

---

# **Proceedings of the REU2022**

Cybertraining

Gregor von Laszewski, laszewski@gmail.com

04 June, 2022

## Contents

<b>1 USEFUL LINKS</b>	<b>7</b>
1.1 REU 2022 	7
1.1.1 Books . . . . .	7
1.1.2 The repository for our report(s) . . . . .	7
1.1.3 The many UVA technical help ticket systems . . . . .	7
1.1.4 Our technical slack . . . . .	7
1.1.5 The administrative slack . . . . .	8
1.1.6 Rivanna Information . . . . .	8
1.1.7 GitBash (Windows only) . . . . .	8
1.1.8 Pycharm . . . . .	8
1.1.9 Bibliography Management . . . . .	8
1.1.10 GitHub repo . . . . .	9
1.1.11 Only important for Gregor (do not use) . . . . .	9
<b>2 INSTALL</b>	<b>9</b>
2.1 Install 	9
2.1.1 Windows . . . . .	10
2.1.1.1 Git Bash install . . . . .	10
2.1.1.2 Python 3.10 install . . . . .	11
2.1.1.3 Installing cloudmesh . . . . .	11
2.1.1.4 Uninstall . . . . .	12
2.1.2 Choco install . . . . .	12
2.1.3 Install Chocolatey . . . . .	12
2.1.4 Installing Useful Developer Programs . . . . .	13
2.1.5 Linux . . . . .	13
2.1.5.1 Install Python 3.10.5 . . . . .	13
2.1.5.2 Setting up the venv . . . . .	13
2.1.5.3 Uninstall . . . . .	14
2.1.5.4 Update . . . . .	14
2.1.6 macOS . . . . .	14
2.1.6.1 Xcode Install . . . . .	14
2.1.6.2 Cloudmesh . . . . .	15
2.1.6.2.1 Install . . . . .	15
2.1.6.2.2 Uninstall . . . . .	15
2.1.6.3 Updating Python . . . . .	15
2.1.6.4 Homebrew install . . . . .	16

2.2 Ramdisk	17
2.2.1 Ubuntu	17
2.2.2 windows	18
<b>3 GRAPH VIZUALIZATION</b>	<b>18</b>
3.1 Python Data Management for Visualizations	20
3.1.1 Lists	20
3.1.1.1 Construction	20
3.1.1.2 Accessing Values	20
3.1.1.3 Updating Values	20
3.1.1.4 Python Built-In Methods	21
3.1.2 Dictionaries	21
3.1.2.1 Construction	21
3.1.2.2 Accessing Values	21
3.1.2.3 Updating Values	21
3.1.2.4 Python Built-In Methods	22
3.1.3 CSV Files	22
3.1.3.1 Installing and Importing	22
3.1.3.2 Construction	22
3.1.3.3 Accessing Values	23
3.1.3.4 Examples	23
3.1.4 Links	25
3.1.4.1 Lists	25
3.1.4.2 Dictionaries	25
3.1.4.3 CSV Files	25
3.2 Python Graphics	26
3.2.1 Matplotlib	27
3.2.1.1 Installation	27
3.2.1.2 Import Statements	27
3.2.1.3 Bar Chart	27
3.2.1.4 Line Chart	28
3.2.1.5 Pie Chart	29
3.2.1.6 Contour Plot	30
3.2.1.7 Titles, Labels, and Legends	31
3.2.1.7.1 Titles	31
3.2.1.7.2 x-axis labels and y-axis labels	31
3.2.1.7.3 Legend	32
3.2.1.8 Rotating Ticks	32

3.2.1.9	Exporting . . . . .	32
3.2.1.9.1	Saving Chart as Files After a chart is created and displayed, it . . . . .	33
3.2.1.9.2	Display . . . . .	33
3.2.1.10	Links . . . . .	33
3.3	Seaborn 	34
3.3.1	Installation . . . . .	34
3.3.2	Import Statements . . . . .	34
3.3.3	Relational Plots . . . . .	34
3.3.4	Distribution Plots . . . . .	35
3.3.5	Categorical Plots . . . . .	36
3.3.6	Regression Plots . . . . .	37
3.3.7	Saving Figures . . . . .	38
3.3.8	Links . . . . .	38
3.4	Bokeh 	38
3.4.1	Installation . . . . .	38
3.4.2	Import Statements . . . . .	38
3.4.3	Bokeh Plotting Interface . . . . .	39
3.4.4	Figure Parameters Example . . . . .	40
3.4.5	Scatter Plot . . . . .	40
3.4.6	Line Plots . . . . .	42
3.4.7	Bar Chart . . . . .	43
3.4.8	Saving Figures . . . . .	43
3.4.8.1	Saving Figures as PNG . . . . .	44
3.4.9	Links . . . . .	45
3.5	Pandas Graphics 	45
3.5.1	Installation . . . . .	45
3.5.2	Import Statements . . . . .	45
3.5.3	Bar Chart . . . . .	46
3.5.4	Line Chart . . . . .	47
3.5.5	Pie Chart . . . . .	48
3.5.6	Exporting . . . . .	49
3.5.7	Links . . . . .	50
<b>4</b>	<b>PYTHON</b>	<b>50</b>
4.1	Cloudmesh StopWatch 	50
4.2	cms sys command generate 	51
4.2.1	Creating CLI commands using the cloudmesh command . . . . .	51
4.2.1.1	Example of the CLI Creation . . . . .	52

4.2.1.2	Example of CLI meshing with a python script . . . . .	52
4.3	Linux  . . . . .	54
4.4	Python  . . . . .	54
4.4.1	Queue . . . . .	54
4.4.1.1	FIFO Queue . . . . .	55
4.4.1.2	LIFO Queue . . . . .	55
4.4.1.3	Priority Queue . . . . .	55
4.5	glob  . . . . .	56
4.5.1	Glob with asterisk . . . . .	56
4.5.2	Single Character Wildcard (?) . . . . .	57
4.5.3	Escape Characters . . . . .	57
4.5.4	Subdirectories . . . . .	58
4.5.5	Links . . . . .	58
4.6	Mmap  . . . . .	58
4.6.1	Reading . . . . .	58
4.6.2	Writing . . . . .	59
4.6.3	Links . . . . .	62
4.7	Pickle  . . . . .	62
4.7.1	Encoding Data . . . . .	62
4.7.2	Decoding Data . . . . .	62
4.7.3	Links . . . . .	63
4.8	Shelve  . . . . .	63
4.8.1	Creating a New Shelf . . . . .	63
4.8.2	Accessing a Shelf . . . . .	64
4.8.3	Making Shelf Read-Only . . . . .	64
4.8.4	Modifying Shelves . . . . .	65
4.8.5	Closing Shelves . . . . .	66
4.8.6	Links . . . . .	66
4.9	FastAPI  . . . . .	66
4.9.1	FastAPI Install . . . . .	66
4.9.2	FastAPI Quickstart . . . . .	67
4.9.2.1	Path . . . . .	67
4.9.3	Query Parameters . . . . .	68
4.9.3.1	Searching in the fastapi . . . . .	68
4.9.3.2	Running Through Git bash . . . . .	69
4.9.4	Running the cc FastAPI service . . . . .	70
4.9.5	Testing . . . . .	70
4.9.5.1	Asynchronous Testing . . . . .	70

4.9.6	Links . . . . .	71
<b>5</b>	<b>RIVANNA</b>	<b>71</b>
5.1	Rivanna  . . . . .	71
5.1.1	Notes: superpod . . . . .	73
5.1.2	Special DGX Nodes on Rivanna . . . . .	74
5.1.2.1	Starting interactive job on special partition . . . . .	74
5.1.3	SSH Config . . . . .	75
5.2	Run Python MPI programs on Rivanna  . . . . .	76
<b>6</b>	<b>NLP</b>	<b>76</b>
6.1	Documentation to get started with AWS Translate Service  . . . . .	76
6.1.1	Step 1: Creating a new iam user account on aws. . . . .	76
6.1.1.1	Step 2: Creating a access key ID and secret ID . . . . .	77
6.2	Documentation to get started with Azure Translate.  . . . . .	83
6.2.1	Step 1: account creation . . . . .	83
6.2.2	Installing and Starting Azure Translate through the command line . . . . .	89
6.2.2.1	Step 2: installing using Homebrew . . . . .	89
6.3	Natural Language Translation Example using Google Service  . . . . .	92
6.3.1	How to get started using this api. . . . .	92
6.4	Implementation of A Hybrid Cloud natural Language Example  . . . . .	102
6.4.1	How to Implement using Command Line Interface and The Cloudmesh Catalog (AWS Provider) . . . . .	102
6.4.2	How to Implement using Command Line Interface and The Cloudmesh Catalog (Google Provider) . . . . .	103
6.4.3	heterogenous cloudmesh nlp service . . . . .	103
<b>7</b>	<b>AI</b>	<b>106</b>
7.1	DL Timeseries  . . . . .	106
<b>8</b>	<b>BIOS</b>	<b>106</b>
8.1	Bios  . . . . .	106
8.1.1	Paul Kiattikhunphan . . . . .	107
8.1.2	Alex Beck . . . . .	107
8.1.3	Alison Lu . . . . .	107
8.1.4	Jackson Miskill . . . . .	107
8.1.5	Jacques Fleischer . . . . .	108
8.1.6	Eric He . . . . .	108
8.1.7	Abdulbaqiy Diyaolu . . . . .	109

8.1.8	Thomas Butler . . . . .	109
8.1.9	Robert Knuuti . . . . .	109
8.1.10	Jake Kolessar . . . . .	110
<b>9</b>	<b>REFERENCES</b>	<b>110</b>

## 1 USEFUL LINKS

### 1.1 REU 2022

#### 1.1.1 Books

- <https://cloudmesh-community.github.io/pub/vonLaszewski-python.pdf>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-python.epub>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-linux.pdf>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-linux.epub>
- <https://cloudmesh.github.io/cloudmesh-mpi/report-mpi.pdf>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-writing.pdf>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-communicate.pdf>
- <https://cloudmesh-community.github.io/pub/vonLaszewski-reu2022.pdf>

#### 1.1.2 The repository for our report(s)

- <https://github.com/cybertraining-dsc/reu2022>

#### 1.1.3 The many UVA technical help ticket systems

- ITS: <https://virginia.edusupportcenter.com/shp/uva/helpcenter>
- Rivanna: <https://varesearchhelp.atlassian.net/servicedesk/customer/user/requests?page=1&reporter=all>
- BII: <https://uva-biocomplexity.atlassian.net/servicedesk/customer/user/requests?page=1&reporter=all&statuses=open&statuses=closed>

#### 1.1.4 Our technical slack

- <https://cloudmesh-reu2022.slack.com>

### 1.1.5 The administrative slack

- <https://biocomplexity-eo05671.slack.com/archives/C031CR0B2QG>

### 1.1.6 Rivanna Information

- Research Computing: <https://www.rc.virginia.edu/>
- Rivanna Overview: <https://www.rc.virginia.edu/userinfo/rivanna/overview/>
- Research Computing Support Center: <https://www.rc.virginia.edu/support/#office-hours>
- access through ssh: <https://www.rc.virginia.edu/userinfo/rivanna/login/>
- access through Web Browser: <https://shibidp.its.virginia.edu/idp/profile/SAML2/Redirect/SSO?execution=e2s1>
- Rivanna Slide Show: <https://learning.rc.virginia.edu/notes/rivanna-intro/>
- Globus Data Transfer (not using for reu2022) <https://www.rc.virginia.edu/userinfo/globus/>
- Slurm Job Manager Documentation <https://www.rc.virginia.edu/userinfo/rivanna/slurm/>
- Rivanna Allocations: <https://www.rc.virginia.edu/userinfo/rivanna/allocations/>

### 1.1.7 GitBash (Windows only)

- <https://git-scm.com/downloads>

### 1.1.8 Pycharm

- pycharm community edition: <https://www.jetbrains.com/pycharm/download>
- md: <https://www.jetbrains.com/help/pycharm/markdown.html>
- (optional) extension makefile: <https://plugins.jetbrains.com/plugin/9333-makefile-language>
- (optional) rst: <https://www.jetbrains.com/help/pycharm/restructured-text.html>
- 80 char: <https://intellij-support.jetbrains.com/hc/en-us/community/posts/206070859-How-do-I-enable-the-80-column-guideline-change>
- background to white as i can not read white on black when in meetings with me I will not support anyone that has black background: <https://www.jetbrains.com/help/pycharm/user-interface-themes.html> use intelliJ Light. After the meeting you can set it to whatever.

### 1.1.9 Bibliography Management

- jabref: <https://www.jabref.org/>
- bibtex: <https://en.wikipedia.org/wiki/BibTeX>

- draft bibtex from urls: <https://addons.mozilla.org/en-US/firefox/addon/bibitnow/> (has to be modified once copied. Not everything will work.

### 1.1.10 GitHub repo

- <https://github.com/cybertraining-dsc/reu2022>
- Sandra, JP: <https://github.com/cloudmesh/cloudmesh-mpi>

### 1.1.11 Only important for Gregor (do not use)

- request storage: <https://www.rc.virginia.edu/form/storage/>
- rivanna partition/account: <https://www.rc.virginia.edu/userinfo/rivanna/allocations/>
- managing groups: <https://mygroups.virginia.edu/>

## 2 INSTALL

### 2.1 Install



#### Learning Objectives

- Learn how to install python from python.org
  - Learn how to use a python venv
  - Learn how to install cloudmesh Shell
- 

In this section, we present an easy-to-follow installation guide for a recent version of python and cloudmesh.

Before you start, please read the entire section and develop a plan for installation.

### 2.1.1 Windows

The installation of cloudmesh benefits from using Git Bash as it allows us to have a terminal that is similar to that of macOS and Linux.

Hence, before we install python and cloudmesh we install Git Bash.

Furthermore, we provide the option to use chocolatey to install packages in similar fashion as on linus.

#### 2.1.1.1 Git Bash install

To install Git Bash, please download it first from

- <https://git-scm.com/downloads>

Click `Download` for Windows. The download will commence. Please open the file once it is finished downloading. Next, please start the downloaded program and follow the instructions carefully.

- The administration window (UAC Prompt) will appear. Click `Yes`. It will show you Git's license: a GNU General Public License. Read it and Click `Next`. To ensure security of the operating system, a UAC prompt allows operating systems, particularly Windows, to prompt for consent or credentials from local administrators before starting a program.
- Click `Next` to confirm that `C:\Program Files\Git` is the directory where you want Git to be installed.
- Select the box to create a shortcut icon on the desktop. Click `Next` to continue with the install.
- Click `Next` to accept the default text editor which is vim,
- Replace the default branch name (`master`) with `main`
- Click `Next` again to run Git from the command line and 3rd party software,
- Click `Next` again to use the OpenSSL library
- Click `Next` again to check out Windows-style,
- Click `Next` again to use MinTTY,
- Click `Next` again to use the default git pull,
- Click `Next` again to use the Git Credential Manager Core,
- Click `Next` again to enable file system caching, and then
- Click `Install` because we do not need experimental features.

A video tutorial on how to install Git and Git Bash on Windows 10 is located at [https://youtu.be/HCoTE\\_x\\_xCfA](https://youtu.be/HCoTE_x_xCfA)

A written tutorial on how to install Git and Git Bash on Windows 10 is located at <https://cybertraining-dsc.github.io/docs/tutorial/reu/github/git/>

### 2.1.1.2 Python 3.10 install

To install Python 3.10 please go to

- <https://python.org>

and download the latest version.

- Click `Download`. The download will commence. Please open the file once it is finished downloading
- Click the checkbox `Add Python 3.10 to PATH`
- Click `Install Now`
- At the end of the installation click the option to `Disable path length limit`

A video tutorial on how to install Professional PyCharm is located at <https://youtu.be/QPESX-VBnEU>

A video on how to configure PyCharm with cloudmesh is located at <https://youtu.be/eb1IQBx0D50>

### 2.1.1.3 Installing cloudmesh

Cloudmesh can be installed in any shell that has python and `git` access. However, it is convenient to use Git Bash as it simplifies the documentation and allows us to interact with Linux commands with the Windows file system. The installation is done with a Python virtual environment `ENV3` using command `venv` so you do not affect your computer's current python configurations or settings. Here are the steps:

```
$ python -m venv ~/ENV3
$ source ~/ENV3/Scripts/activate
$ cd
$ mkdir cm
$ cd cm
$ pip install pip -U
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
$ touch .bashrc
$ echo "source ~/ENV3/Scripts/activate" >> .bashrc
$ echo "cd ~/cm" >> .bashrc
```

To activate it, start new Git Bash and terminate the first Git Bash window. If you see in the new window `(ENV3)`, continue. Git Bash will initialize the environment.

If you do not want to always start in the directory `cm` do replace the line in your `.bashrc` `cd cm` with `cd`

### 2.1.1.4 Uninstall

To remove the virtual environment ENV3 , use the following command:

```
$ rm -f ~/ENV3
```

Next, edit the .bashrc and .bash\_profile file and delete the lines:

```
$ source ~/ENV3/Scripts/activate
$ cd cm
```

### 2.1.2 Choco install

There are a number of useful packages that you can install via choco. This includes Visual Code, Pycharm, Emacs, and make. Even Python could be installed with it; however, we have not tested the installation of python via choco, while we have tested the installation of Emacs, Pycharm, and make.

### 2.1.3 Install Chocolatey

To install, please start a Git Bash terminal as administrator: To do so press the Windows key and type powershell. Click Run as Administrator . Click Yes .

2. In PowerShell execute the following command:

```
PS C:\Windows\system32> Set-ExecutionPolicy AllSigned
```

Then type y .

3. Next type in the command (copy and paste to not make a mistake)

```
PS C:\Windows\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [
    ↘ System.Net.ServicePointManager]::SecurityProtocol = [System.Net.
    ↘ ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
    ↘ System.Net.WebClient).DownloadString('https://community.chocolatey.org/
    ↘ install.ps1'))
```

4. Wait for the installation to complete; once you see

```
PS C:\Windows\system32>
```

with a blinking cursor again, and lines have stopped appearing, then the Chocolatey installation has finished. Type choco and you should see Chocolatey in green text.

Now you can install many programs with choco by launching PowerShell as Administrator or Git Bash.

A list of programs that you can install with choco can be found at:

- <https://community.chocolatey.org/packages/>

### 2.1.4 Installing Useful Developer Programs

The following useful developer programs can be installed. Select the ones you like and install them with the appropriate command

```
$ choco install make -y  
$ choco install emacs -y  
$ choco install pycharm -y  
$ choco install firefox -y  
$ choco install vscode -y  
$ choco install zoom -y
```

Once the installation completes, your program will be ready for you to use.

### 2.1.5 Linux

#### 2.1.5.1 Install Python 3.10.5

The installation from source can be done easily as shown:

```
$ mkdir -p ~/tmp  
$ cd ~/tmp  
$ wget https://www.python.org/ftp/python/3.10.5/Python-3.10.5.tar.xz  
$ tar xvf Python-3.10.5.tar.xz  
$ cd Python-3.10.5/  
$ ./configure --enable-optimizations  
$ make -j $(nproc)  
$ sudo make altinstall  
$ pip install pip -U  
$ python3.10 -V
```

#### 2.1.5.2 Setting up the venv

We assume you use bash

```
$ python3.10 -m venv ~/ENV3  
$ source ~/ENV/bin/activate  
$ cd  
$ mkdir cm  
$ cd cm  
$ pip install cloudmesh-installer  
$ cloudmesh-installer get cmd5  
$ cms help  
$ touch .bashrc
```

```
$ echo "source ~/ENV3/bin/activate" >> .bashrc
$ echo "cd cm" >> .bashrc
$ echo "source ~/ENV3/bin/activate" >> .bash_profile
$ echo "cd cm" >> .bash_profile
```

### 2.1.5.3 Uninstall

```
$ rm -f ~/ENV3
```

Edit the `.zshrc` and `.zprofile` file and delete the lines

```
$ source ~/ENV3/bin/activate
$ cd cm
```

**2.1.5.4 Update** In case you need to update the Python version it is sufficient to follow the instructions provided in the section `Install Python 3.10.5`, while replacing the version number with the current python release number.

In case you need to create a new virtual ENV3. You can first uninstall it and then reinstall it.

An easy way to do all of this with a command is the following:

```
$ cd ~/cm
$ pip install cloudmesh-installer -U
$ pip install --upgrade pip
$ cloudmesh-installer new ~/ENV3 cmd5 --python=/usr/local/bin/python3.10
$ source ~/ENV3/bin/activate
$ python -V
$ which python
```

## 2.1.6 macOS

We assume you use `~/zsh` which is the default on macOS

**2.1.6.1 Xcode Install** There are a number of digital tools that are needed before proceeding further. These tools include git, make, and a c compiler. All of these tools can be downloaded at [Xcode](#), which is an IDE App on the Apple App Store that includes all of these necessary elements.

