Proceedings of the CyberTraining REU2022

CyberTraining for Students and Technologies from Generation Z

Editors Gregor von Laszewski, Geoffrey C. Fox, laszewski@gmail.com

04 June, 2022

Contents

1 PREFACE **15**
1.1 REU 2022 . . . 15
1.1.1 Overview . . . 15
1.2 Teaching Material and Course Websites . . . 16
1.2.1 Books . . . 17
1.2.2 The repository for our report . . . 17
1.2.3 Slack . . . 17
1.2.4 Asking for technical help . . . 17
1.2.5 Rivanna Information . . . 18
1.2.6 Git Bash . . . 18
1.2.7 PyCharm . . . 18
1.2.8 Bibliography Management . . . 18
1.2.9 Administrative Links . . . 19
1.3 CONTRIBUTION . . . 19
1.3.1 Contributors . . . 19
1.3.2 Calender . . . 19
1.3.3 Avoid Duplication . . . 19
1.3.4 GitHub Insights . . . 20
1.3.5 Issues . . . 20
1.3.6 Teamwork . . . 21

2 DEVELOPMENT ENVIRONMENT **22**
2.1 Install . . . 22
2.1.1 Windows . . . 23
2.1.1.1 Git Bash install . . . 23
2.1.1.2 Python 3.10 install . . . 24
2.1.1.3 Installing cloudmesh . . . 25
2.1.1.4 Uninstall . . . 25
2.1.2 Choco install . . . 25
2.1.3 Install Chocolatey . . . 26
2.1.4 Installing Useful Developer Programs . . . 26
2.1.5 Linux . . . 27
2.1.5.1 Install Python 3.10.5 . . . 27
2.1.5.2 Setting up the venv . . . 27
2.1.5.3 Uninstall . . . 27
2.1.5.4 Update . . . 28

2.1.6 macOS . . . 28
2.1.6.1 Xcode Install . . . 28
2.1.6.2 Cloudmesh . . . 28
2.1.6.2.1 Install . . . 28
2.1.6.2.2 Uninstall . . . 29
2.1.6.3 Update . . . 29
2.1.6.4 Homebrew install . . . 30
2.2 WINDOWS . . . 30
2.2.1 Configuring Windows for Research . . . 30
2.2.1.1 Setting Up the Python Environment . . . 30
2.2.1.2 Install Chocolatey . . . 31
2.2.1.3 Install Git Bash . . . 31
2.2.1.4 Install PyCharm, emacs, and Docker . . . 32
2.2.1.5 Configure PyCharm . . . 32
2.2.1.5.1 Set hard wrap . . . 32
2.2.1.5.2 Makefile Tab configuration and formatting . . . 32
2.2.1.6 Preparing for Virtualization . . . 33
2.2.1.6.1 Docker . . . 33
2.2.1.6.2 WSL . . . 33
2.2.2 Using Git Bash on Windows . . . 34
2.2.2.1 Troubleshooting . . . 35
2.2.3 How to Auto-launch SSH Agent on Windows . . . 35
2.2.4 How to Install WSL Ubuntu 22.04 on Command Line for Windows . . . 36
2.2.5 How to Kill a Process using Command Line on Windows . . . 36
2.2.5.1 Using PID . . . 37
2.2.5.2 Using Process Name . . . 37
2.3 HPC RESOURCES . . . 37
2.3.1 Rivanna . . . 37
2.3.1.1 Links . . . 37
2.3.1.2 Logging in to Rivanna via web interface . . . 38
2.3.1.3 Superpod . . . 39
2.3.1.4 Special DGX Nodes on Rivanna . . . 39
2.3.1.4.1 Starting interactive job . . . 40
2.3.1.5 Activating Rivanna on GitBash . . . 41
2.3.1.6 Installing Python 3.10.5 on Rivanna with ENV3 . . . 42
2.3.1.6.1 Activate Python3 . . . 42
2.3.1.6.2 .bashrc . . . 42
2.3.1.6.3 Installing Cloudmesh into Rivanna . . . 42

