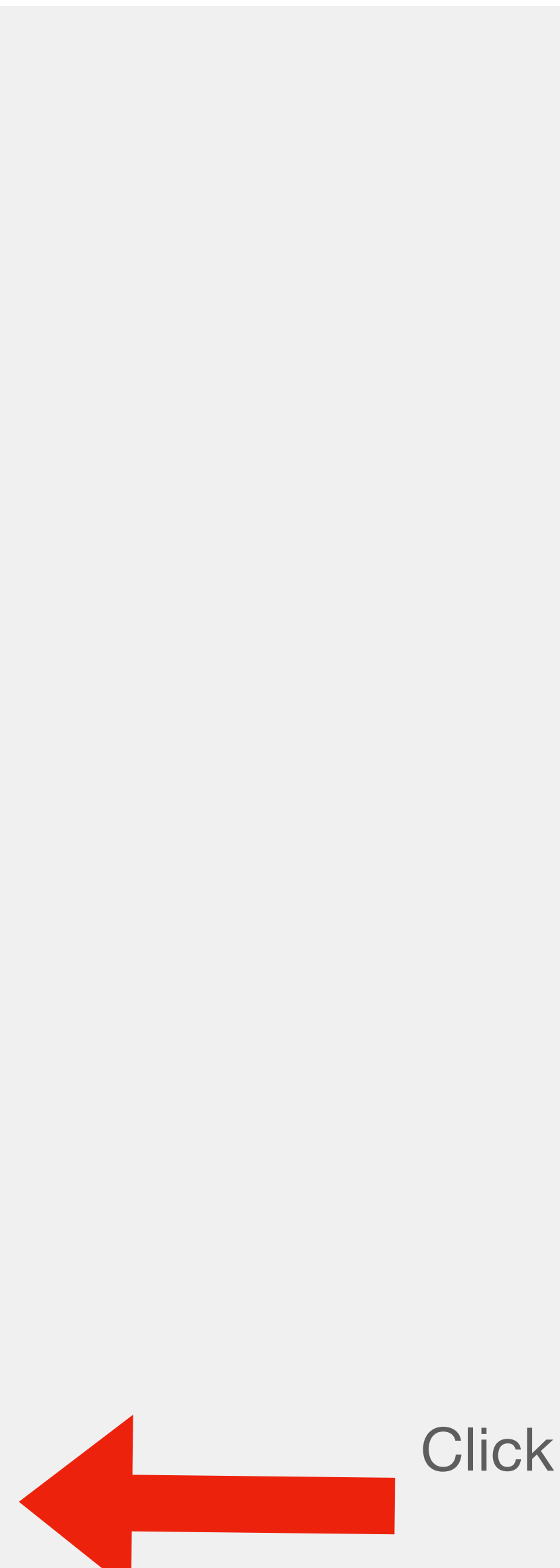
---

▶ **Git repositories - optional**

---

▶ **Tags - optional**

Cancel **Create notebook instance**



Click Create Notebook Instance



# Click "Open Jupyter"

Amazon SageMaker > Notebook instances

Notebook instances Refresh Actions Create notebook instance

Search notebook instances < 1 > Settings

Name	Instance	Creation time	Status	Actions
<input type="radio"/> week12-lab	ml.t2.medium	Mar 30, 2021 00:25 UTC	<span>InService</span>	<a href="#">Open Jupyter</a>   <a href="#">Open JupyterLab</a>



Click Open Jupyter

Note: It may take a few minutes for your notebook to initialize. Once it's started you will be able to click Open Jupyter.

Select items to perform actions on them.

Upload

New



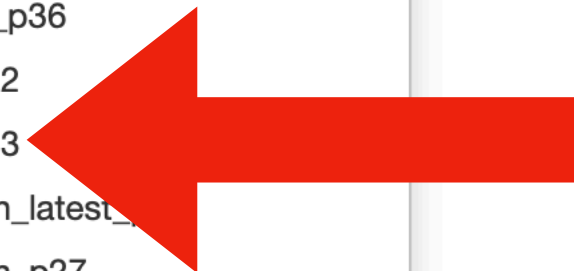
0 /

The notebook list is empty.