Once installed, there is one simple command line command to run:

```
$ xcode-select --install
```

This will install all the necessary command line tools. Xcode can be used as an IDE, but for the most part will not be used outside the command line tools it provides.

### 2.1.6.2 Cloudmesh

**2.1.6.2.1 Install** Before any of the following, make sure to download the current version of python. At the time of this writing, it is python 3.10.5.

Second, execute the following commands in your terminal. Make sure to do this in order.

```
$ cd  
$ python3.10 -m venv ~/ENV3  
$ source ~/ENV3/bin/activate  
$ pip install pip -U  
$ mkdir cm  
$ cd cm  
$ pip install cloudmesh-installer  
$ cloudmesh-installer get cmd5  
$ cms help  
$ echo "source ~/ENV3/bin/activate" >> .zshrc  
$ echo "cd cm" >> .zshrc  
$ echo "source ~/ENV3/bin/activate" >> .zprofile  
$ echo "cd cm" >> .zprofile
```

It creates the virtual environment, a directory called `~/cm`, then installs `cloudmesh`. Following this, it sets the macOS startup commands `.zshrc` and `.zprofile` to start up in the virtual environment `~/ENV3`.

### 2.1.6.2.2 Uninstall

```
$ rm -rf ~/ENV3
```

You may need to enter your system password.

**2.1.6.3 Updating Python** Before starting this process, ensure that python is in the correct path. Test in the terminal.

To do so remove the existing ENV3 first and start a new terminal in which you will be working.

```
$ rm -rf ~/ENV3
```

Start the new terminal and execute the commands to verify if you have the right updated version of python

```
$ which python3.10
$ where python3.10
$ python3.10 -V
```

Now execute:

```
$ cd ~/cm
$ python3.10 -m venv ~/ENV3
$ source ~/ENV3/bin/activate
$ pip install pip -U
$ pip install cloudmesh-installer
$ cloudmesh-installer get cmd5
$ cms help
```

As `~/zsh` is already configured previously, we do not have to set it up again.

**2.1.6.4 Homebrew install** Homebrew is a package management software. The Homebrew command is called `brew`. It is used to install Linux packages on macOS. Installing `brew` is simple.

- First, make sure the computer that is downloading Homebrew is up-to-date with the latest software for its OS.
- Second, ensure that `xcode` has been installed. `xcode` can be installed from the Apple App Store.
- Third, in the terminal, write out:

```
$ xcode-select --install
```

This installs `xcode` command line tools.

- Fourth, run the following command in the terminal:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install
  ↪ /HEAD/install.sh)"
```

- Fifth, enter the administrator password into the desired location.
- It may take a moment for the software to install, but it will eventually say “**Installation successful!**” in the terminal. After that, Homebrew is installed onto the device.

After the user has correctly installed Homebrew, it is simple to install packages directly to the operating system:

```
$ brew install [package name]
```

## 2.2 Ramdisk

---



### Learning Objectives

- Learn how to set up a RAM disk
  - Learn how to use a DAM disk
- 

how to set up and manage a ramdisk on linux

develop cms program

cms ramdisk -... -size=SIZE

use humanize so we can use 1GB for size ...

showcase

- a) dynamic ramdisk no reboot needed, but if reboot, ramdisk needs to be set up new
- b) ramdisk integrated in fstab with reboot
- c) backup and load ramdisk

On macOS a RAM disk with 512MB space can be created with the following command:

```
n = 512 * 2048
os.system('diskutil eraseVolume HFS+ "RAMDisk" `hdiutil attach -nomount ram://{n}`')
```

### 2.2.1 Ubuntu

On Ubuntu, a RAM disk and its read-only shadow can be created by:

```
mount -t tmpfs -o size=512m tmpfs /mnt/ramdisk
mount -t aufs -o br:/mnt/ramdisk=ro none /mnt/readonly

# mkdir /tmp/ramdisk; chmod 777 /tmp/ramdisk
# mount -t tmpfs -o size=256M tmpfs /tmp/ramdisk/
```

using /dev/shm:

<http://ubuntuguide.net/ubuntu-using-ramdisk-for-better-performance-and-fast-response>

Various methods: (in german): [https://wiki.ubuntuusers.de/RAM-Disk\\_erstellen/](https://wiki.ubuntuusers.de/RAM-Disk_erstellen/)

ramfs is an older file system type and is replaced in mostly by tmpfs.

## 2.2.2 windows

<https://forums.guru3d.com/threads/guide-using-imdisk-to-set-up-ram-disk-s-in-windows-with-no-limit-on-disk-size.356046/>

## 3 GRAPH VIZUALIZATION

matplotlib	seaborn	bokeh	pandas
charts			
barchart	<code>plt.bar</code>	<code>sns.barplot</code>	<code>p.vbar</code>
grouped	<code>plt.bar</code>	<code>sns.catplot</code>	<code>-</code>
bar- chart			<code>df.plot.bar</code>
stacked	<code>plt.bar</code>	<code>-</code>	<code>df.plot.bar(stacked=True)</code>
bar- chart			
spline- chart	<code>-</code>	<code>-</code>	<code>-</code>
multiline	<code>plt.plot</code>	<code>sns.lineplot</code>	<code>p.multi_line</code>
chart			<code>df.plot.line</code>
compound line chart	<code>-</code>	<code>sns.lineplot</code>	<code>df.plot.area</code>
histogram	<code>plt.hist</code>	<code>sns.histplot</code>	<code>-</code>
linechart	<code>plt.plot</code>	<code>sns.lineplot</code>	<code>p.line</code>
piechart	<code>plt.pie</code>	<code>-</code>	<code>df.plot.pie</code>

matplotlib	seaborn	bokeh	pandas
exploded pie(explode=)	-	-	-
piechart			
donut pie(wedgeprops=)	-	Donut	-
countplot	sns.kdeplot	-	-
distribution plot	sns.displot	Histogram	-
point chart	sns.pointplot	-	df.plot.scatter
scatter plot	sns.scatterplot	Scatter	df.plot.scatter
bubble chart	-	-	df.plot.scatter(s=..., c=...)
radar chart	-	-	-
boxplot	sns.boxplot	Boxplot	df.plot.boxplot
features			
pdf + ex-port		-	
png + ex-port	via matplotlib	export_png	via matplotlib
svg + ex-port	via matplotlib	export_svg	via matplotlib
color palettes	sns.color_palette	Color Palettes	-
interactive graph	-	+	-
data via Pandas frame	via Pandas	via Pandas	link missing

### 3.1 Python Data Management for Visualizations

In python, there are several ways that data can be interpreted in order to generate the graphics for data visualization.

Through several examples, we will show how to manage different types of data and how to best interact with them. They are lists, dictionaries and CSV files.

#### 3.1.1 Lists

Lists are clumps of continuous values that are put directly next to each other in the computer memory system.

**3.1.1.1 Construction** In python, lists can be constructed like this:

```
example_list = ['example', 2, 'b', 45, False]
```

Lists have order and can hold many types (bool, int, String, etc.) of values.

**3.1.1.2 Accessing Values** In python, it is easy to access values within a list. The following code provides an example of how to access certain values within a list:

```
index_4 = example_list[4]
print(index_4)
# output is False
```

It is important to note that lists have indices, which range from 0 to the length of the list - 1. The indices provide access to values within the list.

**3.1.1.3 Updating Values** To update a value within a list, simply utilize same brackets:

```
example_list[3] = 'bear'
print(example_list[3])
# output is 'bear' instead of 45
```

This will change the value at the third index from 45 to bear.

**3.1.1.4 Python Built-In Methods** Python has several built-in methods for lists. These include `append()`, `clear()`, `copy()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, and `sort()`.

A user can utilize these methods to make changes in the necessary ways to the list.

### 3.1.2 Dictionaries

Dictionaries are like specialized lists. They hold a key-value pair that allows for a user to look up a key and find the associated value. Dictionaries are useful for storing values in a way that is more organized than a linear list. Furthermore, dictionaries make it easy for users to look up the necessary information.

**3.1.2.1 Construction** In python, dictionaries are constructed as follows:

```
example_dictionary = {'motorcycles': 2, 'autocycles': 3,
                      'cars': 4, 'small_trucks': 6
                      'large_trucks': '18'}
```

The string values are the keys, which provide access to the values within the dictionary. The colon provides the computer with the command for assigning key-value pair.

**3.1.2.2 Accessing Values** There are several built-in commands to access both the keys and values in a dictionary. They are as follows:

```
example_dictionary.get()
example_dictionary.keys()
example_dictionary.values()
```

The `.get()` method returns the value that is associated with the given key, the `.keys()` method returns a list of the keys alone, and the `.values()` method returns a list of the values alone. There are more methods (see the Python Built-In Methods section)

**3.1.2.3 Updating Values** To update the dictionary, there is one method that can be used:

```
example_dictionary.update({'motorcycles': 20})

# or to add a new key-value pair to the dictionary:

example_dictionary['New Key'] = 'New Value'
```

This will update the value associated with this particular key-value pair.

**3.1.2.4 Python Built-In Methods** There are several built-in methods that allow for more dictionary manipulation: They are `clear()`, `copy()`, `fromkeys()`, `items()`, `pop()`, `popitem()`, and `setdefault()`.

### 3.1.3 CSV Files

CSV stands for *comma-separated-values* and is a data structure that is incredibly common for data management and analysis. There are many ways to access CSV files. The most common way is to use pandas, a python library. However, python also has a built-in CSV module that can be used. We will show examples of using both below.

**3.1.3.1 Installing and Importing** Before beginning with any of these CSV manipulation tasks, it is necessary to install and import the correct modules and libraries. The following showcases this:

```
$ pip install pandas
```

```
import pandas as pd
import csv
```

**3.1.3.2 Construction** CSV files are files that exist elsewhere and have already been created. Therefore, for creation, we are not creating a CSV files, but rather deconstructing it into something that is usable.

In pandas, this means creating a dataframe, which is essentially and indexed table. In the python `csv` module, this means using the `csvreader` object within the module. Following are two examples showcasing how exactly to do this.

For the `csv` module:

```
# open the csv and creates the reader object for it
file = open('/Users/jacksonmiskill/Downloads/biostats.csv')
reader = csv.reader(file)
```

The `reader` is an object that was created by python developers to help parse through the `csv` files.

For the `pandas` module:

```
file = pd.read_csv("/Users/jacksonmiskill/Downloads/biostats.csv")
df = pd.DataFrame(file)
```

### 3.1.3.3 Accessing Values

For the `csv` module:

It is more challenging to access files using the `csv` module as opposed to the `pandas` library. To make it more simple, it is necessary to convert the values that lie within the `csv` into a list in order to access.

```
data = []
for each_row in reader:
    data.append(each_row[2])

print(data) # output is the whole list
```

This code is available on [GitHub](#)

For the `pandas` module, it is less complicated to access that values. This is because the module includes a function that converts the data into a dataframe. After converting, you can use the various methods that are within the dataframe to essentially pass in the correct values.

### 3.1.3.4 Examples

Once the `csv` values have been accessed, creation of the graphics can begin. Starting with the `csv` module and then moving into `pandas` the following will demonstrate this action.

The `csv` file that will be utilized for the following examples can be found [here](#). It represents made up data on a group of made up people such as age, height, and weight.

It is slow and complicated to create graphs with the `csv` module. We have implemented a separate module called `ConvertCSV` which provides the user with the ability to convert the data received from the `csv` module into doubly nested lists, lists, and dictionaries, depending on necessity.

```
# reading in the data!
filename = path_expand("./biostats")
table = ConvertCSV.create_table(filename)
print(table)

list_names = []
list_heights = []

list_names = ConvertCSV.csv_read_to_list(table=table, index=0, convert=False)
list_heights = ConvertCSV.csv_read_to_list(table=table, index=3, convert=True)
```

```

dict_names = []
dict_heights = []

idict = ConvertCSV.csv_read_to_dict(table=table, index=3)
dict_names = list(idict.keys())
dict_heights = list(idict.values())

# now just plot these values. You can use whichever library you desire
# for the example, we use matplotlib, because it is simple

# list
plt.plot(list_names, list_heights)
plt.xlabel("Names")
plt.ylabel("Heights")
plt.xticks(rotation=90)
plt.savefig('images/csv-list-lineplot.png')
plt.savefig('images/csv-list-lineplot.svg')
plt.savefig('images/csv-list-lineplot.pdf')
plt.title("Names and Corresponding Height")
plt.show()

# dictionary
plt.plot(dict_names, dict_heights)
plt.xlabel("Names")
plt.ylabel("Heights")
plt.xticks(rotation=90)
plt.savefig('images/csv-dict-lineplot.png')
plt.savefig('images/csv-dict-lineplot.svg')
plt.savefig('images/csv-dict-lineplot.pdf')
plt.title("Names and Corresponding Height")
plt.show()

```

This code can be access from [GitHub](#)

This code produces Figure ??, and Figure Figure ??.:

{#fig:csv-list-lineplot width=50%}

Figure ??: created using the data available [here](#).

{#fig:csv-dict-lineplot width=50%}

Figure ??: created using the data available [here](#).

However, it is so much more simple to accomplish this with the `pandas` library:

```
file = pd.read_csv("/Users/jacksonmiskill/Downloads/biostats.csv")
```

```
biostats = pd.DataFrame(file)

plt.plot(biostats['Name'], biostats['Height (in)'])
plt.xlabel("Name")
plt.ylabel("Height")
plt.xticks(rotation=90)
plt.savefig('images/pandas-lineplot.png')
plt.savefig('images/pandas-lineplot.svg')
plt.savefig('images/pandas-lineplot.pdf')
plt.title("Names and Corresponding Height")
plt.show()
```

This code can be accessed from [GitHub](#)

This code produces the Figure ??:

{#fig:pandas-lineplot width=50%}

Figure ??: created using the data available [here](#).

### 3.1.4 Links

#### 3.1.4.1 Lists

- <https://towardsdatascience.com/python-basics-6-lists-and-list-manipulation-a56be62b1f95>
- [https://www.w3schools.com/python/python\\_ref\\_list.asp](https://www.w3schools.com/python/python_ref_list.asp)

#### 3.1.4.2 Dictionaries

- <https://www.pythonforbeginners.com/dictionary/dictionary-manipulation-in-python>
- [https://www.w3schools.com/python/python\\_ref\\_dictionary.asp](https://www.w3schools.com/python/python_ref_dictionary.asp)
- [https://www.w3schools.com/python/ref\\_dictionary\\_update.asp](https://www.w3schools.com/python/ref_dictionary_update.asp)

#### 3.1.4.3 CSV Files

- <https://www.geeksforgeeks.org/creating-a-dataframe-using-csv-files/>
- <https://docs.python.org/3/library/csv.html#examples>
- <https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html>
- <https://docs.python.org/3/library/functions.html#open>
- <https://www.protechtraining.com/blog/post/python-for-beginners-reading-manipulating-csv-files-737#extracting-information-from-a-csv-file>
- <https://stackoverflow.com/questions/13039392/csv-list-index-out-of-range>
- <https://www.geeksforgeeks.org/visualize-data-from-csv-file-in-python/>

### 3.2 Python Graphics

---



#### Learning Objectives

- Introduction to plotting libraries
    - Introduction to matplotlib
    - Introduction to seaborn
    - Introduction to bokeh
    - Introduction to pandas plots
  - Introduction to graph plotting libraries
    - Introduction to networkX
    - Introduction to garphvis
    - Introduction to mermaid
    - Introduction to rackdiag
  - Identify which library to chose
- 

In Python, data and equations can be visually represented using graphs and plots. Here we showcase how to use the different plotting libraries Matplotlib, Bokeh, and Seaborn.

For each of these frameworks exist an extensive example set as part of Galleries included with the original documentation. We encourage to browse through these examples to identify plots that you may want to generate.

- [matplotlib gallery](#)
- [seaborn gallery](#)
- [bokeh gallery](#)
- [pandas gallery](#)

Another combined gallery is available as

- (interactive selection)[<https://www.python-graph-gallery.com/>]

And provides you with choices based on your selection.

### 3.2.1 Matplotlib

Matplotlib is the main plotting library that allows the user to visualize data. Matplotlib creates figures that can be manipulated and transformed. This includes manipulations of axes, labels, fonts, and the size of the images.

#### 3.2.1.1 Installation

To install matplotlib, please use the command:

```
$ pip install matplotlib
```

#### 3.2.1.2 Import Statements

The user will need to supply these import statements at the top of their code in order for Matplotlib to be imported.

```
import matplotlib.pyplot as plt
import numpy as np
```

#### 3.2.1.3 Bar Chart

In Matplotlib, it is easy to create bar charts. For example, this is a demonstration of a simple bar chart using data from a user using Spotify.

```
import matplotlib.pyplot as plt

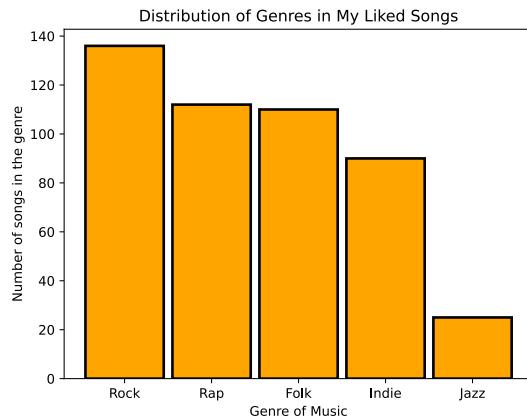
# you can also do this: from matplotlib import pyplot as plt

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
categories = data.keys()
count = data.values()

# Creating the bar chart
plt.bar(categories,
        count,
        align='center',
        color='darkorange',
        width=0.4,
        edgecolor="royalblue",
        linewidth=4)

# Editing the bar chart's title, x, and y axes
plt.xlabel("Genre of Music")
plt.ylabel("Number of songs in the genre")
plt.title("Distribution of Genres in My Liked Songs")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 1.



**Figure 1:** Matplotlib Barchart created from data from Spotify.

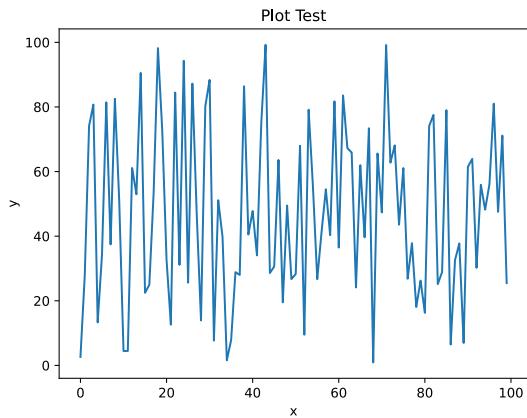
**3.2.1.4 Line Chart** The Matplotlib library in python allows for comprehensive line plots to be created. Here a line chart was created using a for loop to generate random numbers in a range and plot it against the `x` and `y` axis to display the changes between two variables/data sets.

```
import matplotlib.pyplot as plt
import random

x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)

# creating the plot and labeling axes and title
plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plot Test")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 2.



**Figure 2:** Matplotlib Linechart created from random variables

**3.2.1.5 Pie Chart** A pie chart is most commonly used when representing the division of components that form a whole thing e.g. showing how a budget is broken down into separate spending categories. In Matplotlib, the function `pie()` creates a pie chart. In the following code example, a user's Spotify data will be displayed as a pie chart.

```
import matplotlib.pyplot as plt

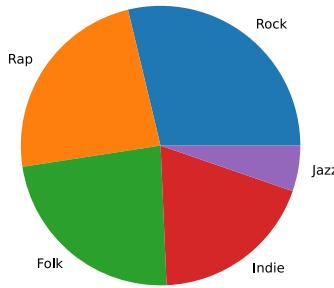
data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
categories = data.keys()
count = data.values()

# Creating the pie chart
plt.pie(count, labels=categories)

plt.show()
```

This program can be downloaded from [GitHub](#)

The output of this program is showcased in Figure 3.



**Figure 3:** Barchart created from data from Spotify

**3.2.1.6 Contour Plot** Unlike the previous types of plots shown, contour plots allow data involving three variables to be plotted on a 2D surface. In this example, an equation of a hyperbolic paraboloid is graphed on a contour plot.

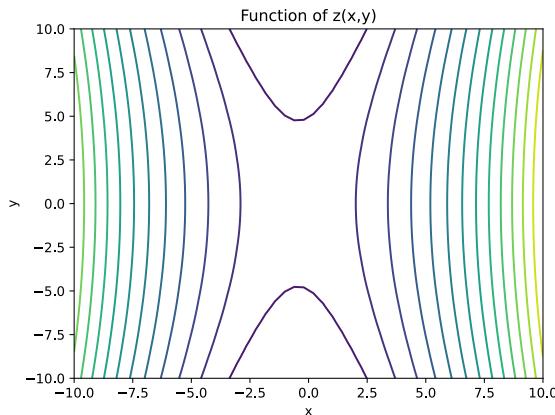
```
import matplotlib.pyplot as plt
import numpy as np

# creating an equation for z based off of variables x,y
x, y = np.meshgrid(np.linspace(-10, 10), np.linspace(-10, 10))
z = 9 * (x ** 2 + 1) + 8 * x - (y ** 2)
levels = np.linspace(np.min(z), np.max(z), 15)

# creating a contour graph based off the equation of z
plt.contour(x, y, z, levels=levels)

plt.xlabel("x")
plt.ylabel("y")
plt.title("Function of z(x,y)")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 4.