2.3.1.6.4 Example Script for Using GPUs . . . 43
2.3.1.6.5 How Do You Activate Different GPUs . . . 43
2.3.1.7 Allocations . . . 44
2.3.1.8 SSH Config . . . 44
2.3.1.9 Links . . . 44
2.3.2 Run Python MPI programs on Rivanna . . . 45
2.3.2.1 SBATCH in Bash Files . . . 45
2.3.2.2 Running in SBATCH . . . 46
2.3.2.3 mlp_mnist.sh . . . 46
2.3.2.3.1 Makefile . . . 47
2.3.3 Run Python MPI programs on Rivanna . . . 48
2.4 DOCKER . . . 48
2.4.1 Running Ubuntu Through Cloudmesh and Docker . . . 48
2.4.1.1 Installing Docker with Chocolatey . . . 48
2.4.1.2 Installing Cloudmesh with Git Bash . . . 48
2.4.1.3 Running Ubuntu with Docker . . . 49
2.4.1.4 Running Ubuntu Remotely on Rivanna . . . 50
2.4.1.5 Links . . . 51

3 LINUX 51
3.1 Cloudmesh Project Ramdisk . . . 51
3.1.1 Ubuntu . . . 52
3.2 USEFUL COMMANDS . . . 52
3.2.1 Linux . . . 52
3.2.1.1 Introduction to Linux . . . 53
3.2.1.2 Github . . . 53
3.2.2 Install pandoc from source . . . 53
3.2.3 Emacs . . . 53
3.2.3.1 Downloading and Installing . . . 53
3.2.3.2 Editing `.zprofile` and `.zshrc` for the REU . . . 53

4 PYTHON 54
4.1 Python Data Management . . . 54
4.1.1 Lists . . . 54
4.1.1.1 Construction . . . 55
4.1.1.2 Accessing Values . . . 55
4.1.1.3 Updating Values . . . 55
4.1.1.4 Python Built-In Methods . . . 55

4.1.2 Dictionaries . . . 55
4.1.2.1 Construction . . . 56
4.1.2.2 Accessing Values . . . 56
4.1.2.3 Updating Values . . . 56
4.1.2.4 Python Built-In Methods . . . 56
4.1.3 CSV Files . . . 56
4.1.3.1 Installing and Importing . . . 57
4.1.3.2 Construction . . . 57
4.1.3.3 Accessing Values . . . 57
4.1.3.4 Examples . . . 58
4.1.4 Links . . . 61
4.1.4.1 Lists . . . 61
4.1.4.2 Dictionaries . . . 61
4.1.4.3 CSV Files . . . 61
4.2 DATA MANAGEMENT MODULES . . . 62
4.2.1 glob . . . 62
4.2.1.1 Glob with asterisk . . . 62
4.2.1.2 Single Character Wildcard (?) . . . 62
4.2.1.3 Escape Characters . . . 63
4.2.1.4 Subdirectories . . . 63
4.2.1.5 Links . . . 64
4.2.2 Mmap . . . 64
4.2.2.1 Reading . . . 64
4.2.2.2 Writing . . . 65
4.2.2.3 Links . . . 67
4.2.3 Pickle . . . 67
4.2.3.1 Encoding Data . . . 67
4.2.3.2 Decoding Data . . . 68
4.2.3.3 Links . . . 68
4.2.4 Shelve . . . 68
4.2.4.1 Creating a New Shelf . . . 68
4.2.4.2 Accessing a Shelf . . . 69
4.2.4.3 Making Shelf Read-Only . . . 69
4.2.4.4 Modifying Shelves . . . 70
4.2.4.5 Closing Shelves . . . 71
4.2.4.6 Links . . . 71
4.2.5 Yaml Database (yamldb) . . . 71
4.2.5.1 Installing and importing . . . 71