- Notebook:
- R
  - Sparkmagic (PySpark)
  - Sparkmagic (Spark)
  - Sparkmagic (SparkR)
  - conda\_amazonei\_mxnet\_p27
  - conda\_amazonei\_mxnet\_p36
  - conda\_amazonei\_pytorch\_latest\_p36
  - conda\_amazonei\_tensorflow2\_p27
  - conda\_amazonei\_tensorflow2\_p36
  - conda\_amazonei\_tensorflow\_p27
  - conda\_amazonei\_tensorflow\_p36
  - conda\_chainer\_p27
  - conda\_chainer\_p36
  - conda\_mxnet\_latest\_p37
  - conda\_mxnet\_p27
  - conda\_mxnet\_p36
  - conda\_python2
  - conda\_python3
  - conda\_pytorch\_latest\_p36
  - conda\_pytorch\_p27
  - conda\_pytorch\_p36
  - conda\_tensorflow2\_p36
  - conda\_tensorflow\_p27
  - conda\_tensorflow\_p36
- Other:
- Text File
  - Folder
  - Terminal

Jupyter notebooks support many types of kernels, which are functionally runtimes configured to compile and run your code.

For this exercise we will only need the conda\_python3 kernel which allows you to run python3 commands



# Cleaning up your infrastructure

The screenshot shows the AWS SageMaker console interface. The top navigation bar includes the AWS logo, a search bar, and user information. The left sidebar contains navigation options for SageMaker Studio, Dashboard, Search, Images, Ground Truth, Notebook (with 'Notebook instances' selected), Processing, Training, Inference, Edge Manager, and Augmented AI. The main content area displays the 'Notebook instances' page with a table of instances. A red arrow points to the 'Stop' option in the 'Actions' dropdown menu for the 'week12-lab' instance.

Name	Instance	Creation time	Status
week12-lab	ml.t2.medium	Mar 30, 2021 00:25 UTC	InService

Actions menu options:

- Open Jupyter
- Open JupyterLab
- Stop
- Start
- Update settings
- Add/Edit tags
- Delete

# Cleaning up your infrastructure

The screenshot shows the AWS SageMaker console interface. On the left is a navigation sidebar with options like Amazon SageMaker Studio, Dashboard, Search, Images, Ground Truth, Notebook (with Notebook instances selected), Lifecycle configurations, Git repositories, Processing, Training, Inference, Edge Manager, and Augmented AI. The main content area is titled 'Amazon SageMaker > Notebook instances'. It features a search bar, a refresh button, and a 'Create notebook instance' button. Below is a table with columns for Name, Instance, Creation time, and Status. One instance, 'week12-lab' (ml.t2.medium, Mar 30, 2021 00:25 UTC), is selected. An 'Actions' dropdown menu is open for this instance, listing options: Open Jupyter, Open JupyterLab, Stop, Start, Update settings, Add/Edit tags, and Delete. A large red arrow points from the right towards the 'Delete' option in the menu.

Name	Instance	Creation time	Status
week12-lab	ml.t2.medium	Mar 30, 2021 00:25 UTC	Stopped

# Summary

- Data Analysis
- Amazon SageMaker
- Python - 3 Ways to code
  - Directly in a terminal - Great for testing and running small code
  - Using an IDE - Great for working in teams and on large projects
  - Notebooks - Great for prototyping, data analysis with visuals

# Activity

- Clone this repo and upload the files to your notebook server:
  - <https://github.com/cs2541-21s/week12-lab>
- Figure out how to graph the following items:
  - Mean increase in revenue for companies by year
  - Mean increase in valuation for companies by year.
  - The max profit by year