**Figure 4:** Multivariable (x, y, z) Equation Plotted

A contour plot allows data and equations consisting of three variables to be plotted through plotting 3D surfaces as 2D slices on a `xy` plane. Matplotlib can display data and equations through contour graphs after they are inputted. Shown below are the parameters for `plt.contour`.

```
plt.contour([x, y], z, levels)
```

The independent variables `x` and `y` must be defined so the dependent variable `z` can be defined. The variables can come in the form of a list or dictionary or as an equation. The `levels` parameter determines the number of contour lines that can be drawn.

### 3.2.1.7 Titles, Labels, and Legends

**3.2.1.7.1 Titles** Titles are necessary to let the reader know about your graph or plot is exactly about. To give a title to your whole graph in matplotlib, simply type:

```
plt.title("Title you want to set").
```

**3.2.1.7.2 x-axis labels and y-axis labels** Within the Matplotlib library are the functions `plt.xlabel()` and `plt.ylabel()`. All these functions do is set a string to the two axes. To use these functions, simply type:

```
plt.xlabel("Label you want to set")
plt.ylabel("Label you want to set")
```

**3.2.1.7.3 Legend** Sometimes, a legend may be necessary to let the reader know which part of the graph/plot corresponds to each part of the data shown. To show a legend, use the command:

```
plt.legend()
```

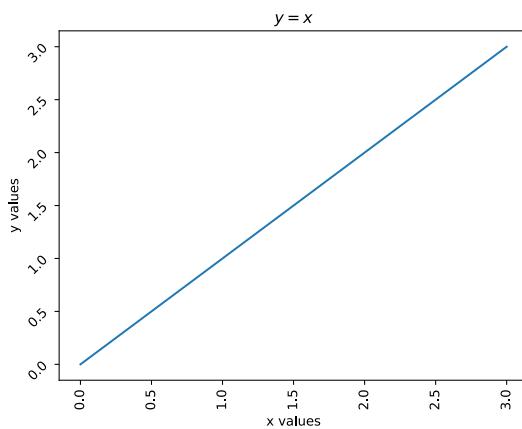
**3.2.1.8 Rotating Ticks** When a chart is created, ticks are automatically created on the axes. By default, they are set horizontally; however, they can be rotated using `plt.xticks(degrees)` for the x-axis or `plt.yticks(degrees)` for the y-axis. This can be shown by this simple [example](#). The output is shown in Figure 5.

```
import matplotlib.pyplot as plt

x = range(0, 4)
y = x
plt.plot(x, y)

# Rotating Ticks
plt.xticks(rotation=90)
plt.yticks(rotation=45)

plt.xlabel('x values')
plt.ylabel('y values')
plt.title(r'$y=x$')
plt.show()
```



**Figure 5:** x-axis ticks rotated by 90° and y-axis ticks rotated by 45 degrees

### 3.2.1.9 Exporting

**3.2.1.9.1 Saving Chart as Files After a chart is created and displayed, it** can be exported as a file outside the code using this command:

```
plt.savefig("fname", dpi='figure')
```

The name and format of the file are set as a string using `fname`. Make sure to specify the format of the file by using a `.` after the file name and specify the type after such as `.pdf`, `.png`, `svg`, etc.

The parameter `dpi` sets the DPI (Dots per Inch) of the image being saved. Specify this number in the form of a float. For example, set `dpi=300`.

Additionally, there is another way to save files that may be faster than calling a specific method for each file. The following code showcases this:

```
import matplotlib.pyplot as plt
import os
from matplotlib import pyplot

def save():
    name = os.path.basename(__file__).replace(".py", "")
    plt.savefig(f'/filepath/{name}.png')
    plt.savefig(f'/filepath/{name}.pdf')
    plt.savefig(f'/filepath/{name}.svg')
    plt.show()
```

This code can be accessed on [GitHub](#)

**3.2.1.9.2 Display** The very last command that should be written is `plt.show()`, as this command displays the graph that you made. To show, simply type:

```
plt.show()
```

### 3.2.1.10 Links

- <https://matplotlib.org/>
- [https://matplotlib.org/stable/api/pyplot\\_summary.html](https://matplotlib.org/stable/api/pyplot_summary.html)
- <https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/>
- <https://www.geeksforgeeks.org/bar-plot-in-matplotlib/>

### 3.3 Seaborn

Seaborn, like Matplotlib, is a data visualization tool. However, the graphs and charts that Seaborn can create are more complex than Matplotlib. The graphs that are created in Seaborn are more statistically detailed. Unlike matplotlib, Seaborn draws upon other imported libraries such as Matplotlib, Numpy, and Pandas. This is because Seaborn relies on more complex math (Numpy) and data frames (generated from Pandas) that are passed into its functions as the data.

Several types of plots can be made from Seaborn; they are relational, distributional, categorical, regression, and matrix plots.

We have created examples to demonstrate the abilities of Seaborn.

#### 3.3.1 Installation

Seaborn can be installed in the same way as the other libraries installed earlier. The user who is installing the library should make sure that it is being installed in the correct environment.

```
$ pip install seaborn
```

#### 3.3.2 Import Statements

The user will need to supply these import statements at the top of their code in order for Seaborn to be imported. Additionally, the data created for the examples represents a user's Liked songs from Spotify.

```
import seaborn as sns
import matplotlib.pyplot as plt

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
categories = list(data.keys())
count = list(data.values())
personal_rank = [3, 4, 2, 1, 5]
```

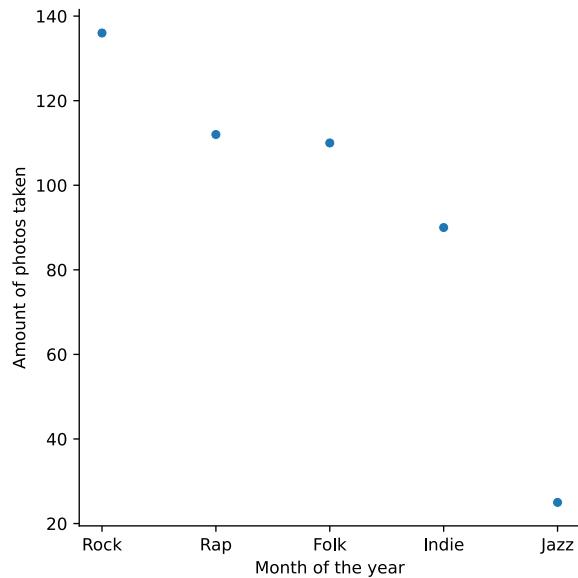
#### 3.3.3 Relational Plots

Relational plots showcase the relationship between variables in a visual format. It is a broad term for data representation. Examples of relational plots in Seaborn are `relplot` `lineplot` and `scatterplot`.

It is simple to create a relational plot with Seaborn:

```
sns.relplot(x=months, y=photos)
plt.xlabel("Month of the year")
plt.ylabel("Amount of photos taken")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 6.



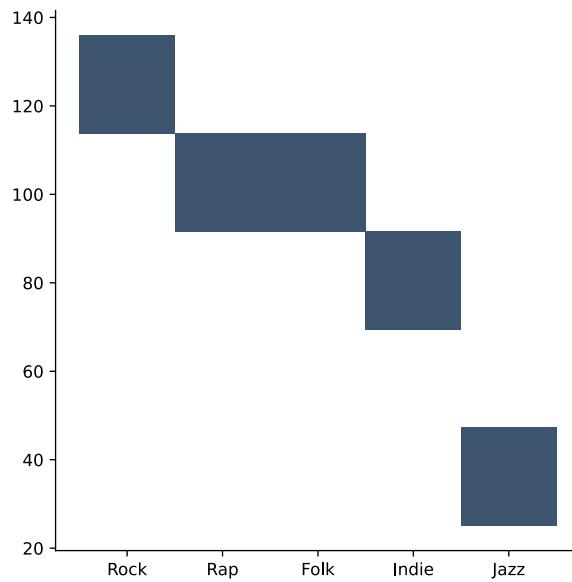
**Figure 6:** Seaborn Lineplot created from user Spotify data.

### 3.3.4 Distribution Plots

A distribution plot shows how the data is concentrated in a range of values. The graph that appears looks similar to a bar graph in that there are bars. However, these bars show the concentration of a variable across a range of values rather than the quantity possessed by a singular variable. The distributional plots in Seaborn are `displot` `histplot` `kdeplot` `ecdfplot` and `rugplot`.

```
sns.displot(x=source, y=value)
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 7



**Figure 7:** Seaborn Displot created from user Spotify data

### 3.3.5 Categorical Plots

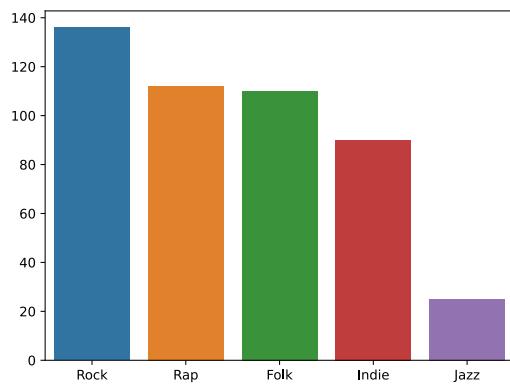
Categorical plots are statistical graphs that help visualize the magnitudes of different variables in a dataset. A type of categorical plot is a bar chart, exactly like the example produced in the Matplotlib section. The categorical plots are `catplot` `stripplot` `swarmplot` `boxplot` `violinplot` `boxenplot` `pointplot` `barplot` and `countplot`.

Categorical plots are relatively simple to implement. If using the `catplot` method, it is necessary to include the `kind` parameter.

```
sns.barplot(x=source, y=value)
plt.show()
```

This program can be downloaded from [GitHub](#)

The output from the program is showcased in Figure 8



**Figure 8:** Seaborn Catplot created from user Spotify data.

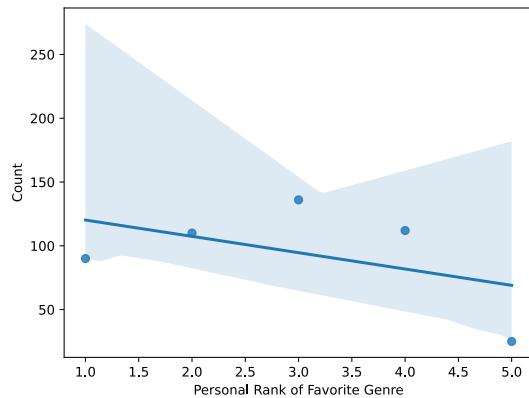
### 3.3.6 Regression Plots

Regression plots are like relational plots in the way that they help visualize the relationship between two variables. Regression plots, however, show the linear correlation that may or may not exist in a scatter plot of data. Their regression plots are `lmplot` `regplot` and `residplot`.

Regression plots are simple to implement:

```
sns.regplot(x=months, y=photos)
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure 9.



**Figure 9:** Seaborn regplot created from user Spotify data

Each of these plots can be manipulated to the users needs via the API that is listed in the sources section.

### 3.3.7 Saving Figures

Saving figures created by Seaborn is quite simple. This is because it is the exact same as in Matplotlib.

To save a figure:

```
plt.savefig('figure_path/figure_name')
```

This program can be downloaded from [GitHub](#)

### 3.3.8 Links

- <https://seaborn.pydata.org/api.html>
- <https://www.geeksforgeeks.org/python-seaborn-tutorial/>
- <https://www.geeksforgeeks.org/introduction-to-seaborn-python/>
- <https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/>
- <https://stackoverflow.com/questions/30336324/seaborn-load-dataset>
- <https://github.com/mwaskom/seaborn-data/blob/master/planets.csv>

## 3.4 Bokeh

Bokeh is a Python library useful for generating visualizations for web browsers. It generates graphics for all types of plots and dashboards powered by JavaScript without the user's need to write any JavaScript code. The guide below will walk you through useful Bokeh commands and features.

### 3.4.1 Installation

To install Bokeh, please use the command:

```
$ pip install bokeh
```

### 3.4.2 Import Statements

To plot figures, we import the `show` and `figure` functions from the Bokeh libraries.

```
from bokeh.io import show
from bokeh.plotting import figure
```

### 3.4.3 Bokeh Plotting Interface

`bokeh.plotting` is the library's main interface. It gives the ability to generate plots easily by providing parameters such as axes, grids, and labels. The following code shows some of the simplest examples of plotting a line and a point on a chart.

```
from bokeh.io import show
from bokeh.plotting import figure

# labeling the title, specifying the range of the x-axis, labeling the
# y-axis, specifying the height to be 500 pxls

p = figure(title="My Graph", x_range=[0, 20], y_axis_label="the y axis",
           height=500)

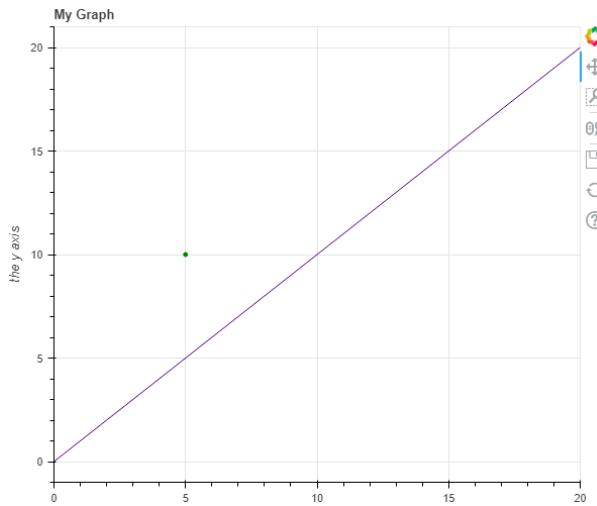
# plotting a line from (0,0) to (20,20); any of the CSS colors can be
# used

p.line([0, 20], [0, 20], color='indigo')

# plotting a point (circle) at (5,10)
p.circle(5, 10, color='green')

show(p)
```

This program can be downloaded from [GitHub](#). The output is shown in Figure 10



**Figure 10:** Figure created with Bokeh.

#### 3.4.4 Figure Parameters Example

- **x\_axis\_label** and **y\_axis\_label**: labels for the x and y axis
- **x\_range** and **y\_range**: specifications for the range of the x
- **and y axis title**: text title for your graph width\*\* and
- \*\*\*\*height\*\*: width and height of your graph in pixels
- **background\_fill\_color**: the background of the figure (takes any
- \*\*CSS colors)

#### 3.4.5 Scatter Plot

The Bokeh library provides various marker shapes for marking points on the scatter plot. The example below demonstrates how to create a scatter plot with two points at locations (1,3) and (2,4) respectively with circular and square marker shapes. The size parameter controls the size of the marker.

```
from bokeh.io import show
from bokeh.plotting import figure

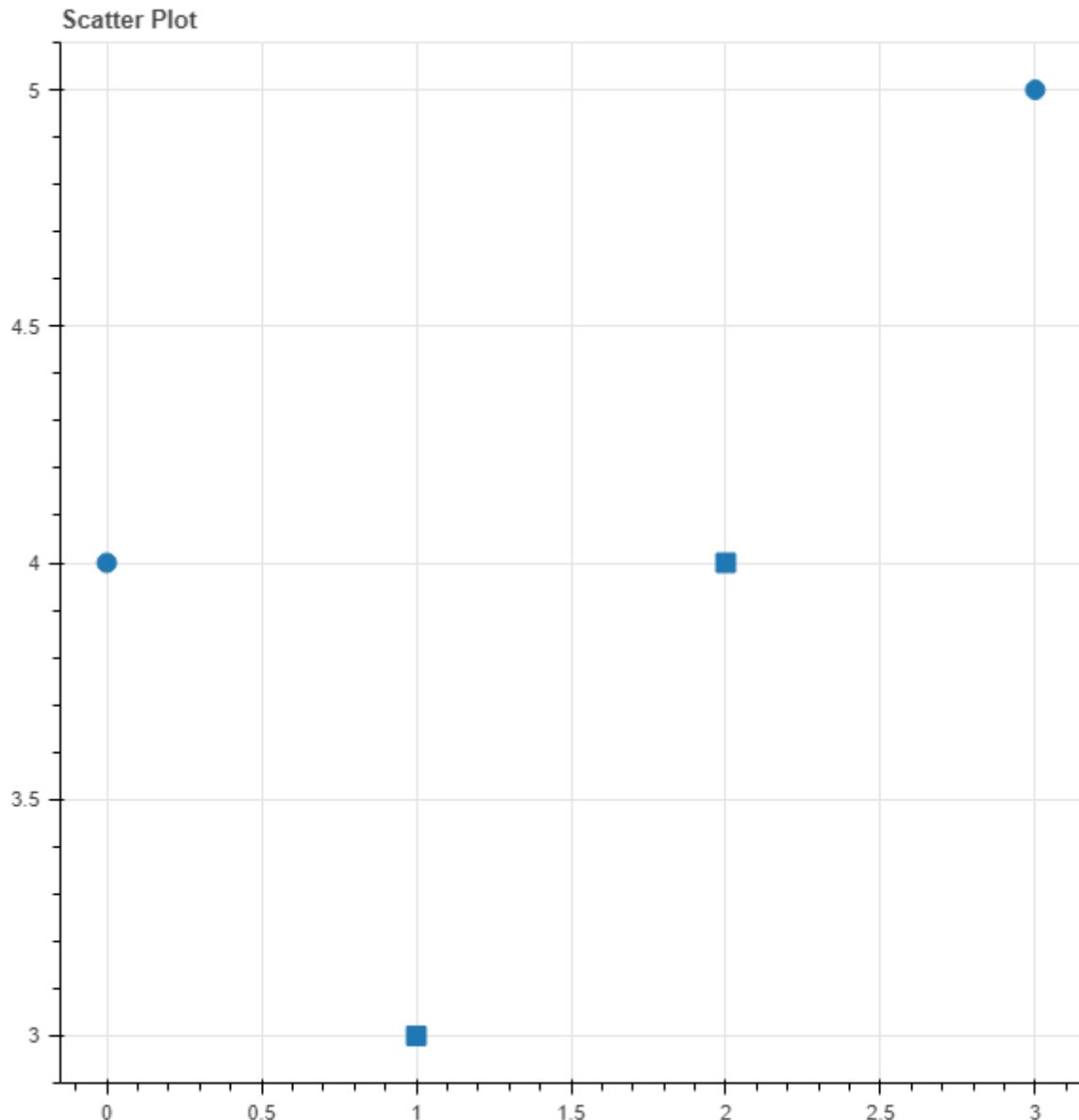
p = figure(title="Scatter Plot")

# Circle
p.circle([0, 3], [4, 5], size=10)

# Square
p.square([1, 2], [3, 4], size=10)
```

```
show(p)
```

This program can be downloaded from [GitHub](#). The output is shown in Figure 11



**Figure 11:** Scatter Plot created with user Spotify data.

The list possible marker types and the functions used to create them can be found [here](#).

### 3.4.6 Line Plots

The library provides a series of functions for creating various types of line graphs ranging from a single line graph, step line graph, stacked line graph, multiple line graph, and so on.

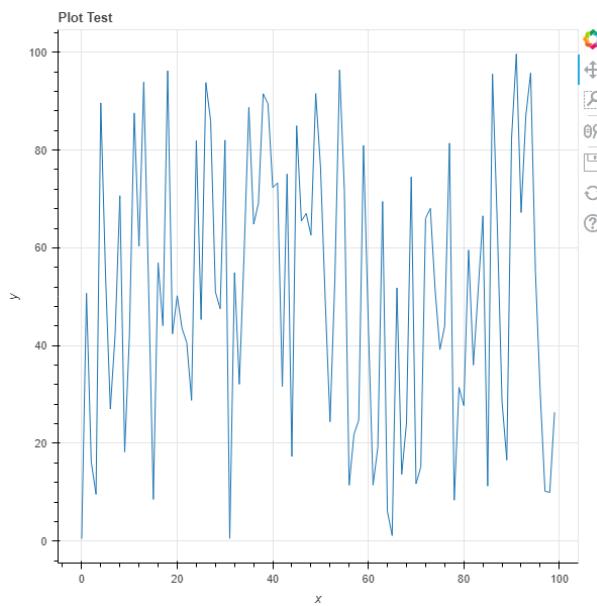
```
from bokeh.io import show, export_png, export_svg
from bokeh.plotting import figure
import random

x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)

p = figure(title="Plot Test", x_axis_label="x", y_axis_label="y")
p.line(x, y)

show(p)
```

This program can be downloaded from [GitHub](#)



**Figure 12:** Line Chart created with user Spotify data.

You can find the source code for other types of line plots here: [http://docs.bokeh.org/en/latest/docs/user\\_guide/plotting.html](http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html)

### 3.4.7 Bar Chart

Similarly, the `hbar()` and `vbar()` functions can be used to display horizontal and vertical bar graphs, respectively.