4.2.5.2 Using yamldb . . . 71
4.2.5.3 Accessing Values . . . 72
4.2.5.4 Save, Load, and Search . . . 72
4.2.6 Requests with Python . . . 73
4.2.6.1 Installing and Importing . . . 73
4.2.6.2 Using Requests . . . 73
4.2.6.3 Links . . . 74
4.2.7 FastAPI . . . 74
4.2.7.1 FastAPI Install . . . 74
4.2.7.2 FastAPI Quickstart . . . 75
4.2.7.3 Path . . . 76
4.2.7.3.1 Path Arguments . . . 76
4.2.7.3.2 Query and Search Parameters . . . 77
4.2.7.4 Running Through Git bash . . . 78
4.2.7.5 Integration with pedantic . . . 78
4.2.7.6 Running the cc FastAPI service . . . 78
4.2.7.7 Running through Docker . . . 79
4.2.7.8 Testing . . . 80
4.2.7.8.1 Asynchronous Testing . . . 80
4.2.7.9 Links . . . 81
4.2.8 Queue . . . 81
4.2.8.1 FIFO Queue . . . 81
4.2.8.2 LIFO Queue . . . 82
4.2.8.3 Priority Queue . . . 82
4.3 CHARTS . . . 83
4.3.1 Python Graphics Introduction . . . 83
4.3.2 Matplotlib . . . 84
4.3.2.1 Installation . . . 84
4.3.2.2 Import Statements . . . 85
4.3.2.3 Bar Chart . . . 85
4.3.2.4 Line Chart . . . 86
4.3.2.5 Pie Chart . . . 87
4.3.2.6 Contour Plot . . . 88
4.3.2.7 Annotations . . . 89
4.3.2.7.1 Titles . . . 89
4.3.2.7.2 XY Labels . . . 89
4.3.2.7.3 Legend . . . 90
4.3.2.7.4 Rotating Ticks . . . 90

4.3.2.8 Export . . . 91
4.3.2.8.1 Display . . . 91
4.3.2.9 Links . . . 91
4.3.3 Seaborn . . . 91
4.3.3.1 Installation . . . 92
4.3.3.2 Import Statements . . . 92
4.3.3.3 Relational Plots . . . 92
4.3.3.4 Distribution Plots . . . 93
4.3.3.5 Categorical Plots . . . 94
4.3.3.6 Regression Plots . . . 95
4.3.3.7 Saving Figures . . . 96
4.3.3.8 Links . . . 96
4.3.4 Bokeh . . . 96
4.3.4.1 Installation . . . 96
4.3.4.2 Import Statements . . . 96
4.3.4.3 Bokeh Plotting . . . 97
4.3.4.4 Annotations . . . 98
4.3.4.5 Dimensions and Color . . . 98
4.3.4.6 Scatter Plot . . . 98
4.3.4.7 Line Plots . . . 99
4.3.4.8 Bar Chart . . . 100
4.3.4.9 Saving Figures . . . 101
4.3.4.9.1 Saving Figures as PNG . . . 102
4.3.4.10 Links . . . 103
4.3.5 Pandas Graphics . . . 103
4.3.5.1 Installation . . . 103
4.3.5.2 Import Statements . . . 103
4.3.5.3 Bar Chart . . . 103
4.3.5.4 Line Chart . . . 105
4.3.5.5 Pie Chart . . . 106
4.3.5.6 Exporting . . . 106
4.3.5.7 Links . . . 107
4.4 GRAPHS WITH PYTHON . . . 107
4.4.1 Networkx . . . 107
4.4.1.1 Installing and Importing . . . 108
4.4.1.2 Creating a graph . . . 108
4.4.1.3 Accessing . . . 108
4.4.1.4 Removing nodes . . . 109

4.4.1.5 Colors and Labels . . . 109
4.4.2 Graphviz . . . 110
4.4.2.1 Installation and Importing . . . 110
4.4.2.2 Creating the graph, adding nodes, and adding edges . . . 110
4.4.2.3 Subgraphs . . . 112
4.4.2.4 Data Structures . . . 113
4.4.2.5 Colors and Labels . . . 114
4.4.2.6 Links . . . 115
4.5 SELECTED CLOUDMESH TOPICS . . . 115
4.5.1 cms sys command generate . . . 115
4.5.1.1 Creating CLI commands using the cloudmesh command . . . 116
4.5.1.1.1 Example of the CLI Creation . . . 116
4.5.1.1.2 Example of CLI meshing with a python script . . . 117
4.5.2 Cloudmesh StopWatch . . . 118
4.5.3 How to Set Up Browser Defaults on Windows . . . 119
4.5.3.1 Changing Default Browser . . . 119
4.5.3.2 Changing Default Search Engine . . . 119
4.5.3.2.1 Microsoft Edge . . . 119
4.5.3.2.2 Google Chrome . . . 119
4.5.3.2.3 Firefox . . . 120
4.5.3.2.4 Opera . . . 120
4.5.4 CLOUDMESH WORKFLOW . . . 120
4.5.4.1 Documentation . . . 120
4.5.4.2 Workflow . . . 120
4.5.4.2.1 Set-Up . . . 121
4.5.4.2.2 Methods . . . 123
4.5.4.3 Queue Data Structure . . . 124
4.5.4.3.1 Location . . . 124
4.5.4.3.2 Installation . . . 124
4.5.4.3.3 Contents . . . 124
4.5.4.4 Databases Program . . . 125
4.5.4.4.1 Location . . . 125
4.5.4.4.2 Contents . . . 125
4.5.4.4.3 Initialization . . . 125
4.5.4.4.4 Usage . . . 126
4.5.4.4.5 Extension . . . 127
4.5.4.5 Workflow . . . 127
4.5.4.5.1 Set-Up . . . 128