```
from bokeh.io import show, export_png, export_svg
from bokeh.plotting import figure

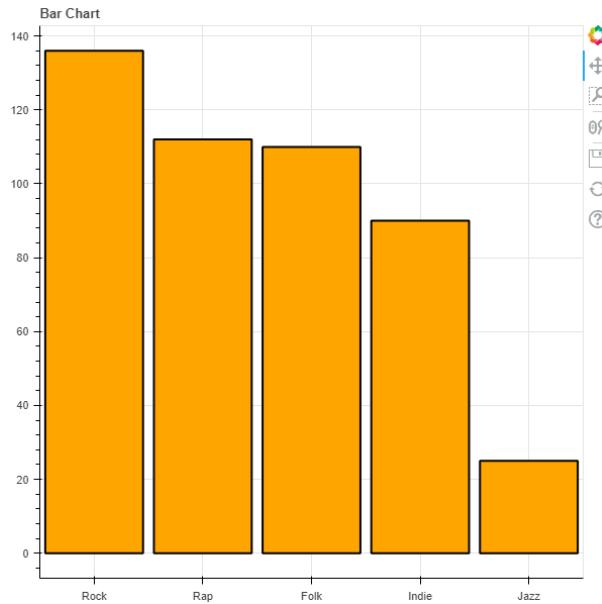
data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
x = list(data.keys())
y = list(data.values())

p = figure(x_range=x, title="Bar Chart")

p.vbar(x=x, top=y, line_color='black', color='orange', width=0.9, line_width=2)

show(p)
```

This program can be downloaded from [GitHub](#). The output is shown in Figure 13



**Figure 13:** Bokeh Bar Chart created with user Spotify data.

### 3.4.8 Saving Figures

Bokeh supports outputs to a static HTML file with a specific name.

```
from bokeh.plotting import output_file  
  
output_file("name.html")
```

After importing the Bokeh plotting interface, it is possible to be able to create different types of plots utilizing the figure created with the figure function.

**3.4.8.1 Saving Figures as PNG** As the purpose of Bokeh is to create interactive `.html` visualizations, it's recommended to keep your visualizations in this format. However, it may sometime be necessary to save as an image file.

In order to save figures as a PNG, both Selenium and a web driver will need to be installed. We will use Chromium here for our web driver. To install both at once, use the commands:

(Windows)

```
$ pip install selenium chromedriver-binary  
$ pip install chromedriver-binary-auto
```

There seems to be issues installing `chromedriver-binary` on Mac computers due to the built-in security, so it is recommended to simply save Bokeh figures as a `.html` file.

When writing a program, Chromium must be added to the PATH through these import statements:

```
from selenium import webdriver  
import chromedriver_binary
```

Bokeh appears to support saving files as a `.svg` but it seems to have bugs and is not recommended. To use the functions, `export_png()` and `export_svg()` must be imported, and can be used as follows:

```
from bokeh.io import export_png, export_svg  
  
export_png(fig, filename="file-name.png")  
export_svg(fig, filename="file-name.svg")
```

Note that Chromium is slow and this process may take delay the execution and performance of the program.

Similarly to Matplotlib, Bokeh can utilize a function to save all created images.

```
from matplotlib import pyplot as plt  
from bokeh.io import export_png, export_svg  
import os
```

```
def save(p):
    name = os.path.basename(__file__).replace(".py", "")
    export_png(p, filename=f"images/{name}.png")
    export_svg(p, filename=f"images/{name}.svg")
    plt.show(p)
```

This code can be accessed on [GitHub](#).

### 3.4.9 Links

- [http://docs.bokeh.org/en/latest/docs/user\\_guide/plotting.html](http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html)
- [http://docs.bokeh.org/en/latest/docs/user\\_guide/plotting.html](http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html)
- <http://docs.bokeh.org/en/latest/>
- <https://docs.bokeh.org/en/latest/docs/reference/plotting/figure.html>
- [https://docs.bokeh.org/en/latest/docs/user\\_guide/export.html](https://docs.bokeh.org/en/latest/docs/user_guide/export.html)

## 3.5 Pandas Graphics

Pandas is a useful library for working with data. It relies on storing data in data frames. Instead of using a set of ordered pairs, a data frame in Pandas is similar to an Excel Spreadsheet or an SQL data frame and supports multiple rows and columns in a tabular fashion.

Visualization for Panda's data frames are an important tool for understanding the data set. Panda's visualization tools are based off of Matplotlib, so many of the plotting functions are the same.

### 3.5.1 Installation

To install Pandas, please use the command:

```
$ pip install pandas
```

### 3.5.2 Import Statements

The user will need to import both Pandas and Matplotlib in order to create and visualize data frames. In addition, Python's `numpy` may be a useful library for mathematical procedures on the data.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

### 3.5.3 Bar Chart

Creating a bar chart with data frames is similar to creating bar charts with Matplotlib, with a couple differences in how you manipulate the data. In the following program, we use the same data and modifications as the [Matplotlib bar chart example](#) that can be found on Github.

```
import matplotlib.pyplot as plt
import pandas as pd

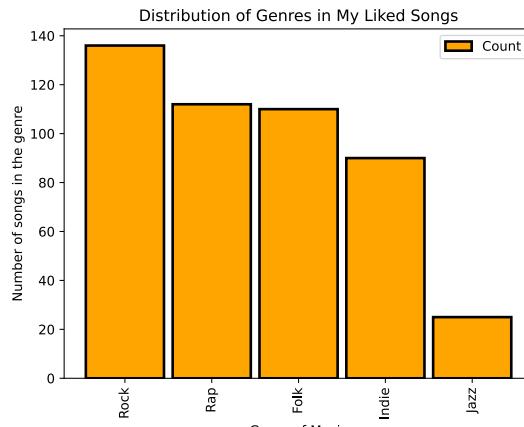
data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
categories = data.keys()
count = data.values()

df = pd.DataFrame({'Count':count, 'Categories':categories})

# Creating the bar chart
df.plot.bar(
    x='Categories',
    y='Count',
    align='center',
    color='orange',
    width=0.9,
    edgecolor="black",
    linewidth=2)

# Editing the bar chart's title, x, and y axes
plt.xlabel("Genre of Music")
plt.ylabel("Number of songs in the genre")
plt.title("Distribution of Genres in My Liked Songs")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure [Figure 14](#).



**Figure 14:** Pandas barchart created from data from Spotify.

Note the differences in creating the chart. Since data frames support multiple dimensions of data, the `x` and `y` we want to graph must be specified in `df.plot.bar()`. However, editing the title and axes are the same as in Matplotlib.

### 3.5.4 Line Chart

A line chart is typically used for time series data and non-categorical data. Pandas supports line chart visualization with `plot.line()`. We use the same data and modifications as the [Matplotlib line chart example](#) that can be found on Github. Note that since this data relies on random number generation the graphs will look slightly different each time.

```
import matplotlib.pyplot as plt
import pandas as pd
import random

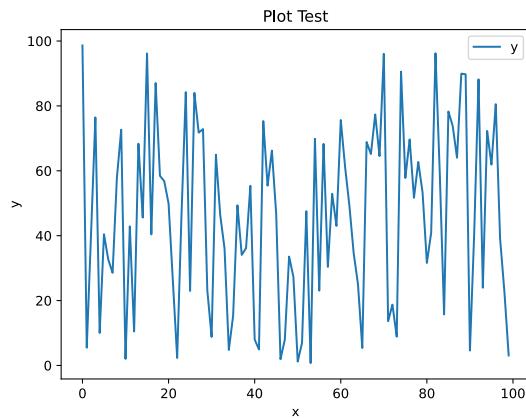
x = []
y = []
for i in range(0, 100):
    x.append(i)
    value = random.random() * 100
    y.append(value)

df = pd.DataFrame({'x':x, 'y':y})

# creating the plot and labeling axes and title
df.plot.line(x='x', y='y')
plt.ylabel("y")
```

```
plt.title("Plot Test")
plt.show()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure Figure 15.



**Figure 15:** Pandas linechart created from random number generation.

### 3.5.5 Pie Chart

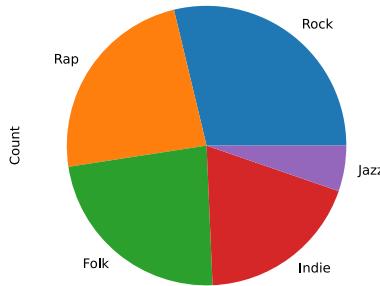
A pie chart is useful for showing a division of a whole. Data that can be represented by a pie chart can also be used to make a bar chart, since both typically use categorical counts. Pandas uses `plot.pie()` to make a pie chart, in a manner similar to `plot.bar()`. We use the same data used to create the line chart for this visualization.

```
import matplotlib.pyplot as plt
import pandas as pd

data = {'Rock': 136, 'Rap': 112, 'Folk': 110, 'Indie': 90, 'Jazz': 25}
categories = data.keys()
count = data.values()

df = pd.DataFrame({'Count':count},index=categories)
plot = df.plot.pie(y='Count',legend=None)
save()
```

This program can be downloaded from [GitHub](#). The output of this program is showcased in Figure Figure 16.



**Figure 16:** Pandas piechart created from data from Spotify.

Note that instead of listing both the Categories and the Count as data, we use the categories as index. This gets the proper labeling for our pie chart. In addition, Pandas automatically adds a legend, but this is unnecessary so we can remove the legend by setting the parameter `legend=None` in `plot.pie()`.

### 3.5.6 Exporting

Note that this is the same as the Matplotlib tutorial found [here](#) on Github.

To export your graph as an image file, you can use the Matplotlib function `savefig("fname.x")`. You can specify the file type by filling in `.x` with `.pdf`, `.png`, `svg`, etc.

The parameter `dpi` sets the DPI (Dots per Inch) of the image being saved. Specify this number in the form of a float. For example, set `dpi=300`.

Additionally, there is another way to save files that may be faster than calling a specific method for each file. The following code showcases this:

```
import matplotlib.pyplot as plt
import os
from matplotlib import pyplot

def save():
    name = os.path.basename(__file__).replace(".py", "")
    plt.savefig(f'/filepath/{name}.png')
    plt.savefig(f'/filepath/{name}.pdf')
    plt.savefig(f'/filepath/{name}.svg')
    plt.show()
```

This code can be accessed on [GitHub](#)

The very last command is `plt.show()`, as this command displays the graph that you made. To show, simply type:

```
plt.show()
```

### 3.5.7 Links

- <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html>
- <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html>
- <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html>

## 4 PYTHON

### 4.1 Cloudmesh StopWatch

---



#### Learning Objectives

- Using timers in python
  - Using `cloudmesh.common.StopWatch`
  - Create benchmarks from `StopWatch`
- 

Improve the documentation of Cloudmesh StopWatch if needed

<https://github.com/cloudmesh/cloudmesh-common/blob/main/cloudmesh/common/StopWatch.py>

Please be reminded that we could develop a program that read the documentation form the docstring. Then we can save it to a file, so we could autogenerate the md file from the docstring.

## 4.2 cms sys command generate

---



### Learning Objectives

- Learn how to create your own cloudmesh commands
- 

locate in the book how to use cms sys command generate. Generate a command with your username. No commit of this is necessary, but we need to make sure you understand how to create a command.

#### 4.2.1 Creating CLI commands using the cloudmesh command

Command line interface commands essentially allow a user to execute commands that have been programmed in `python` from the command line (i.e. on a `macOS` terminal). In the `cloudmesh` project, it is easy to do this, as there is a built-in module that allows a user to develop this command. We assume that the user has properly set up cloudmesh on their own device. The following showcases the `sys command generate`:

- Execute `cms sys command generate` in the home directory of the device. This essentially calls a function within cloudmesh cmd5 that allows
- Once you have done this, it is necessary to `cd` to that directory. For instance, if you typed: `cms sys command generate apples`, then that would have created a command within the directory you were in. Thus, you need to change directory and navigate to that new directory. Then, it is necessary to run `python setup.py` in your command line. Then execute `pip install .` This will configure your device to have the proper requirements for the command that you are generating.
- After this, you can create python scripts within the directory that can be called as an actual command. You will notice a directory called `cloudmesh -> "your command" -> command ->` and then finally, `"yourcommand".py`. In this final `python` file, you can edit and create what it is you were intending on creating.
- When you add new arguments to the command, you have to add those arguments to the `arguments parser` function. This ensures that the parser correctly separates all arguments of the program.

- Finally, in the `if` statements at the end of the program, it is necessary to indicate what arguments/options are specified and what should be done if these commands + arguments + options are called.

### Important Sidebar

The generate command function relies on an API base called `docopt` which is a framework for creating a command line interface. What this does in essence is it creates command line commands based on the framework of a python script.

In this framework there are a few parts to understand:

- \* Usage- indicate how the commands would be called, specifically.
- \* Arguments- indicates what can be passed into the commands to make them work in a specific way.
- \* Options- indicates the options that can be specified. These are optional.

It is necessary to update the arguments in the corresponding if statement.

#### 4.2.1.1 Example of the CLI Creation

```
from cloudmesh.shell.command import command
from cloudmesh.shell.command import PluginCommand
class GregorCommand(PluginCommand):
    @command
    def do_gregor(self, args, arguments):
        """
        ::

        Usage:
        gregor -f FILE
        gregor list
        This command does some useful things.

        Arguments:
        FILE a file name
        Options:
        -f specify the file
        """
        print(arguments)
        if arguments.FILE:
            print("You have used file: ", arguments.FILE)
        return ""
```

This example can be found [here](#) in section **6.0.7.1.4 Create your own Extension**.

Again, all of this relies on having cloudmesh properly downloaded and installed.

#### 4.2.1.2 Example of CLI meshing with a python script

CLI Implementation:

```
def do_cc(self, args, arguments):
```

```
"""
::

Usage:
    cc upload --data=FILENAME
    cc update --data=FILENAME
    cc delete --data=FILENAME
    cc create --queues=QUEUES --database=DATABASE
    cc add --queue=QUEUE --job=JOB --command=COMMAND
    cc run --queue=QUEUE --scheduler=SCHEDULER
    cc remove --queue=QUEUE --job=JOB
    cc remove --queue=QUEUE
    cc list --queue=QUEUE
    cc start
    cc stop
    cc doc
    cc test
    cc temperature

. . .

```

This code can be found [here](#)

Part of the code implementation of the above CLI:

```
class Queues:
    """
    The Queues data structure is a structure that holds all of the queues
    with their corresponding names. It is a meta-queue, essentially. The queues
    class will be a dictionary of dictionaries of jobs, which are
    job names and commands.
    An example command would likely look like:
        cms cc queues list --queues=ab
    """

    def __init__(self, database='yamlDb'):
        """
        Initializes the giant queue structure.
        Default database is yamlDb
        :param name: name of the structure
        :return: creates the queues structure
        """
        if database.lower() == 'yamlDb':
            from cloudmesh.cc.db.yamlDb.database import Database as QueueDB
            self.filename = path_expand("~/cloudmesh/queue/queue")

        elif database.lower() == 'shelve':
```

```
from cloudmesh.cc.db.shelve.database import Database as QueueDB
self.filename = path_expand("~/cloudmesh/queue/queue")

else:
    raise ValueError("This database is not supported for Queues, please fix.
    ↪ ")
```

This code can be found on [GitHub](#)

## 4.3 Linux

---



### Learning Objectives

- Learn how to use the most basic Linux commands
- 

The book has some introduction material to linux, please contribute to the book. Make sure you do not duplicate what others have done or are doing, coordinate. Create a pull request with your contribution.

## 4.4 Python

### 4.4.1 Queue

---



### Learning Objectives

- Learn how to use the built-in Python queue
- 

A queue is a data structure. Essentially, it is a list that has order, meaning that the queue has an object to be removed first and an object to be removed last.

There are three types of queues: FIFO (first in first out), LIFO (last in first out), and Priority Queue (order is based on the data's individual priority score).

```
import queue
```

Then, the data structures can be built as follows:

**4.4.1.1 FIFO Queue** A FIFO queue is the most basic type of queue. Simply put, the values that go into the data structure first are the values that come out of the data structure first.

FIFO queues are easy to implement:

```
# this queue is a FIFO queue
fq = queue.Queue()

for x in range(0, 10):
    fq.put(x)

while not fq.empty():
    print(fq.get())
```

This code can be accessed on [GitHub](#)

**4.4.1.2 LIFO Queue** A LIFO queue is another very basic type of queue. Simply put, it is a queue that operates the last item inserted being the first item that is removed.

LIFO queues are simple to implement:

```
# this queue is a LIFO queue

lq = queue.LifoQueue()

for x in range(0, 10):
    lq.put(x)
```

This code can be accessed on [GitHub](#)

**4.4.1.3 Priority Queue** Finally, priority queues are a more complex queue type. Priority queues work by inserting elements with a sorted priority. For example, if an 8 is inserted before a 2, the 2 will be removed first.

Priority queues are easily implemented:

```
# this queue is a Priority queue
```

```
    pq = queue.PriorityQueue()

    for x in range(0, 10):
        value = int(random.random() * 100)
        print('number being inserted into the priority queue', value)
        pq.put(value)

    print()

    while not pq.empty():
        print('number being removed from the pq', pq.get())
```

This code can be accessed on [GitHub](#)

All specifics on queues can be researched at the below links:

- <https://pymotw.com/3/queue/index.html>
- <https://www.geeksforgeeks.org/priority-queue-in-python/>
- <https://docs.python.org/3/library/queue.html>

## 4.5 glob

`glob` is a small module that searches for files by reading the patterns of filenames. However, `glob` doesn't work in the same way regular expressions do as they follow standard Unix path expansion rules.

### 4.5.1 Glob with asterisk

The example showcases different states under a single directory and counties under subdirectories. The files consist of `.txt` files consisting of different versions of a program under a single directory. Let us first create some file sin a temporary directory:

```
$ cd
$ mkdir tmp/subdir
$ touch tmp/a.txt
$ touch tmp/a-1.txt
$ touch tmp/b.txt
$ touch tmp/b+.txt
$ touch tmp/subdir/c.txt
```

Now let us create the following `file` file in the home directory. To list all the file in the `tmp` directory you can use the asterisk “\*”:

```
import glob

for name in sorted(glob.glob('tmp/*')):
    print(name)
```

It will list the files in the directory tmp in alphabetical order.

```
tmp/a.txt
tmp/a-1.txt
tmp/b.txt
tmp/b+.txt
```

#### 4.5.2 Single Character Wildcard (?)

A question mark (?) can be used to search for files with the same pattern of names through singling out one character as a wildcard. This can be shown in this [example](#).

```
import glob

for name in sorted(glob.glob('tmp/a-?.txt')):
    print(name)
```

This program lists the files starting with a-, a single character and as prefix .txt

```
tmp/a-1.txt
```

#### 4.5.3 Escape Characters

`glob` can also search for files that contain a special characters through using the command `glob.escape(char)`. This can be shown in this [example](#).

```
import glob

specials = '!+('

for char in specials:
    pattern = 'tmp/*' + glob.escape(char) + '.txt'
    for name in sorted(glob.glob(pattern)):
        print(name)
```

The output shown here is a every file that specifically contains the characters !, +, or (. There is only file that does so which is:

```
tm/b+.txt
```

#### 4.5.4 Subdirectories

Not only `glob` can search for files recursively in directries with the `**` query

```
import glob

for name in sorted(glob.glob('tmp/**')):
    print(name)
```

In this example, will produce

```
tmp/subdir
tmp/a.txt
tmp/a-1.txt
tmp/b.txt
tmp/b+.txt
tmp/subdir/c.txt
```

#### 4.5.5 Lins

- <https://pymotw.com/3/glob/index.html>

### 4.6 Mmap

`mmap` standards for memory-map files. Memory-mapping a file involves accessing files directly without the use of traditional I/O functions.

#### 4.6.1 Reading

The example shown here is a short, simple story that is contained in a `.txt` file. The file can be accessed [here](#) as `example.txt`.