4.5.4.5.2 Methods . . . 130
4.5.4.6 Running Tensorflow on Rivanna . . . 131
4.5.4.6.1 Table of Contents . . . 131
4.5.4.6.2 Activating Rivanna . . . 131
4.5.4.6.3 Installing Tensorflow Using Conda . . . 132
4.5.4.6.4 Loading and Running Singularity . . . 132
4.5.4.6.5 Copying Tensorflow Container Image . . . 132
4.5.4.6.6 Running the Singularity Shell . . . 132
4.5.4.6.7 Running Slurm Jobs . . . 133
4.5.4.6.8 Examples from GitHub . . . 133
4.5.4.6.9 Links . . . 134

5 INTRODUCTION TO DEEP LEARNING 134
5.1 AI . . . 134
5.1.1 Analytics services . . . 135
5.1.2 Reports . . . 135
5.1.3 Modules . . . 135
5.1.4 Course material . . . 135
5.1.4.1 Introduction to AI-Driven Digital Transformation . . . 136
5.1.5 Publistaions . . . 136
5.2 OVERVIEW . . . 136
5.2.1 2022 Lecture on AI-First Deep Learning . . . 136
5.2.2 Overview to AI-Driven Digital Transformation . . . 136
5.2.2.1 Big Data Applications and Analytics (A) . . . 137
5.2.2.2 Big Data Applications and Analytics (B) . . . 137
5.2.2.3 Big Data Applications and Analytics (C) . . . 137
5.2.2.4 Big Data Applications and Analytics (D) . . . 137
5.2.2.5 Big Data Applications and Analytics (E) . . . 138
5.2.2.6 Big Data Applications and Analytics (F) . . . 138
5.2.2.7 Big Data Applications and Analytics (G) . . . 138
5.2.2.8 Big Data Applications and Analytics (H) . . . 139
5.2.3 Overview AI-First Engineering Cybertraining . . . 139
5.2.3.1 Assignments . . . 139
5.2.3.2 Introduction . . . 139
5.2.3.3 Deep Learning Examples, 1 . . . 140
5.2.3.4 Deep Learning Examples, 2 plus Components . . . 140
5.2.3.5 Deep Learning Networks plus Overview of Optimization . . . 140
5.2.3.6 Deep Learning and AI Examples in Health and Medicine . . . 141