```
Let's play with the dog, it's really nice out today!
```

First, the actual file should be opened using the `open` command with the parameter '`r`' for reading and is indicated with the header `f`.

After that, memory-map file can be created using the command `mmap.mmap()` and can be indicated with the header `m`. Within the parentheses `()`, various arguments should be made.

The first argument should be `f.fileno()`, a file descriptor that opens and closes the `mmap` file.

The second argument is the size in bytes, in the form of a float, of the portion of the file to map. If it's `0`, like in this example, the entire file is mapped.

The third argument, which is optional, is the accessibility settings. In this example, it's set to read-only through `access=mmap.ACCESS_READ`.

The code in this example will read various parts of `story.txt` both progressively and through slices.

```
import mmap

with open('example.txt', 'r') as f:
    with mmap.mmap(f.fileno(), 0, access=mmap.ACCESS_READ) as m:
        # Reads the first ten characters
        print('Char. 1-10 (Read) :', m.read(10))

        # Reads a slice of characters
        print('Char. 1-10 (Slice):', m[5:14])

        # Reads the next ten characters
        print('Char. 11-20 (Read) :', m.read(10))
```

The following output is produced:

```
Char. 1-10 (Read) : b"Let's play"
Char. 1-10 (Slice): b' play wit'
Char. 11-20 (Read) : b' with the '
```

#### 4.6.2 Writing

In order to modify files, do the same procedure as reading files by opening the actual file with the `open` command; however, this time, use the parameter `r+` instead of `r`.

Then, create the `mmap` file with `mmap.mmap` with the same required arguments. The optional argument set to the default access mode of `access=mmap.ACCESS_WRITE` in this case.

After those commands, edits can then be made to the `mmap` file as shown in the following [example](#).

```

import mmap
import shutil

# Copy the example file
shutil.copyfile('example.txt', 'example_copy.txt')

word = b'dog'

with open('example_copy.txt', 'r+') as f:
    with mmap.mmap(f.fileno(), 0, access=mmap.ACCESS_WRITE) as m:

        # Memory-map file before change

        print('Memory Before:\n{}'.format(m.readline().rstrip()))
        m.seek(0) # rewind

        loc = m.find(word)
        m[loc:loc + len(word)] = b'cat'
        m.flush()

        # Memory-map file after change

        m.seek(0) # rewind
        print('Memory After:\n{}'.format(m.readline().rstrip()))

        # Actual file after change

        f.seek(0) # rewind
        print('File After:\n{}'.format(f.readline().rstrip()))

```

The following output shows that this access mode allows the modification of the actual file.

```

Memory Before:
b"Let's play with the dog, it's really nice out today!"
Memory After:
b"Let's play with the cat, it's really nice out today!"
File After:
Let's play with the cat, it's really nice out today!

```

This can be changed by setting the access mode to `access=mmap.ACCESS_COPY` as shown in this [example](#).

```

import mmap
import shutil

```

```

# Copy the example file
shutil.copyfile('example.txt', 'example_copy.txt')

word = b'dog'

# Changing access settings

with open('example_copy.txt', 'r+') as f:
    with mmap.mmap(f.fileno(), 0,
                   access=mmap.ACCESS_COPY) as m:

        # Memory-map file before change

        print('Memory Before:\n{}'.format(m.readline().rstrip()))

        # Actual file before change

        print('File Before:\n{}'.format(f.readline().rstrip()))

        m.seek(0) # rewind
        loc = m.find(word)
        m[loc:loc + len(word)] = b'cat'

        # Memory-map file after change

        m.seek(0) # rewind
        print('Memory After:\n{}'.format(m.readline().rstrip()))

        # Actual file after change

        f.seek(0)
        print('File After:\n{}'.format(f.readline().rstrip()))

```

The following output shows that only the `mmap` file and not the actual file was modified in the end.

```

Memory Before:
b"Let's play with the dog, it's really nice out today!"
File Before:
Let's play with the dog, it's really nice out today!
Memory After:
b"Let's play with the cat, it's really nice out today!"
File After:
Let's play with the dog, it's really nice out today!

```

### 4.6.3 Links

- <https://pymotw.com/3/mmap/index.html>

## 4.7 Pickle

`pickle` is a module that turns Python objects into series of bytes that can be transmitted, stored, or reconstructed.

### 4.7.1 Encoding Data

A data structure can be encoded into a string by using the command `pickle.dumps(data)`. In this example, a dictionary is being encoded.

```
import pickle

# Creating dictionary of data
temperatures = {
    'red': 50,
    'blue': 30,
    'yellow': 20,
}
print('Temperatures:', temperatures)

# Pickling the data
pickle_temperatures = pickle.dumps(temperatures)
print('Pickle:', pickle_temperatures)
```

This following output is produced:

```
temperatures: [{'Red': 50, 'Blue': 30, 'Yellow': 20}]
Pickle: b' ... A binary string that we ommitted here ... .'
```

### 4.7.2 Decoding Data

The encoded data can then be decoded using the command `pickle.loads(data)`:

```
import pickle

# Creating dictionary of data
temperatures_0 = {'red': 50, 'blue': 30, 'yellow': 20}
print('Initial temperatures:', temperatures_0)
```

```
# Encoding the data
pickle_temperatures_0 = pickle.dumps(temperatures_0)

# Decoding the data
temperatures_1 = pickle.loads(pickle_temperatures_0)
print('From pickle database:', temperatures_1)

# Checking authenticity
print("Same:", temperatures1 is temperatures2)
print("Equal:", temperatures1 == temperatures2)
```

This can be shown in the following output:

```
Initial temperatures: [{'Red': 5, 'Blue': 3, 'Yellow': 2}]
From pickle database: [{'Red': 5, 'Blue': 3, 'Yellow': 2}]
```

This command will produce data that is equal to the original data, but it's not the same as shown by the following output:

```
Same: False
Equal: True
```

### 4.7.3 Links

- <https://pymotw.com/3/pickle/index.html>

## 4.8 Shelve

`shelve` is a library that gives users the ability to create, store, modify, and control the accessibility of data without a relational database. A shelf has a string key and data that can be near anything as long as it is supported by `pickle`. This includes integers, dictionaries, and Objects.

### 4.8.1 Creating a New Shelf

A new shelf can be created using the command: `d = shelve.open(filename)`. Shelve can store objects; you can insert a dictionary, for example, as shown next:

```
import shelve

computers = shelve.open('computers.db')
```

```
computers['temperature'] = {
    'red': 80,
    'blue': 40,
    'yellow': 50,
}
```

*On a Windows:* This creates three files `computers.db.bak`, `computers.db.dat`, and `computers.db.dir`. These `.bak`, `.dat`, and `.dir` files hold the shelf information that can be accessed for future use.

For other computers, this creates the file ‘computers.db’.

#### 4.8.2 Accessing a Shelf

After it’s been created, it can be accessed while reading objects into variables.

```
import shelve

with shelve.open('computers.db') as computers:
    t = computers['temperature']

print(t)
```

This produces the following output:

```
{'red': 80, 'blue': 40, 'yellow': 50}
```

#### 4.8.3 Making Shelf Read-Only

The user can also make their data read-only by adding the `flag` parameter as shown:

```
shelve.open('computers.db', flag='r')
```

That way, when a user tries to modify it, it produces an error:

```
import dbm
import shelve

computers = shelve.open('computers.db', flag='r')
print('Temperature:', computers['temperature'])
try:
    computers['temperature']['green'] = 100
except dbm.error as err:
    print('ERROR:', err)
```

The following output is produced:

```
Temperature: {'red': 80, 'blue': 40, 'yellow': 50}
ERROR: The database is opened for reading only
```

#### 4.8.4 Modifying Shelves

In order to modify a shelf, simply open up the shelf again as shown:

```
shelve.open('shelf_name')
```

Make sure to use this before making the modification, or else it won't work. For simple assignments, this will suffice.

However, in most cases, you use the `writeback=True` parameter. This caches the modifications and slows down the saving process. As seen, however, you can now directly access and modify the shelf entries as it was a two-dimensional dictionary.

You can also delete shelf items using their keys with the `del` method.

```
import shelve
from pprint import pprint

computers = shelve.open("computers.db", writeback=True)
print('Initial temperature:')
pprint(computers['temperature'])

computers['temperature']['green'] = 101
del computers['temperature']['yellow']
print()
print('Modified temperature:')
pprint(computers['temperature'])
```

Modifications are preserved as shown in this output:

```
Initial temperature:
{'red': 80, 'blue': 40, 'yellow': 50}

Modified temperature:
{'blue': 40, 'green': 101, 'red': 80}
```

#### 4.8.5 Closing Shelves

Lastly, in order to save the shelf so that it may be accessed next time, use the command ‘close()’

```
import shelve

computers = shelve.open('computers.db')
# do your shelf operations here
computers.close()
```

---

#### 4.8.6 Links

- <https://pymotw.com/3/shelve/index.html>
- <https://docs.python.org/3/library/shelve.html>

### 4.9 FastAPI



#### Learning Objectives

- Learn how to use fastAPI
- 

FastAPI is a Python framework that allows developers set up a REST service and define its functionality with an easy-to-use API.

#### 4.9.1 FastAPI Install

As FastAPI will need a web server, we will use `uvicorn` for development purposes. In a production environment other, more mature Web services are recommended. To install FastAPI and uvicorn simply use the command:

```
$ pip install "fastapi[all]"
```

TODO: THERE ARE TOO MANY DIFFERENT EXAMPLES, PLEASE CREATE ONE THAT BUILDS ON TOP OF EACH OTHER, USE COMPUTERS WITH TEMPERATURES

#### 4.9.2 FastAPI Quickstart

One of the simplest FastAPI file looks like this, which we assume is placed in a file called `main.py`:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"processor": "5950X"}
```

Start the live FastAPI app in the `uvicorn` server use the command:

```
$ uvicorn main:app --reload
```

This will yield the output

```
1 INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
2 INFO: Started reloader process [28720]
3 INFO: Started server process [28722]
4 INFO: Waiting for application startup.
4 INFO: Application startup complete.
```

The first line includes information about which URL is used to contact it to obtain a response. An easy way to view it is to enter <http://127.0.0.1:8000> in your browser. The JSON response will appear as:

```
{"message": "Hello World"}
```

One of the embedded features of FastAPI is its build in documentation framework based on OpenAPI schema but also alternative formats such as redoc. You can look at it while going with your browser to the URL

- OpenAPI: <http://127.0.0.1:8000/docs>.
- OpenAPI json: <http://127.0.0.1:8000/openapi.json>
- Redoc: <http://127.0.0.1:8000/redoc>

**4.9.2.1 Path** One of the mechanisms FastAPI provides it to easily specify the URL that is needed to trigger the functionality of the defined function after its definition.

We have seen such an example in `@app.get("/")` which activates the `root` function when the URL of the server is specified followed by `/`

You can add other path's and functions. Let us assume you add to our initial program the function

```
@app.get("/temperature")
async def temperature():
    return {"temperature": 0}
```

The if you use the URL <http://127.0.0.1:8000/temperature>, we will see

```
{"temperature": 0}
```

#### 4.9.3 Query Parameters

When you declare other function parameters that are not part of the path parameters, they are automatically interpreted as URL “query” parameters.

```
from fastapi import FastAPI

app = FastAPI()
jobs = [{"name": "Foo"}, {"name": "Bar"}, {"name": "Baz"}]

@app.get("/jobs/")
async def get_job(skip: int = 0, limit: int = 10):
    return jobs[skip : skip + limit]
```

The query is the set of key-value pairs that go after the `?` in a URL, separated by `&` characters. For example, in the URL:

```
http://127.0.0.1:8000/jobs/?skip=0&limit=10
```

In this case the query parameters are:

- `skip`: with a value of 0
- `limit`: with a value of 10

##### 4.9.3.1 Searching in the fastapi

```
from fastapi import FastAPI

app = FastAPI()
jobs = [{"name": "Foo"}, {"name": "Bar"}, {"name": "Baz"}]
```

```
@app.get("/job/")
async def search_job(name:str):
    result = None
    for item in jobs:
        if item['name'] == name:
            result = name
    return result
```

For example, in the URL

```
http://127.0.0.1:8000/job/?name=Foo')
```

Output

```
"Foo"
```

#### 4.9.3.2 Running Through Git bash

```
import requests

result = requests.get('http://127.0.0.1:8000/job/?name=Foo')

print(result.text)

print(result.status_code)
print(result.headers['content-type'])
print(result.encoding)
print(result.text)
print(result.json())
```

Run python code on Git bash

```
$ python r.py
```

where r.py is the file name, yeilding in the output

```
"Foo"
200
application/json
utf-8
"Foo"
Foo
```

#### 4.9.4 Running the cc FastAPI service

The cloudmesh cc FastAPI service can be started with

```
$ cms cc start
```

To see the documentation you do not have to type in the URL in the browser, but instead you can use the command

```
$ cms cc doc
```

which will open the url `http://127.0.0.1:8000/docs` in the browser.

To stop the server, use the command

```
$ cms cc stop
```

#### 4.9.5 Testing

In order to test the FastAPI code, we want to create a new test file. In this file, we import `TestClient` from `fastapi.testclient`. `TestClient` creates a test object that follows pytest conventions. In addition, you must import the `app` FastAPI object from the main module.

You can then create the test object by passing this FastAPI object into the `TestClient`. You use assert statements to test for validity.

```
from fastapi.testclient import TestClient
from .main import app

client = TestClient(app)

def test_temperature():
    response = client.get("/temperature")
    assert response.status_code == 200
    assert response.json() == {"temperature": 0}
```

You can create several test functions and run them with `pytest`. Lastly, there are other parameters for `client.get()` that may be further explored in the [requests documentation](#).

**4.9.5.1 Asynchronous Testing** To test asynchronous functions, for example `async` queries, we can no longer use `TestClient` due to Pytest's inherent sync nature. Instead, we use a very similar client called `HTTPX` that can make both synchronous and asynchronous requests.

In addition, we must mark these async test functions with the `pytest.mark.anyio` flag.

```
import pytest
from httpx import AsyncClient

from .main import app

@pytest.mark.anyio
async def test_temperature():
    async with AsyncClient(app=app, base_url="http://test") as ac:
        response = await ac.get("/temperature")
    assert response.status_code == 200
    assert response.json() == {"temperature": 0}
```

The `await` functions sends the asynchronous request.

This can once again be run with pytest, and additional async or sync test functions can be added as well.

#### 4.9.6 Links

- <https://fastapi.tiangolo.com/tutorial/first-steps/>
- <https://fastapi.tiangolo.com/tutorial/testing/#using-testclient>
- <https://fastapi.tiangolo.com/advanced/async-tests/>
- cloudmesh cc TODO

## 5 RIVANNA

### 5.1 Rivanna

---



#### Learning Objectives

- Learn how to use Rivanna
  - Learn how to set up VPN to access Rivanna from commandline
  - Learn how to access Rivanna from a terminal including Windows
-

Presentation:

- Knuuti
- UVA Rivanna presentation, June 8th, 2022

Logging in to Rivanna via web interface

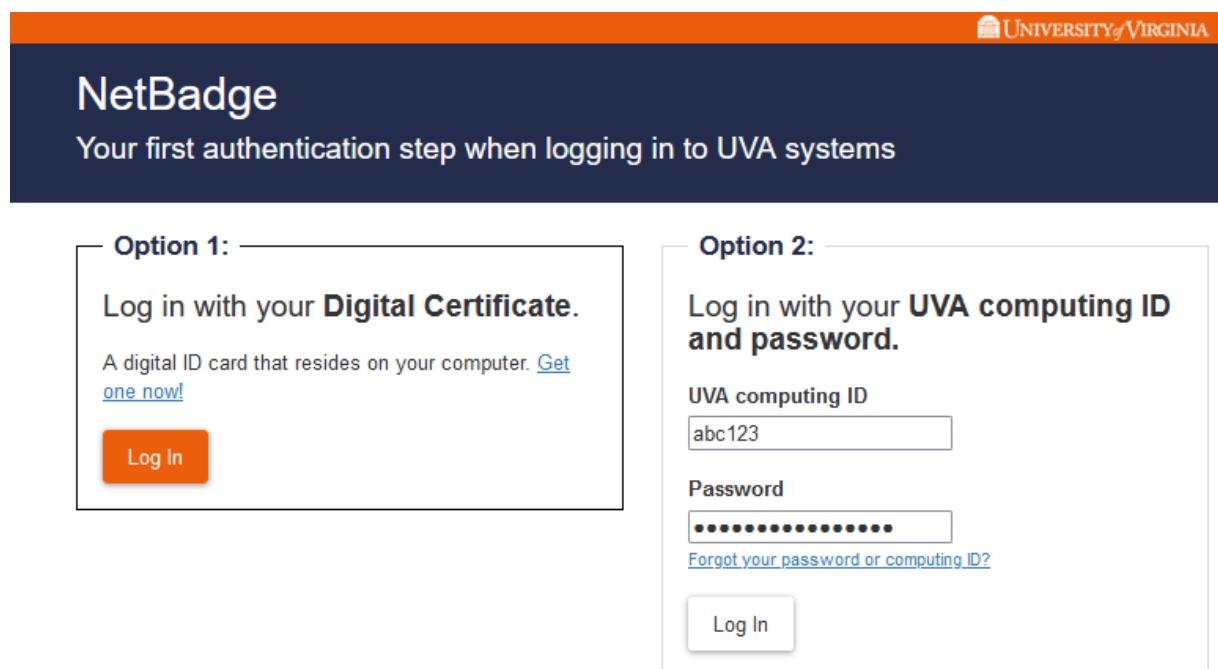
Documentation: <https://www.rc.virginia.edu/userinfo/rivanna/login/#web-based-access>

Login: <https://rivanna-portal.hpc.virginia.edu/>

UVA VPN: <https://in.virginia.edu/vpn>

Shell access: <https://rivanna-portal.hpc.virginia.edu/pun/sys/shell/ssh/rivanna.hpc.virginia.edu>

JupyterLab: [https://rivanna-portal.hpc.virginia.edu/pun/sys/dashboard/batch\\_connect/sys/jupyter\\_lab/session\\_contexts/new](https://rivanna-portal.hpc.virginia.edu/pun/sys/dashboard/batch_connect/sys/jupyter_lab/session_contexts/new)



The screenshot shows the UVA login page with the title "NetBadge" and subtitle "Your first authentication step when logging in to UVA systems". There are two main options for logging in:

- Option 1:** Log in with your **Digital Certificate**. A digital ID card that resides on your computer. [Get one now!](#)
- Option 2:** Log in with your **UVA computing ID and password**.
  - UVA computing ID:** abc123
  - Password:**  [Forgot your password or computing ID?](#)
  -

**Figure 17:** UVA Login

**Figure:** UVA Login

The user must install Duo Mobile on smartphone to use as an authentication service to approve logins.

For security reasons we suggest never saving the password within the browser autofill.

After logging in, you will receive an email through your UVA email inbox to create an account on Rivanna. Once completing the sign-up process, it will take around 1 hour for your account creation to be finalized.

If connecting through SSH, then a VPN is required. Follow the instructions to download UVA Anywhere at the following link: <https://in.virginia.edu/vpn>

To log in to Rivanna, ensure you are connected to UVA Anywhere and issue the following (make sure you replace `abc123` with your UVA id):

```
you@yourcomputer$ ssh-copy-id abc123@rivanna.hpc.virginia.edu
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
    ↳ any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
    ↳ now it is to install the new keys
abc123@rivanna.hpc.virginia.edu's password:
```

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'abc123@rivanna.hpc.virginia.edu'" and check to make sure that only the key(s) you wanted were added.

```
you@yourcomputer$ ssh abc123@rivanna.hpc.virginia.edu
Last login: Tue May 31 11:55:43 2022
Authorized Use Only!
-bash-4.2$
```

### 5.1.1 Notes: superpod

Estimated deployment for testing by the end of this summer.