5.2.3.7 Deep Learning and AI Examples . . . 141
5.2.3.8 Deep Learning and AI Examples . . . 141
5.2.3.9 GitHub for the Class project . . . 141
5.2.3.10 The Final Project . . . 142
5.2.3.11 Practical Issues in Deep Learning for Earthquakes . . . 142
5.2.3.12 Practical Issues in Deep Learning for Earthquakes . . . 142
5.3 APPLICATIONS . . . 142
5.3.1 Introduction to AI in Health and Medicine . . . 142
5.3.1.1 Introduction (A) . . . 142
5.3.1.2 Diagnostics (B) . . . 143
5.3.1.3 Examples (C) . . . 143
5.3.1.4 Impact of Covid-19 (D) . . . 144
5.3.1.5 Covid-19 and Recession (E) . . . 144
5.3.1.6 Tackling Covid-19 (F) . . . 144
5.3.1.7 Data and Computational Science and Covid-19 (G) . . . 145
5.3.1.8 Screening Drug and Candidates (H) . . . 145
5.3.1.9 Areas for Covid19 Study and Pandemics as Complex Systems (I) . . . 145
5.3.2 Mobility (Industry) . . . 146
5.3.2.1 Introduction (A) . . . 146
5.3.2.2 Self Driving AI (B) . . . 147
5.3.2.3 General Motors View (C) . . . 147
5.3.2.4 Self Driving Snippets (D) . . . 147
5.3.2.5 Electrical Power (E) . . . 147
5.3.2.6 Energy (A) . . . 148
5.3.2.7 Clean Energy startups from Bill Gates (B) . . . 148
5.3.2.8 Space (C) . . . 149
5.3.3 AI In Banking . . . 149
5.3.3.1 The Transition of legacy Banks (A) . . . 149
5.3.3.2 FinTech (B) . . . 149
5.3.3.3 Neobanks (C) . . . 149
5.3.3.4 The System (D) . . . 150
5.3.3.5 Examples (E) . . . 150
5.3.3.6 Banking as a Service (E) . . . 150
5.3.4 Transportation Systems . . . 151
5.3.4.1 Introduction (A) . . . 151
5.3.4.2 Components of a Ride-Hailing System (B) . . . 151
5.3.4.3 Different AI Approaches in Ride-Hailing (C) . . . 151

5.3.5 Commerce . . . 152
5.3.5.1 The Old way of doing things (A) . . . 152
5.3.5.2 AI in Retail (B) . . . 152
5.3.5.3 The Revolution that is Amazon (C) . . . 152
5.3.5.4 DLMalls e-commerce (D) . . . 152
5.3.5.5 Recommender Engines, Digital media (E) . . . 153
5.4 CLOUD COMPUTING . . . 153
5.4.1 Cloud Computing . . . 153
5.4.1.1 Defining the Cloud . . . 153
5.4.1.1.1 Definition . . . 153
5.4.1.1.2 Service-oriented architectures . . . 153
5.4.1.1.3 Market Share . . . 154
5.4.1.2 Virtualization . . . 154
5.4.1.3 Cloud Infrastructure . . . 154
5.4.1.4 Cloud Software . . . 154
5.4.1.5 Cloud Applications . . . 155
5.4.1.6 Parallel Computing Analogies . . . 155
5.4.1.6.1 Parallel Computing in pictures . . . 155
5.4.1.6.2 Real Parallel Computing . . . 155
5.4.1.7 Storage . . . 155
5.4.1.8 HPC and Clouds . . . 155
5.4.1.9 Comparison of Data Analytics with Simulation . . . 156
5.4.1.10 The Future . . . 156
5.5 COLAB . . . 156
5.5.1 Google Colab . . . 156
5.5.2 Python On Colab . . . 157
5.5.2.1 Python Exercise on Google Colab . . . 157
5.5.2.2 Simple For Loop . . . 157
5.5.2.3 List . . . 157
5.5.2.3.1 Retrieving an Element . . . 158
5.5.2.3.2 Append New Values . . . 158
5.5.2.3.3 Remove an Element . . . 158
5.5.2.4 Dictionary . . . 158
5.5.2.4.1 Retrieving an Item by Key . . . 158
5.5.2.4.2 Append New Item with Key . . . 159
5.5.2.4.3 Delete an Item with Key . . . 159
5.5.2.5 Comparators . . . 159
5.5.2.6 Arithmetic . . . 160

5.5.2.7 Numpy . . . 160
5.5.2.7.1 Create a Random Numpy Array . . . 160
5.5.2.7.2 Reshape Numpy Array . . . 161
5.5.2.7.3 Manipulate Array Elements . . . 161
5.5.3 Deep Learning Colab Examples . . . 161
5.5.3.1 Distributed Training for MNIST . . . 161
5.5.3.1.1 Pre-requisites . . . 161
5.5.3.2 Sample MLP + LSTM with Tensorflow Keras . . . 162
5.5.3.2.1 Import Libraries . . . 162
5.5.3.2.2 Download Data and Pre-Process . . . 162
5.5.3.2.3 Define Model . . . 163
5.5.3.2.4 Train . . . 165
5.5.3.2.5 Test . . . 165
5.5.3.2.6 References . . . 166
5.5.3.3 MLP + LSTM with MNIST on Google Colab . . . 166
5.5.3.3.1 Pre-requisites . . . 166
5.5.3.3.2 Sample MLP + LSTM with Tensorflow Keras . . . 167
5.5.3.3.3 Import Libraries . . . 167
5.5.3.3.4 Download Data and Pre-Process . . . 167
5.5.3.3.5 Define Model . . . 168
5.5.3.3.6 Train . . . 169
5.5.3.3.7 Test . . . 169
5.5.3.3.8 References . . . 170
5.5.3.4 MNIST Classification on Google Colab . . . 170
5.5.3.4.1 Import Libraries . . . 170
5.5.3.4.2 Warm Up Exercise . . . 171
5.5.3.4.3 Pre-process data . . . 171
5.5.3.4.4 Identify Number of Classes . . . 171
5.5.3.4.5 Image Reshaping . . . 171
5.5.3.4.6 Resize and Normalize . . . 172
5.5.3.4.7 Create a Keras Model . . . 172
5.5.3.4.8 Compile and Train . . . 173
5.5.3.4.9 Testing . . . 173
5.5.3.4.10 Final Note . . . 174
5.5.3.4.11 Assignments . . . 174
5.5.3.4.12 References . . . 174
5.5.3.5 MNIST With PyTorch . . . 174
5.5.3.5.1 Import Libraries . . . 174

5.5.3.5.2 Pre-Process Data . . . 174
5.5.3.5.3 Define Network . . . 175
5.5.3.5.4 Define Loss Function and Optimizer . . . 176
5.5.3.5.5 Train . . . 176
5.5.3.5.6 Model Evaluation . . . 177
5.5.3.5.7 References . . . 177
5.5.3.6 MNIST-AutoEncoder Classification on Google Colab . . . 178
5.5.3.6.1 Prerequisites . . . 178
5.5.3.6.2 Import Libraries . . . 178
5.5.3.6.3 Download Data and Pre-Process . . . 178
5.5.3.6.4 Define Model . . . 179
5.5.3.6.5 Train . . . 181
5.5.3.6.6 Test . . . 181
5.5.3.6.7 Visualize . . . 181
5.5.3.7 MNIST-CNN Classification on Google Colab . . . 182
5.5.3.7.1 Prerequisites . . . 182
5.5.3.7.2 Import Libraries . . . 182
5.5.3.7.3 Download Data and Pre-Process . . . 182
5.5.3.7.4 Define Model . . . 183
5.5.3.7.5 Train . . . 185
5.5.3.7.6 Test . . . 185
5.5.3.8 MNIST-LSTM Classification on Google Colab . . . 185
5.5.3.8.1 Pre-requisites . . . 185
5.5.3.8.2 Sample LSTM with Tensorflow Keras . . . 185
5.5.3.8.3 Import Libraries . . . 186
5.5.3.8.4 Download Data and Pre-Process . . . 186
5.5.3.8.5 Define Model . . . 187
5.5.3.8.6 Train . . . 187
5.5.3.8.7 Test . . . 188
5.5.3.8.8 References . . . 188
5.5.3.9 MNIST-MLP Classification on Google Colab . . . 189
5.5.3.9.1 Pre-requisites . . . 189
5.5.3.9.2 Sample MLP with Tensorflow Keras . . . 189
5.5.3.9.3 References . . . 192
5.5.3.10 MNIST-RMM Classification on Google Colab . . . 192
5.5.3.10.1 Prerequisites . . . 193
5.5.3.10.2 Import Libraries . . . 193
5.5.3.10.3 Download Data and Pre-Process . . . 193

5.5.3.10.4 Define Model . . . 194
5.5.3.10.5 Train . . . 195
5.5.3.10.6 Test . . . 195
5.5.3.10.7 References . . . 196
5.6 NLP . . . 196
5.6.1 AWS Translate Service . . . 196
5.6.1.1 Step 1: Creating a new iam user account on aws. . . . 196
5.6.1.1.1 Step 2: Creating a access key ID and secret ID . . . 197
5.6.2 Azure Translate . . . 200
5.6.2.1 Installing and Starting Azure Translate through the command line . . . 202
5.6.2.1.1 Step 2: installing using Homebrew . . . 202
5.6.3 Google Translation Service . . . 204
5.6.4 Implementation of A Hybrid Cloud natural Language Example . . . 210
5.6.4.1 How to Implement using Command Line Interface and The Cloudmesh Catalog (AWS Provider) . . . 210
5.6.4.2 How to Implement using Command Line Interface and The Cloudmesh Catalog (Google Provider) . . . 211
5.6.4.3 heterogenous cloudmesh nlp service . . . 212
5.7 TIMESERIES . . . 214
5.7.1 DL Timeseries . . . 214