Hardware Components:

- 10 DGX-A100 (80GB) Servers (8 GPUs)
- 2 DGX-A100 (40GB) Servers (16 GPUs)
- HDR Infiniband (200GB/s) IB network fabric for GPU-to-GPU direct communication
- 500T ESS3200 pure SSD SpectrumScale (aka GPFS) direct-to-GPU storage array

The SuperPod is a collection of GPU servers (Nvidia DGX-A100) integrated into the Rivanna Cluster (on the GPU partition) with an 200Gb/s IB fabric interconnecting the GPUs with each other and with dedicated temporary storage for [Nvidia GPUDirect](#) features. The GPU Direct features allow for very fast transfers between the GPUs, storage and also for larger distributed GPU models.

### 5.1.2 Special DGX Nodes on Rivanna

DGX A100 (udc-an36-1) is now available for your bii\_dsc and bii\_dsc\_community members to test.

Here is the current status:

- The server is NOT YET integrated into the NVIDIA SuperPod because we are still awaiting networking equipment for implementing the SuperPod. We will be in touch if there is a need for a maintenance outage to integrate the server into the SuperPod.
- There is a RAID0 array of NVMe disks mounted locally at /localscratch. The capacity is 27TB. Please keep in mind that /localscratch is not backed up.
- The server is named udc-an36-1 and is currently in the bii-gpu partition with a permanent reservation named bi\_fox\_dgx for only bii\_dsc and bii\_dsc\_community allocation members to use. To use this reservation for the A100 node, your researchers and students will have to use the following slurm flags:

```
#SBATCH --reservation=bi_fox_dgx
#SBATCH --account=<enter relevant allocation here>
#SBATCH --partition=bii-gpu
#SBATCH --gres=gpu:<number of GPUs to request>
```

For -account, users will enter either bii\_dsc or bii\_dsc\_community depending on which group they belong to. You can find this by running the allocations utility at the commandline. For -gres=gpu:, users should enter the number of GPUs requested.

The full details of the reservation are below. I named the Slurm reservation “bi\_fox\_dgx”. It’s not a typo. To change the name of the reservation, I would have to delete the reservation and re-create it and the actual name of the reservation does not affect the reservation’s usability. I’ve successfully tested the ability to use this reservation for all the current bii\_dsc and bii\_dsc\_community members using the Slurm parameters I sent previously.

```
ReservationName=bi_fox_dgx StartTime=2022-06-01T08:37:38 EndTime=2022-06-02T08:37:38
    ↪ Duration=1-00:00:00
Nodes=udc-an36-1 NodeCnt=1 CoreCnt=256 Features=(null) PartitionName=bii-gpu
    ↪ Flags=DAILY,SPEC_NODES
TRES=cpu=256
Users=(null) Groups=(null) Accounts=bii_dsc,bii_dsc_community Licenses=(null)
    ↪ State=ACTIVE BurstBuffer=(null) Watts=n/a
MaxStartDelay=(null)
```

#### 5.1.2.1 Starting interactive job on special partition

```
ssh $USERNAME@rivanna.hpc.virginia.edu
```

```
$ ijob --reservation=bi_fox_dgx --account bii_dsc --partition=bi-i-gpu --gres=gpu:1
salloc: Pending job allocation 39263336
salloc: job 39263336 queued and waiting for resources
salloc: job 39263336 has been allocated resources
salloc: Granted job allocation 39263336
salloc: Waiting for resource configuration
salloc: Nodes udc-an36-1 are ready for job

$ nvidia-smi
```

which will result in

```
+-----+
| NVIDIA-SMI 470.103.01    Driver Version: 470.103.01    CUDA Version: 11.4 |
+-----+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC | | | | |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
| |          |          |           |           |          | MIG M. |
+=====+=====+=====+=====+=====+=====+=====+=====+
| 0  NVIDIA A100-SXM... Off | 00000000:07:00.0 Off |          0 | | | | |
| N/A   29C     P0    54W / 400W |    85MiB / 81251MiB |    0%     Default |
| |          |          |           |           |          | Disabled |
+-----+
+-----+
| Processes:
| GPU  GI  CI      PID  Type  Process name          GPU Memory |
|       ID  ID
| ======+=====+=====+=====+=====+=====+=====+=====
| 0    N/A  N/A    13486    G  /usr/bin/X          63MiB |
| 0    N/A  N/A    13639    G  /usr/bin/gnome-shell  21MiB |
+-----+
```

### 5.1.3 SSH Config

```
$ cat ~/.ssh/config
host rivanna
    User <USERNAME>
    HostName rivanna.hpc.virginia.edu
    IdentityFile ~/.ssh/id_rsa
```

## 5.2 Run Python MPI programs on Rivanna

see the book Python MPI

add chapter if not there

# 6 NLP

## 6.1 Documentation to get started with AWS Translate Service

AWS offers there a Text-translation service. A comorehensive manual on how to use it and sign up for this service is located at

- <https://docs.aws.amazon.com/translate/latest/dg/setting-up.html>

In this documentation we provide a summary of steps that give you quickly access to the service.

### 6.1.1 Step 1: Creating a new iam user account on aws.

To use AWS-translate you need to create an account via the IAM user account application form:

- <https://portal.aws.amazon.com/billing/signup#/start/email>

Like many other cloud services you will have to enable billing in order to use this service. Sign up for an iam user account and make the username name `adminuser` :



## Sign up for AWS

**Explore Free Tier products with a new AWS account.**

To learn more, visit [aws.amazon.com/free](https://aws.amazon.com/free).



**Root user email address**  
Used for account recovery and some administrative functions

**AWS account name**  
Choose a name for your account. You can change this name in your account settings after you sign up.

**Verify email address**

OR

**Sign in to an existing AWS account**

**6.1.1.1 Step 2: Creating a access key ID and secret ID** Once you have signed up with an IAM user account and have implemented billing you can navigate to the aws console:

- <https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/home>

The screenshot shows the AWS Identity and Access Management (IAM) dashboard. On the left, a sidebar menu includes options like 'Dashboard', 'Access management' (User groups, Users, Roles, Policies, Identity providers, Account settings), 'Access reports' (Access analyzer, Archive rules, Analyzers, Settings), and 'Service control policies (SCPs)'. The main content area displays 'Security recommendations' with a red notification badge (1). It shows a warning about adding MFA for the root user and a note that the root user has no active access keys. Below this is a summary of 'IAM resources': User groups (2), Users (1), Roles (2), Policies (1), and Identity providers (0). A 'What's new' section lists recent changes, such as 'Right-size permissions for more roles in your account using IAM Access Analyzer to generate 50 fine-grained IAM policies per day.' To the right, the 'AWS Account' section provides details like Account ID (213055063682), Account Alias (213055063682), and a sign-in URL (https://213055063682.siginin.aws.amazon.com/console). A 'Quick Links' section includes links to 'My security credentials', 'Policy simulator', and 'Web identity federation playground'.

Here you can access secret keys and set permissions. At the top of this page search for text translate.

From here we are going to create an access key ID and a secret key id. This step is trivial to the success of a translation example

From the IAM dashboard page select the 'Users' option under IAM Resources

After clicking users you can create a user to run credentials.

From here click add user.

The screenshot shows the AWS IAM Users list interface. At the top, a blue banner introduces the new Users list experience, stating: "We've redesigned the Users list experience to make it easier to use. [Let us know what you think.](#)" Below the banner, the navigation path is IAM > Users. The main title is "Users (1) Info". A sub-instruction says: "An IAM user is an identity with long-term credentials that is used to interact with AWS in an account." There is a search bar with placeholder text "Find users by username or access key" and a pagination indicator showing page 1 of 1. The table header includes columns for "User name", "Groups", "Last activity", "MFA", and "Password a...". The single user listed is "Administrator", which is part of the "Administrators" group, last active "Yesterday", has no MFA, and was created "11 days ago".

You will be required to add a name.

Make sure you check the box for Access Key and go to the next step.

AWS user

1 2 3 4 5

### Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*  This field is required.

[+ Add another user](#)

### Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

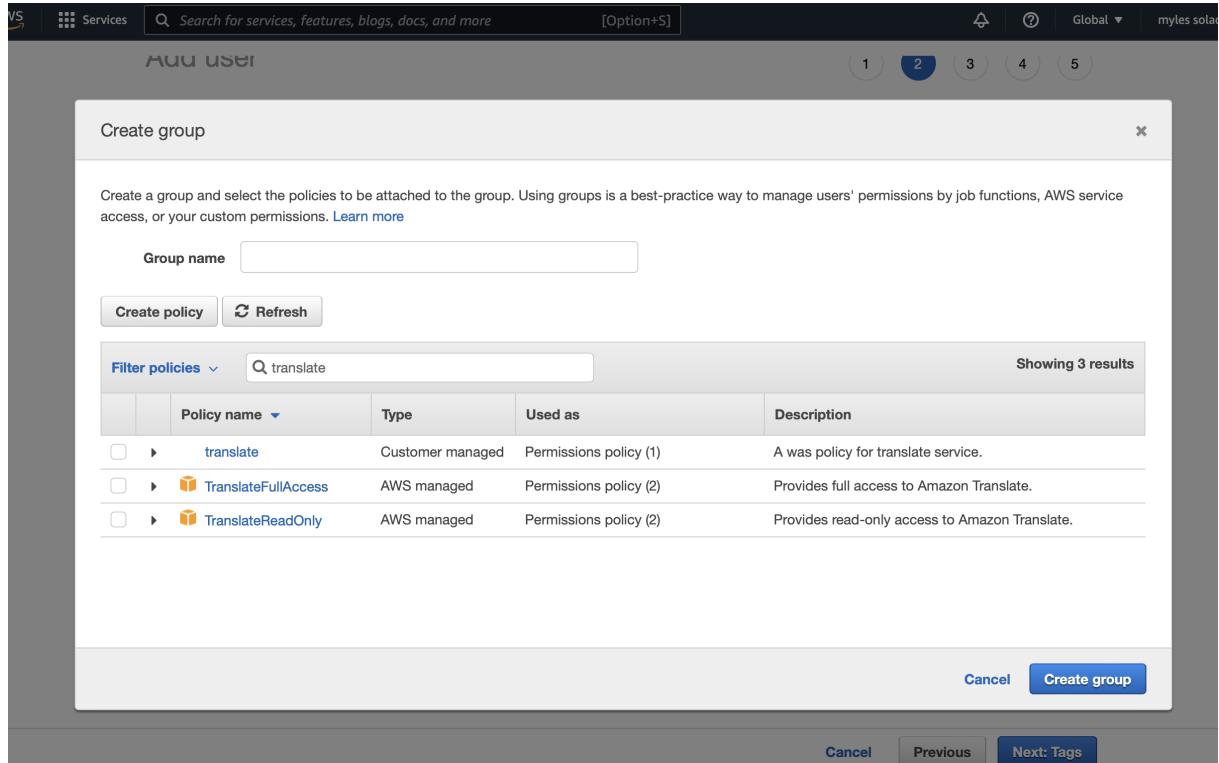
Select AWS credential type\*  Access key - Programmatic access  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

Password - AWS Management Console access  
Enables a **password** that allows users to sign-in to the AWS Management Console.

\* Required [Cancel](#) [Next: Permissions](#)

Here is a section to add permissions for your IAM account User Group. From here search translate and check the boxes for `Translate`, and `TranslateFullAccess`.

You can then click create group.



Next step is tags. this step is meant for users that want to give optional tags to their project.

You can skip this as it is irrelevant to this project.

After reaching this page you will want to confirm the creation of your user group.

## Add user

1 2 3 4 5

## Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

## User details

User name	pop
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

## Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	Administrators

## Tags

No tags were added.

Cancel Previous Create user

From here you will be sent to a screen where you can download credentials

## Add user

1 2 3 4 5

✓ Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://213055063682.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key
▶	pop	AKIATDGYSI2BF6MKXHOR	***** Show

download these credentials as a csv file for later use.

Open up a terminal window to install aws on the command line.

Run these commands:

Start of a virtual environment

```
$ python3.10 -m venv ~/ENV3
$ source ~/ENV3/bin/activate
$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
$ sudo installer -pkg ./AWSCLIV2.pkg -target /
$ which aws
$ aws --version
$ aws configure
```

Here they will give you an output like

AWS Access Key ID [\*\*\*\*\*4FCA]:

You will open the file you downloaded from the user creation earlier. You will see your personalized access key in that file. You will copy that key and paste it into the terminal.

You then will add your secret key which is also in the csv you downloaded.

AWS Secret Access Key [\*\*\*\*\*/kIg]:

You then will add the region

Default region name [eu-west-1]:

The region depends on which one is closest to you. For me, it is eu-west-1 .

Then you will insert json

Default output format [json]:

Here It is recommended to insert json :

From here you can get started working in the command line found here: [CLI Start](#)

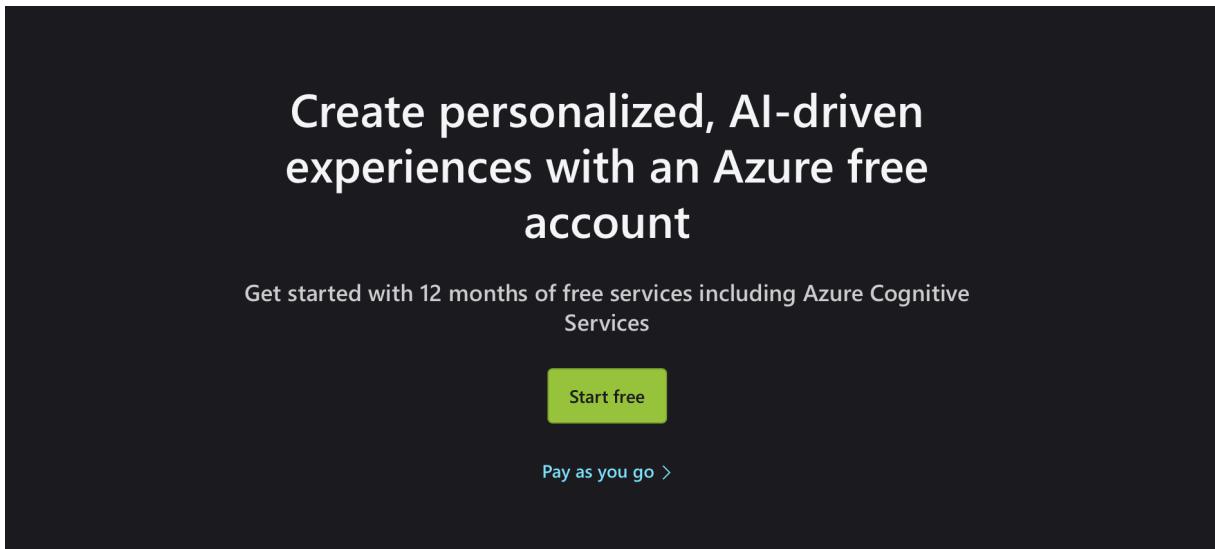
## 6.2 Documentation to get started with Azure Translate.

### 6.2.1 Step 1: account creation

Here are steps to get started with the azure translate example in cloudmesh nlp.

navigate to his link to sign up with an azure account:<https://azure.microsoft.com/en-us/free/cognitive-services/>

After selecting this link you will follow the instructions to set up a microsoft azure account. This requires billing.



furthermore, after setting up an account with an email and billing you will be prompted to the console screen.

from here, you will click the top left link to go to the console.

You can either type translators in the search bar or follow this link: [https://portal.azure.com/#blade/Microsoft\\_Azure\\_Proc](https://portal.azure.com/#blade/Microsoft_Azure_Proc)

The screenshot shows the Microsoft Azure Cognitive Services Translator blade. At the top, there's a search bar and various navigation links like 'Overview' and 'All Cognitive Services'. On the left, there are category filters: 'Speech service', 'Language' (with 'Translator' selected), 'Vision', 'Decision', and 'Multipurpose'. The main area displays a table header with columns: Name ↑↓, Kind ↑↓, Location ↑↓, Custom Domain ... ↑↓, Pricing tier ↑↓, and Status ↑↓. Below the table, a large gray box contains a circular icon with Japanese characters 'あ' and 'る' and the text 'No translators to display. Try changing or clearing your filters.' A blue 'Create translator' button and a 'Learn more' link are also present.

From here you will click the create translator button.

This is what the Create Translator console will look like. Here there is a form for creating an endpoint for this translate service

Microsoft Azure   [Upgrade](#)    Search resources, services, and docs (G+/)

Home > Create a resource > Translator >

## Create Translator

multi-language support such as website localization, e-commerce, customer support, messaging applications, internal communication, and more.

**Project Details**

Subscription \*

Resource group \*   
[Create new](#)

**Instance Details**

**Info** Please choose the Global region unless your business or application requires a specific region. Applications that do not offer a region selection use the Global region.

Region \*

Name \*

Pricing tier \*

[View full pricing details](#)

[Review + create](#)   [< Previous](#)   [Next : Network >](#)

This what the form should look like filled out with the proper applied information.

The screenshot shows the Microsoft Azure 'Create a resource' wizard for creating a 'Translator' service. The top navigation bar includes 'Microsoft Azure', 'Upgrade', and a search bar. The breadcrumb path is 'Home > Create a resource > Translator >'. The main title is 'Create Translator'.

**Project Details:**

- Subscription \*: Azure subscription 1
- Resource group \*: cloudmesh (with a 'Create new' link)

**Instance Details:**

- Region \***: East US
- Name \***: cloudmesh

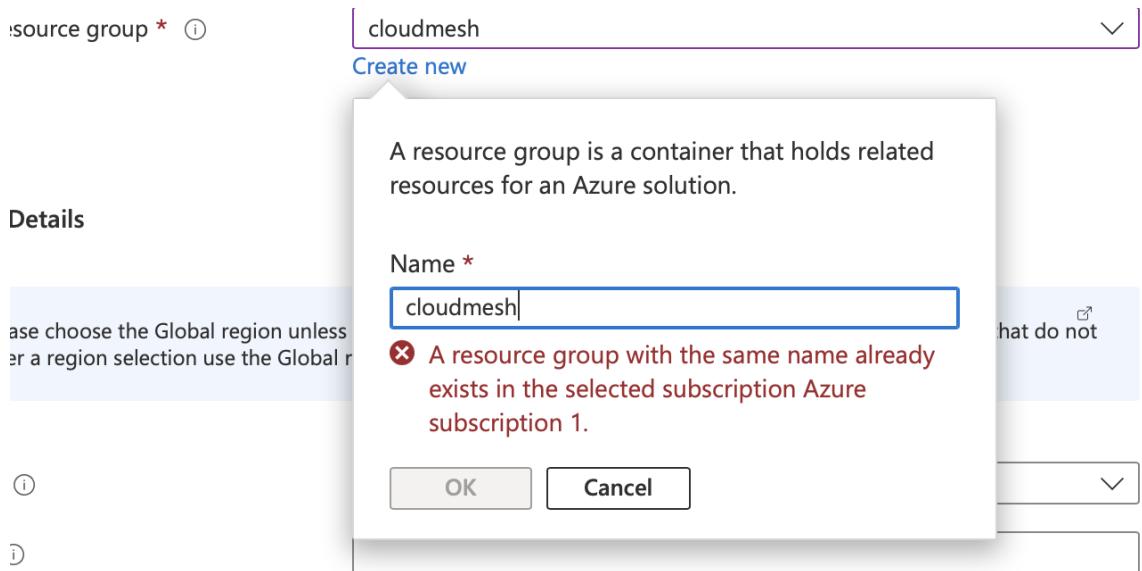
**Pricing tier \***: Standard S1 (Pay as you go)

**Buttons at the bottom:**

- Review + create
- < Previous
- Next : Network >

Resource groups are mostly irrelevant they are for bigger scaled projects where multiple people are working. Title this something for the project since it is required for an endpoint.

After this scroll down and hit 'Create'



This is what the screen looks like after deployment. Download the deployment details and click next steps.

A screenshot of the Microsoft Azure portal showing the "Microsoft.CognitiveServicesTextTranslation-20220505164908 | Overview" page. The page header includes the Microsoft Azure logo, a "Home" link, and a search bar. The main content area shows a green checkmark icon and the text "Your deployment is complete". Below this, deployment details are listed: Deployment name: Microsoft.CognitiveServicesTextTranslation-2... Start time: 5/5/2022, 4:52:07 PM; Subscription: Azure subscription 1; Correlation ID: d8412de6-fc83-45f1-a882-df87; Resource group: cloudmesh; Microsoft.CognitiveServicesTextTranslation-20220505164908. There are also links for "Deployment details (Download)" and "Next steps".

### 6.2.2 Installing and Starting Azure Translate through the command line

**6.2.2.1 Step 2: installing using Homebrew** Homebrew is by far the easiest way of installing Microsoft Azure.

On the command line use the command

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD  
↳ /install.sh)"
```

This will install Homebrew and all of its packages. Now run the command below to install azure cli.

```
$ brew update && brew install azure-cli
```

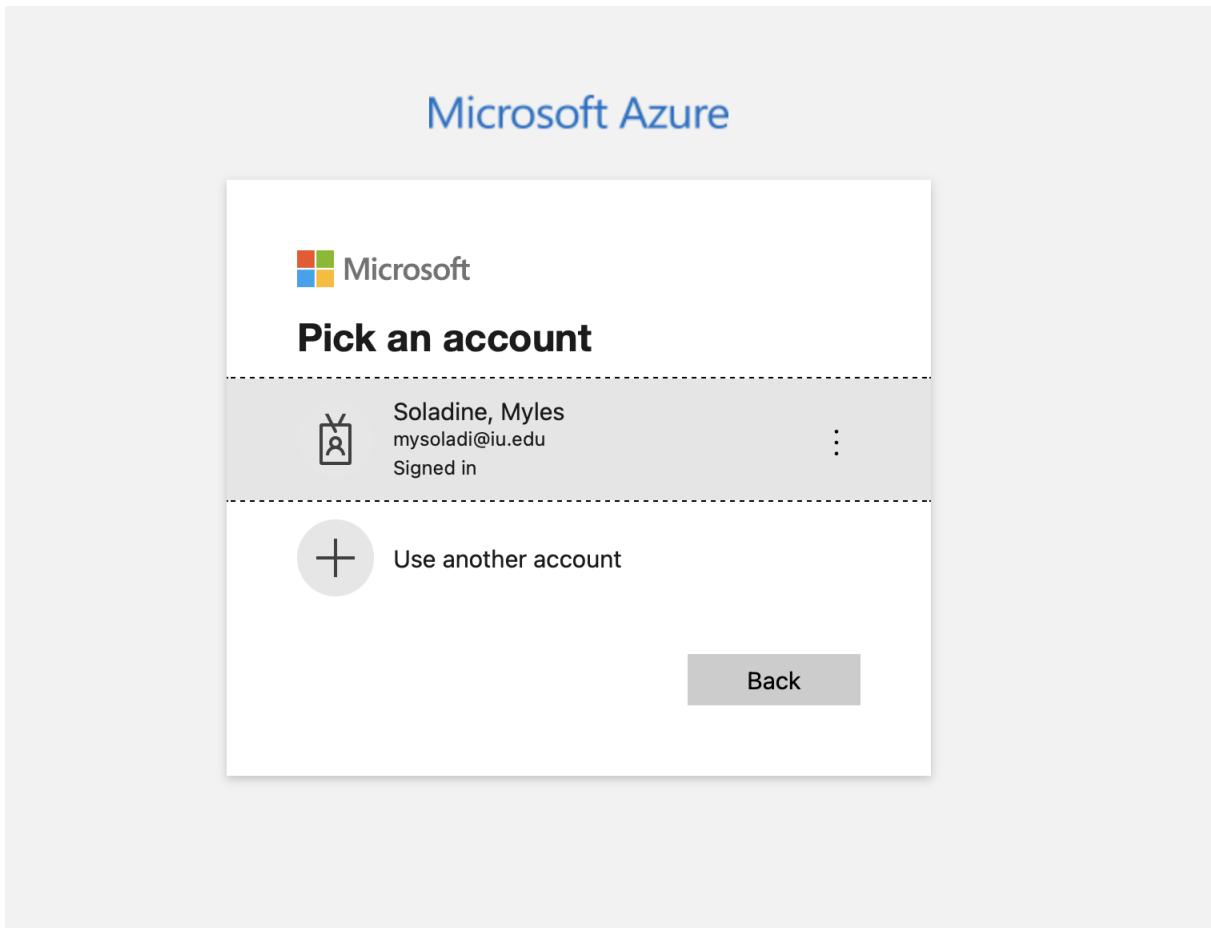
This installs the azure command line interface in the current directory.

From here we need to login to azure and its console that we created above.

start with

```
$ az login
```

This command will prompt you with a new window page with a microsoft login.



From here choose your proper login that was created with azure translate.

you will see a success in login.

information about your account will be displayed on the command line.

```
[  
 {  
   "cloudName": "AzureCloud",  
   "homeTenantId": "1113be34-aed1-4d00-ab4b-cdd02510be91",  
   "id": "d0ff5454-d152-4d11-8fe8-58a0c08581f1",  
   "isDefault": true,  
   "managedByTenants": [],  
   "name": "Azure subscription 1",  
   "state": "Enabled",  
   "tenantId": "1113be34-aed1-4d00-ab4b-cdd02510be91",  
   "user": {  
     "name": "mysoladi@iu.edu",  
     "type": "user"  
   }  
 }
```

```
}
```

Use the command line to create a translate example using azure:

```
curl -X POST "https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&to=es" \
-H "Ocp-Apim-Subscription-Key:<secret key>" \
-H "Ocp-Apim-Subscription-Region:<region>" \
-H "Content-Type: application/json" \
-d "[{"Text": "Hello, what is your father?"}]"
```

For `secret key` you must insert the endpoint key that was generated in the previous account creation (same as key 1 in the figure below).

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', a search bar, and user information ('mysoladi@iu.edu INDIANA UNIVERSITY (INDIANA...)'). Below the navigation is a breadcrumb trail: 'Home > cloudmesh'. The main content area has a title 'cloudmesh | Keys and Endpoint' with a 'Translator' subtitle. On the left, a sidebar lists 'Resource Management' sections: 'Keys and Endpoint' (selected), 'Encryption', 'Commitment tier pricing', 'Pricing tier', 'Networking', 'Identity', 'Cost analysis', 'Properties', 'Locks', 'Monitoring' (with 'Alerts', 'Metrics', and 'Diagnostic settings' listed). The 'Keys and Endpoint' section contains a note: 'These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely—for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.' It shows 'KEY 1' and 'KEY 2' fields, both containing redacted text. Below this is a 'Location/Region' field set to 'eastus'. Under 'Web API', there are two tabs: 'Containers' (selected) and 'Web API'. The 'Web API' tab has a note: 'Use the below endpoints while using the Web API. To force the request to be handled by a specific geography, see [here](#)'. It lists 'Text Translation' with endpoint '<https://api.cognitive.microsofttranslator.com/>' and 'Document Translation' with endpoint '<https://cloudmesh.cognitiveservices.azure.com/>'.

Region is also highlighted in this form.

This will return

```
[{"detectedLanguage":{"language":"en","score":1.0},"translations":[{"text":"Hallo Welt","to":"de"}}}]
```

Using azure in a program sample found [here](#) you can use the endpoints, region, and secret key found in account creation to return a text translation.

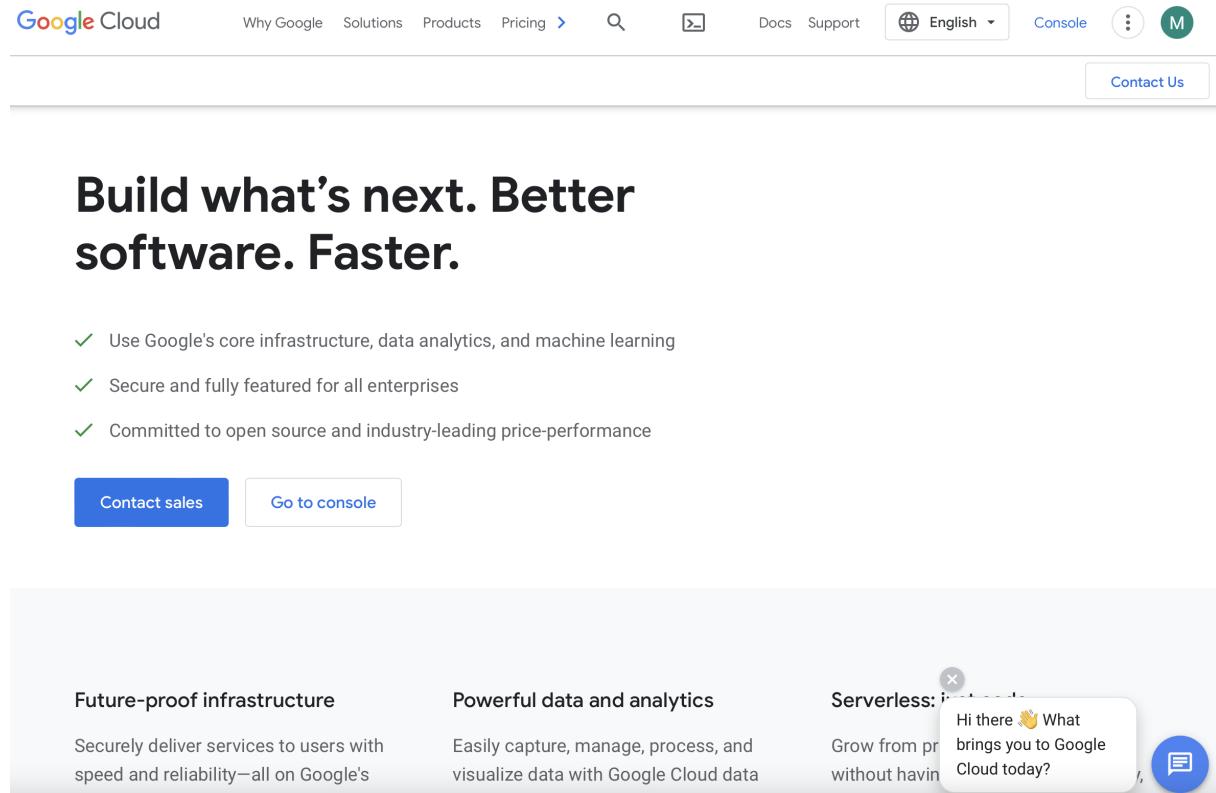
### 6.3 Natural Language Translation Example using Google Service

In order to get started using google translate there are steps for setup.

#### 6.3.1 How to get started using this api.

Navigate to the link

- <https://cloud.google.com>



Google Cloud     Why Google     Solutions     Products     Pricing >     Docs     Support     English     Console     M

Contact Us

# Build what's next. Better software. Faster.

- ✓ Use Google's core infrastructure, data analytics, and machine learning
- ✓ Secure and fully featured for all enterprises
- ✓ Committed to open source and industry-leading price-performance

Contact sales     Go to console

Future-proof infrastructure  
Securely deliver services to users with speed and reliability—all on Google's

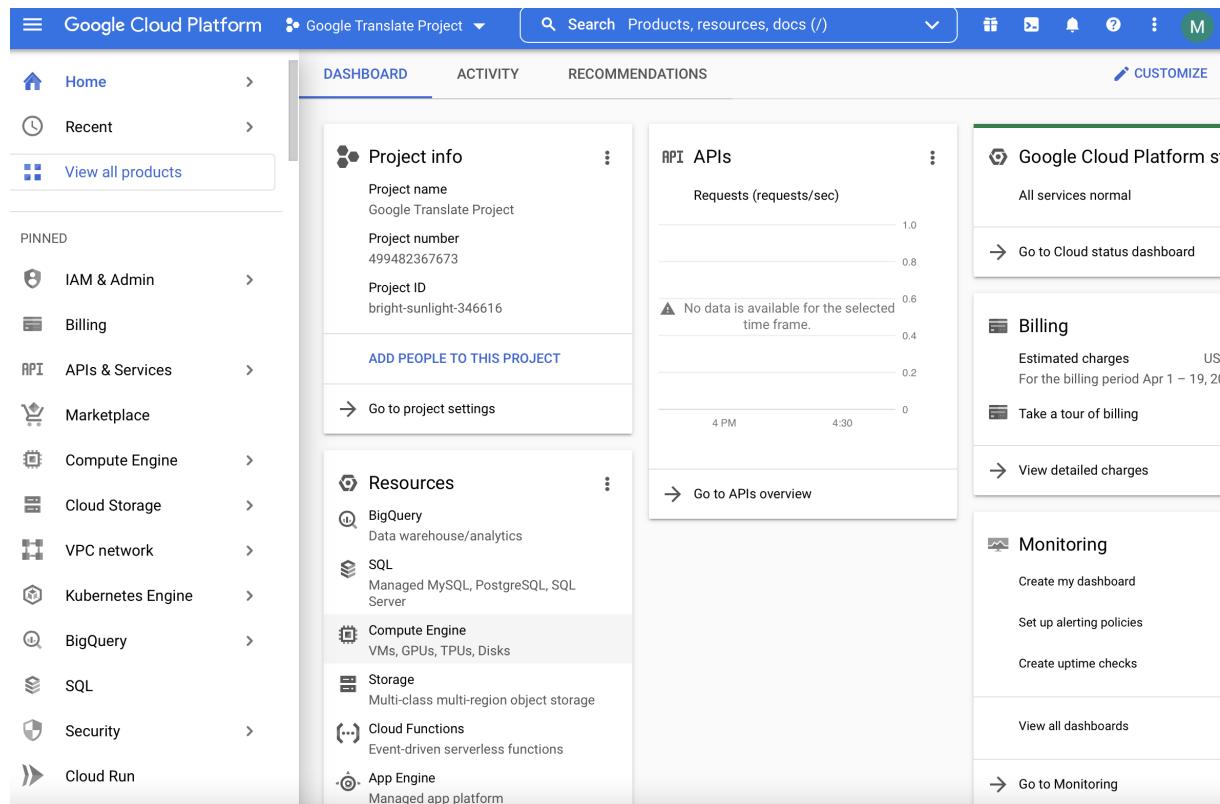
Powerful data and analytics  
Easily capture, manage, process, and visualize data with Google Cloud data

Serverless:  
Grow from pre-built services without havin

Hi there! What brings you to Google Cloud today?

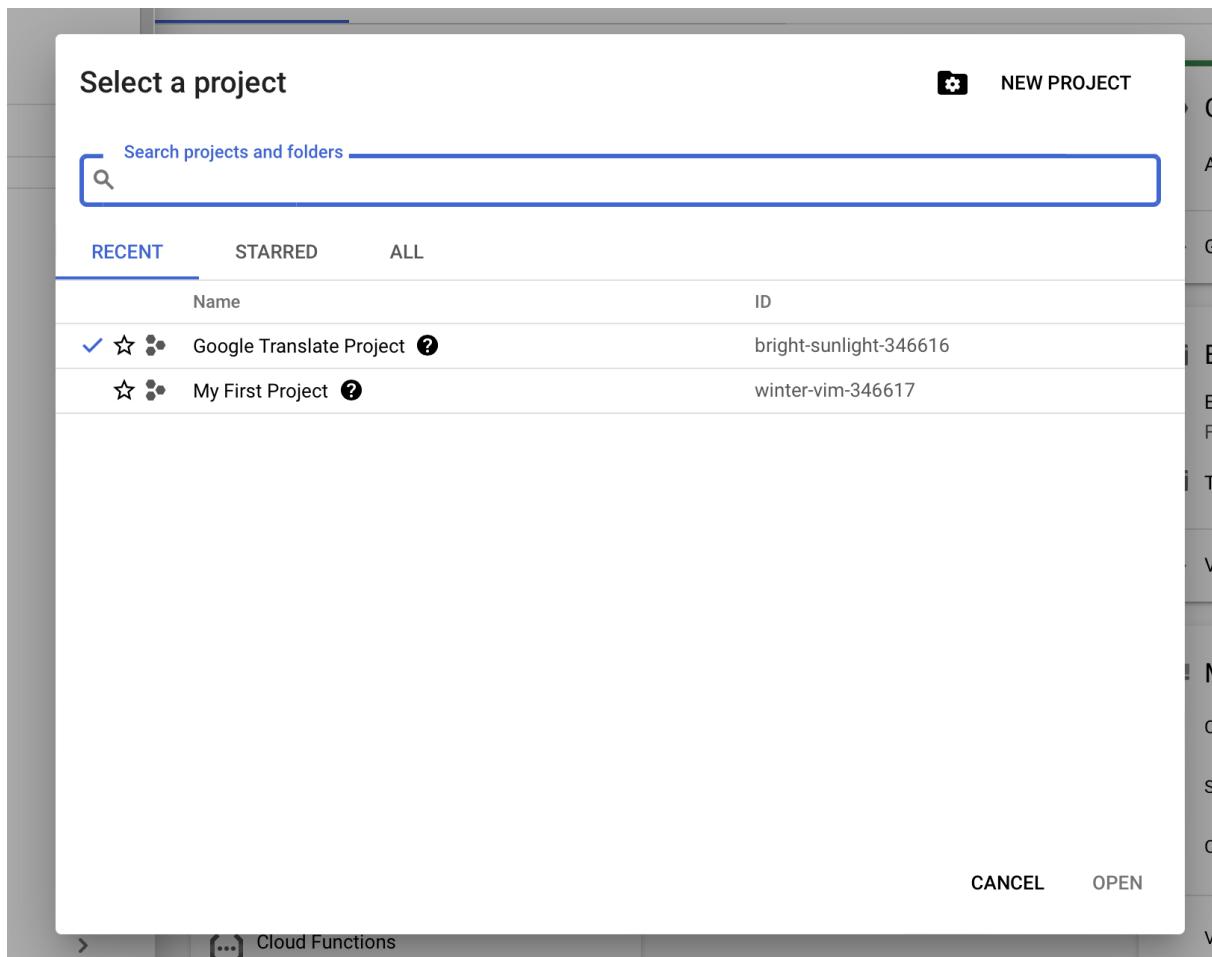
This is the homepage for Google cloud. Will need to activate your console with a Google account and billing.

Google offers a free trial of up to \$300 of language translation to test. After activation of account, you will want to click console in the top right.



Here is the console for all google projects. The next step is to make a new project by selecting the dropdown at the top left.

The project creation page will look like this:



Then, click create new project.

The screenshot shows the 'New Project' page on the Google Cloud Platform. At the top, there's a blue header bar with the Google Cloud logo and a search bar. Below the header, the title 'New Project' is displayed. A warning message in a grey box says: '⚠ You have 23 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)'. Below this, a blue button labeled 'MANAGE QUOTAS' is visible. The main form area starts with a 'Project name \*' field containing 'TEST translation project'. To the right of this field is a question mark icon. Below the project name, it says 'Project ID: psyched-elixir-347823. It cannot be changed later.' followed by an 'EDIT' link. Under the 'Location \*' section, there's a 'No organization' option with a 'BROWSE' button to its right. A note below says 'Parent organization or folder'. At the bottom of the form are two buttons: a blue 'CREATE' button on the left and a white 'CANCEL' button on the right.

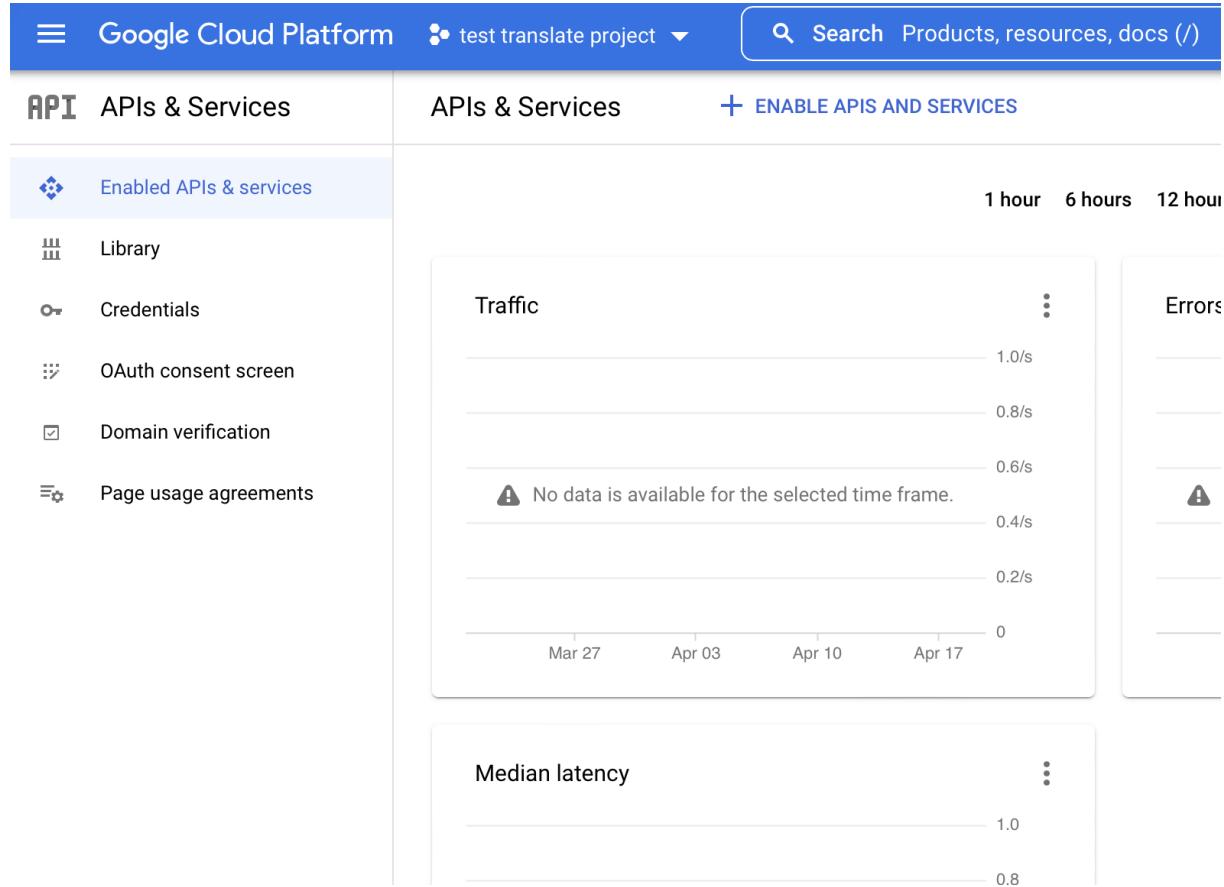
Give your project a title.

The dashboard will automatically update to use your new project, If not, select your project at the top left.

From here we need to activate the api for language translation. Select the api tab in the sidebar to the left.

The screenshot shows the Google Cloud Platform homepage with the project "test translate project". The main navigation bar includes "Home", "Recent", and "View all products". Below this, a "PINNED" section lists several services: "IAM & Admin", "Billing", "APIs & Services", "Marketplace", "Compute Engine", "Cloud Storage", "VPC network", "Kubernetes Engine", "BigQuery", "SQL", "Security", and "Cloud Run". A modal window titled "RECOMMENDATIONS" is open over the page, showing sections for "APIs" and "Cloud Functions". The "APIs" section displays a message: "No data is available for this project". A button in the bottom right corner of the modal says "Go to APIs overview". A dark overlay at the bottom right of the screen states "Now viewing project "test translate project"".

From here you can click the enable apis and services tab at the top



Scroll down to machine learning. Here you will see a cloud translation api. Click the api and enable it.

ary

Search for APIs &amp; Services

Machine learning

## Machine learning

11 results

The screenshot shows a search results page for 'Machine learning' on the Google Cloud Platform. There are 11 results displayed in a grid format:

- AI Platform Training & Prediction API** (Google Enterprise API) - An API to enable creating and using machine learning models.
- Cloud AutoML API** (Google Enterprise API) - Train high-quality custom machine learning models with minimum effort and machine learning...
- Cloud Natural Language API** (Google Enterprise API) - Provides natural language understanding technologies, such as sentiment analysis, entity...
- Cloud Optimization API** (Google) - Solve operational optimization problems rapidly at massive scale
- Cloud Speech-to-Text API** (Google Enterprise API) - Speech recognition
- Cloud Translation API** (Google Enterprise API) - Integrates text translation into your website or application.

The screenshot shows the 'API/Service Details' page for the Cloud Translation API in the Google Cloud Platform. The sidebar on the left shows 'Enabled APIs & services' with 'Library' selected. The main content area displays the following information:

- Cloud Translation API** (Integrates text translation into your website or application. By Google Enterprise API)
- Service name**: translate.googleapis.com
- Type**: Public API
- Status**: Enabled
- CREATE CREDENTIALS** button
- METRICS**, **QUOTAS**, **CREDENTIALS**, **COST** tabs
- Filter options: Select Graphs (4 Graphs), Versions (v1, v2, v3, v3alpha1, and v...), Credentials (Unspecified, Anonymous, ...), Methods (32 options selected)
- Traffic by response code chart showing no data available for the selected time frame.

The api is now enabled, and you will see it on your dashboard.

For each project you will have to enable credentials. this is a vital part to the continuation of the project.

on the tab to the left you will see credentials. here we are going to click create service account.

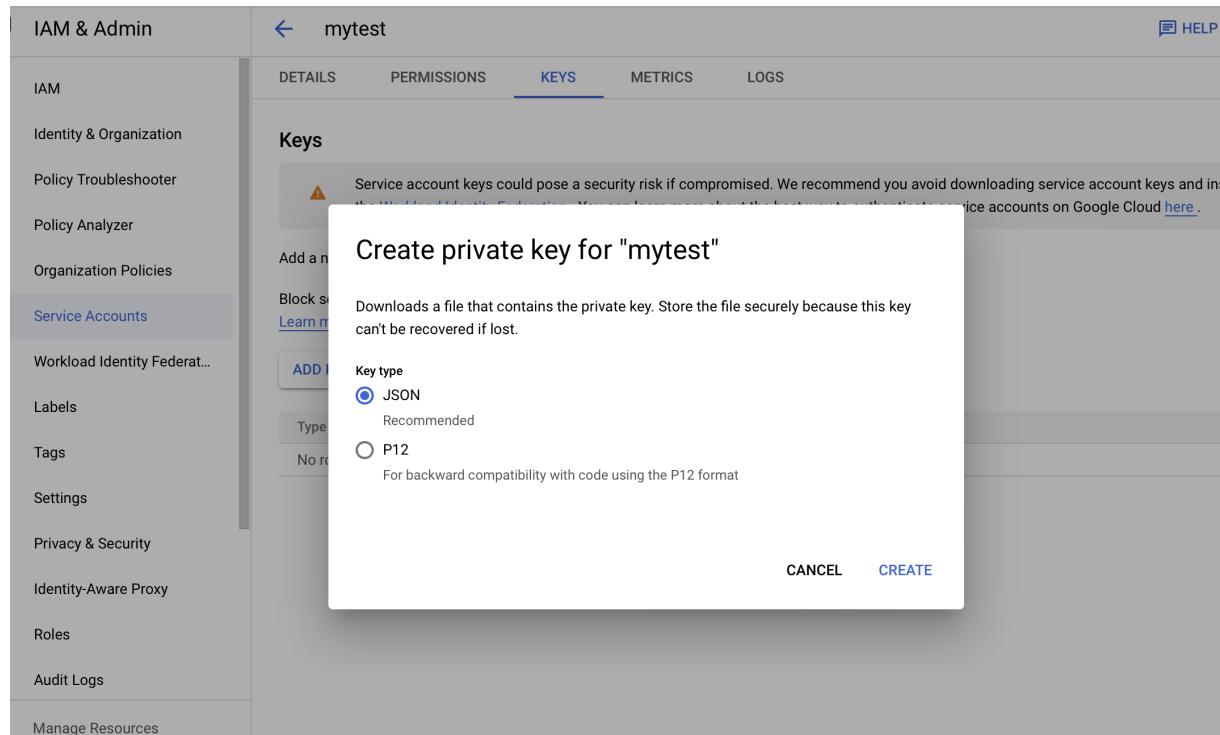
The screenshot shows the 'APIs & Services' section of the Google Cloud Platform console. The 'Credentials' tab is selected. A modal window is open, titled 'Create credentials to access your project'. It contains four options: 'API key' (described as identifying the project using a simple API key), 'OAuth client ID' (described as requesting user consent so the app can access the user's data), 'Service account' (described as enabling server-to-server, app-level authentication using robot accounts), and 'Help me choose' (described as asking a few questions to help decide which type of credential to use). Below the modal, there are sections for 'OAuth 2.0 Client IDs' and 'Service Accounts', both currently showing no items.

The next step is to title your service account. You will proceed and click the role owner.

this step is to decide what role each user has in the terms of the project. owners typically have access to most resources so we are going to select that one.

The screenshot shows the 'Grant this service account access to project (optional)' step of the Google Cloud Platform Service Account creation process. A modal window titled 'Select a role' is open, listing several categories: 'Quick access', 'Currently used' (which is selected), 'Basic', 'By product or service', 'Access Approval', 'Access Context', 'Manager', and 'Actions'. Under 'Currently used', the 'Owner' role is selected. A detailed description of the 'Owner' role is provided: 'Owner' includes 'Full access to most Google Cloud services' and 'included permissions'. A 'DONE' button is visible at the bottom left of the modal.

After the creation of the service account you will be prompted to download the json private key.



This private key will be placed in your Downloads folder. It is a very important piece of information. When creating this private key make sure to download as a json format.

In the command line you must create a virtual environment for the location of this project

```
$ python3.10 -m venv ~/ENV3
$ source ~/ENV3/bin/activate
```

You will need to give this private key a GOOGLE\_CREDENTIALS\_PATH. This is a very important step.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="KEY_PATH"
```

Example:

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/service-account-file.
→ json"
```

In order to use the client library for Natural language translation you will need to install some packages

```
pip install google-cloud-translate==2.0.1
```

or

```
pip install -r requirements.txt
```

From here you should have a linked api-key with a service account, and will be ready to use some examples of natural language programming.

An example for a small natural language program is showcased in [snippets.py](#).

Documentation for getting started in the command line can be found in [CLI Start](#).

## 6.4 Implementation of A Hybrid Cloud natural Language Example

### 6.4.1 How to Implement using Command Line Interface and The Cloudmesh Catalog (AWS Provider)

Step 1: Use the cloudmesh catalog to start the natural language example. [AWS Natural Language Example](#)

Here you will install the catalog first.

On mac

```
python3.10 -m venv ~/ENV3
source ~/ENV3/bin/activate
mkdir cm
cd cm
pip install cloudmesh-installer
cloudmesh-installer -ssh install catalog
```

From here you will install the source packages for this command line reference:

```
pip install -r requirements.txt
```

```
cms help
cms nlp translate --provider=aws --from=en --to=de --region=eu-west-1 hello world
```

The output to this command should look like:

```
{'date': '05/02/2022 14:45:45',
'input': 'hello world',
'input_language': 'en',
'output': 'hallo welt',
'output_language': 'de',
'provider': 'aws',
'time': 0.2641}
Timer: 0.3864s Load: 0.0004s nlp translate --provider=aws --from=en --to=de --region
    ↪ =eu-west-1 hello world
```

### 6.4.2 How to Implement using Command Line Interface and The Cloudmesh Catalog (Google Provider)

Step 1: Use the cloudmesh catalog to start the Natural Language example. [Google Natural Language Example](#)

Here you will install the catalog first.

On mac

```
python3.10 -m venv ~/ENV3
source ~/ENV3/bin/activate
mkdir cm
cd cm
pip install cloudmesh-installer
cloudmesh-installer -ssh install catalog
```

From here you will install the source packages for this command line reference:

```
pip install -r requirements.txt
```

```
cms help
cms nlp translate --provider=google --from=en --to=de --region=eu-west-1 hello world
```

The output to this command should look like:

```
{'date': '05/02/2022 14:45:45',
 'input': 'hello world',
 'input_language': 'en',
 'output': 'Hallo Welt',
 'output_language': 'de',
 'provider': 'aws',
 'time': 0.2641}
Timer: 0.3864s Load: 0.0004s nlp translate --provider=aws --from=en --to=de --region
→ =eu-west-1 hello world
```

### 6.4.3 heterogenous cloudmesh nlp service

In this documentation we have an example of using a natural language operator from different providers. This is a service that is started with cloudmesh catalog. After installation of the catalog there are a list of services that can be used. Using `cms help` on the command line will give that list of services this output will look like:

```
Documented commands (type help <topic>):
```

```
=====
EOF      commands  dryrun  host       nlp      quit    stopwatch  var
banner   config    echo     info      pause   set     sys      version
catalog  debug    gui     inventory  py      shell   sysinfo
clear    default  help    man      q       sleep   term
```

Here there is a newly implemented `nlp` command. This can be accessed by `cms nlp` in the command line. the source code can be found [here](#)

In order to start the translate service from CMS there are a few arguments we will be using. `cms nlp translate` takes the arguments:

```
--provider=google
--from=en
--to=de
--region=eu-west-1
```

The provider is interchangeable from the implemented services ‘aws’, ‘google’ and ‘azure’. The argument `from` takes an interchangeable language code offered from the cloud-providers. This is the language your initial text is decoded in. The argument `to` is the target language the text will be translated to. This argument also takes a language code offered from the cloud providers. A list of Language codes are found here:

Provider	Google	aws
Language	Supported Language Codes	Supported Language Codes
Afrikaans	af-ZA	af-ZA
Arabic, Gulf	ar-AE	ar-AE
Arabic, Modern Standard	ar-SA	ar-SA
Chinese, Simplified	zh-CN	zh-CN
Chinese, Traditional	zh-TW	zh-TW
Danish	da-DK	da-DK
Dutch	nl-NL	nl-NL
English, Australian	en-AU	en-AU
English, British	en-GB	en-GB
English, India	en-IN	en-IN

Provider	Google	aws
English, Irish	en-IE	en-IE
English, New Zealand	en-NZ	en-NZ
English, Scottish	en-AB	en-US
English, South African	en-ZA	en-ZA
English, US	en-US	en-US
English, Welsh	en-WL	en-WL
French	fr-FR	fr-FR
French, Canadian	fr-CA	fr-CA
Farsi	fa-IR	fa-IR
German	de-DE	de-DE
German, Swiss	de-CH	de-CH
Hebrew	he-IL	he-IL
Hindi, Indian	hi-IN	hi-IN
Indonesian	id-ID	id-ID
Italian	it-IT	it-IT
Japanese	ja-JP	ja-JP
Korean	ko-KR	ko-KR
Malay	ms-MY	ms-MY
Portuguese	pt-PT	pt-PT
Portuguese, Brazilian	pt-BR	pt-BR
Russian	ru-RU	ru-RU
Spanish	es-ES	es-ES
Spanish, US	es-US	es-US
Tamil	ta-IN	ta-IN
Telugu	te-IN	te-IN
Thai	th-TH	th-TH
Turkish	tr-TR	tr-TR

When selecting a `region` parameter it is recommended to use `eu-west-1` for best success.

FAST API REDOCS DOCS \* how to stop it \* how to use it \* how to see the documentation with docs and redoc

How to enable the service for google and aws given the previous readmes

## 7 AI

### 7.1 DL Timeseries

---



#### Learning Objectives

- Learn how to use deep learning for time series analysis
- 

create and document simple time series command

use cloudmesh sys command generate to create an extension so we can do a commandline tool

```
cms timeseries --config=CONFIG
```

where the config file is a yaml file.

Work with Gregor as he will create a separate github repo for this and create the command template so you can fill it out with content.

## 8 BIOS

### 8.1 Bios

Please add here a 2-3 paragraph professional Bio. Look up who a professional bio is isn IEEE papers. Write in 3rd person. TOD: provide link example.

Review other peoples bios and improve or give improvement tips where needed.

If it turns out you never contributed to anything, your bio will be removed (as well as your name in this proceedings).

### **8.1.1 Paul Kiattikhunphan**

Paul Kiattikhunphan is a second year at the University of Virginia majoring in computer science.

### **8.1.2 Alex Beck**

Alex Beck has completed his first year at the University of Virginia where he is majoring in electrical engineering. He is set to receive his Bachelor of Science degree in the Spring of 2025. He currently maintains a 3.9 GPA.

Alex is currently working in research this summer at the UVA Biocomplexity Institute's Computing for Global Challenges program under Dr. Gregor Von Laszewski and Dr. Geoffrey Fox where he is planning to gain experience in programming and data science. Prior to that, he has previous experience in sales from working in retail.

At UVA, Alex is involved in a few extracurricular organizations. He is currently active in the Virginia Eta Chapter of Sigma Phi Epsilon, the UVA Climbing Team, and the UVA Chapter of the QuestBridge Scholars Network.

### **8.1.3 Alison Lu**

Alison Lu has completed her second year (Class of 2024) at the University of Virginia pursuing a double major in CS and Chemistry with a minor in Japanese. She is conducting research with the physics department studying quantum computing and photon resolution using machine learning. In addition, she works with UVA's Repair Lab to study gentrification in Norfolk, VA.

She is currently participating in the Biocomplexity Institute's C4GC REU program. Her interests include computer architecture, machine learning, and quantum computing alongside quantum mechanics.

### **8.1.4 Jackson Miskill**

Jackson Miskill has completed his second year at the University of Virginia where he is studying Computer Science and Cognitive Science. He will receive a Bachelor of Arts degree from UVa in Spring of 2024. Jackson has studied python and java in his courses, delving into concepts from basic syntax to data structures and algorithms.

Jackson is currently working at the UVa Biocomplexity Institute under Dr. Gregor von Laszewski as a part of the Computing for Global Challenges program. He is studying the intersection between python and cloud computing. In the future, Jackson plans to continue research.

### 8.1.5 Jacques Fleischer



**Figure 18:** Jacques's Picture

Jacques Fleischer is a sophomore at the Miami Dade Honors College. He is set to receive his associate degree in computer science in summer 2022. He received the Miami Dade Honors College Fellows Award and currently maintains a 4.0 GPA on the Dean's List.

In the summer of 2021, he participated in the Florida-Georgia Louis Stokes Alliance for Minority Participation REU Data Science and AI Research Program; his research focused on predicting the price of cryptocurrency using artificial intelligence. This was done in conjunction with faculty from Florida A&M University and Indiana University. He presented his findings at the Miami Dade College School of Science Symposium in October 2021. Jacques was accepted to the 2022 Emerging Researchers National (ERN) Conference in STEM after applying with his abstract on cryptocurrency time-series. Additionally, he was one of four Miami Dade College students to be nominated for the Barry Goldwater Scholarship due to his research findings.

Jacques is active in extracurriculars; for instance, he is the current Vice President of the MDC Computer Club. There, he hosts virtual workshops on how to use computer software, including Adobe Premiere Pro and PyCharm. He is also a member of Phi Theta Kappa. Furthermore, he is an active contributor to Cloudmesh: an open-source, all-in-one grid-computing solution written in Python. He presently participates in the University of Virginia's Computing for Global Challenges program with Dr. Gregor von Laszewski and Dr. Geoffrey C. Fox to find high performance computing solutions using Raspberry Pis.

### 8.1.6 Eric He

Junyang (Eric) He completed his first year of study at the University of Virginia majoring in Computer Science. He anticipates to receive his B.S. in Computer Science in 2025. He is currently a member of

the Engineering Student Council and the Chinese Student and Scholars Society at UVA.

In the summer of 2021, Eric worked as a Data Analyst intern at Nint (Shanghai) Co., Ltd, a company that provides market data analysis products for E-commerce His work included time series data cleaning and natural language processing.

Eric is currently conducting research with Prof. Geoffrey C. Fox on a Deep Learning model based on LSTM networks trained to predict hydrological features like streamflow, precipitation, and temperature at different locations in the US. He focused primarily on the possibilities of extending the model to countries outside of the US such as Chile and UK.

### **8.1.7 Abdulbaqiy Diyaolu**

AbdulBaqiy Diyaolu is a Computer science and Mathematics Major from Mississippi Valley State University. He will be receiving his bachelor's degrees in both Computer science and Mathematics in the year 2025. AbdulBaqiy is currently a presidential scholar at Mississippi Valley State University and he maintains a 4.0 GPA.

AbdulBaqiy currently works at Fedex Logistics at MVSU. He helps in data entry and data Analysis. He is hoping to polish his data analysis skills with this opportunity. In the summer of 2022, he joins the Bio complexity research program at UVA where he will be able to use his skills in support of different researches, and also learn more research skills along the way.

Abdulbaqiy participates in several extracurricular activities in MVSU he is a member of African Student Union(ASU), National Society of Black engineers (NSBE), and the google developer's club. He is also a Strada scholar at MVSU where he participates in several leadership development activities.

### **8.1.8 Thomas Butler**

Thomas Butler is a Graduate Student at University of Virginia's School of Data Science. His undergraduate degree is in Biomedical engineering. He has over eight years of experience in the Infertility field helping patients, jointly running a Andrology lab, and contributing research to advance the field through joint research on how AMH effects pregnancy outcomes and sperm antibodies effect PSA. He has a certificate in Data Analytics from Georgia Institute of Technology.

### **8.1.9 Robert Knuuti**

Robert Knuuti is a Graduate Student at University of Virginia's School of Data Science. He has over 10 years experience in system architecture and software engineering, and specializes in Development

Operations and Cloud Computing. He has constructed air gapped Continuous Integration and Continuous Delivery systems for multiple organizations each supporting more than 100 developers and has facilitated the construction of repeatable, tractable builds for users of these systems.

#### **8.1.10 Jake Kolessar**

Jake Kolessar is a Graduate Student at the University of Virginia's School of Data Science. He has a background in mechanical engineering and 2 years of experience as a Modeling, Simulation and Analysis Engineer. He has supported the software design and development of modeling capabilities for event simulation products as well as the integration of models into the simulation framework.

## **9 REFERENCES**