6 MNIST 215

6.1 Setting Parameters while Running `mnist` Jobs . . . 215
6.1.1 Having Cloudmesh Installed . . . 215
6.1.2 Installing the `reu2022` folder . . . 215
6.1.3 Setting the `user`, `host` parameters . . . 215
6.1.4 Setting the `cpu` and `gpu` parameters . . . 216
6.2 Setting Parameters while Running `mnist` Jobs . . . 217
6.2.1 Table of Contents . . . 217
6.2.2 Having Cloudmesh Installed . . . 217
6.2.3 Installing the `reu2022` folder . . . 217
6.2.4 Setting the `user`, `host` parameters . . . 218
6.2.5 Setting the `cpu` and `gpu` parameters . . . 218
6.2.6 Running the `mnist` Files . . . 219
6.3 MNIST tensorflow rivanna . . . 219
6.4 Run Python MPI programs on Rivanna . . . 219
6.5 MNIST pyTorch Rivanna . . . 219

7 BIOS **220**
7.1 Bios . . . 220
7.1.1 Alex Beck . . . 220
7.1.2 Alison Lu . . . 220
7.1.3 Jackson Miskill . . . 220
7.1.4 Jacques Fleischer . . . 221
7.1.5 Eric He . . . 222
7.1.6 Abdulbaqiy Diyaolu . . . 222
7.1.7 Thomas Butler . . . 222
7.1.8 Robert Knuuti . . . 223
7.1.9 Jake Kolessar . . . 223

8 REFERENCES **223**

1 PREFACE

1.1 REU 2022

1.1.1 Overview

Information technology is playing a dramatically increasing role in society, industry, and research. This includes design and use of large databases, simulations and artificial intelligence applications hosted on clouds and supercomputers with convergent technologies. Correspondingly, there is an increasing need for research workforce job skills in these and related areas. This project takes course material on this cyberinfrastructure and adapts it for training with an emphasis on the needs of under-represented communities. The techniques of the successful open-source software movement are used to create sustainable communities around the course curriculum and software. The project is creating new technologies to enable this for today's generation of students. Skills in core cloud computing, big data, supercomputing and artificial intelligence are exemplified by applications in the life science and nanotechnology areas. This project enables the future research workforce to contribute effectively using advanced cyberinfrastructure, promoting the progress of science and advancing the national health, prosperity, and welfare, which serves the national interest, as stated by NSF's mission.

The future economic progress and research leadership of the U.S. is dependent on having a research workforce that is capable of making use of advanced cyberinfrastructure (CI) resources as articulated by the National Strategic Computing Initiative (NSCI). This requires a curriculum that changes and integrates modern concepts and practices for the new generation of students aiming at a "data-enabled computational science and engineering" expertise. This project takes what Indiana University has

learned from a brand new four-year undergraduate engineering curriculum designed ab initio and taught so far to its first two undergraduate classes, and invests it into developing active training modules. The innovative curriculum integrates big data, simulations, clouds and high performance computing systems presented in a uniform framework. The course material is customized for communities of cyberinfrastructure researchers nucleated, built, and sustained via the dynamic use of GitHub and enhanced by innovative tools to build a novel learning management system optimized for cyberinfrastructure-intensive classes. The project modules include Cloud Computing, Big Data Applications and Analytics, Networking, High-Performance Computing, Artificial Intelligence/Machine Learning, and Information Visualization. There are residential sessions, with a call for participants, and purely online courses and these have both "teach the student" and "teach the teacher" modes; the latter enables easy spread of the classes. Hands-on learning with research projects built around the class material is fully supported. The project offers CyberTraining with all the popular approaches used by the Apache Software Foundation, including Meetups and Hackathons. Modules for domain scientists and engineers, e.g., the cyberinfrastructure users that exploit advanced CI methods for research in nanoengineering and bioengineering are included. Both students and teachers contribute to the course improving the text, the software, including a unique set of examples and the project aims to show that one can build both learning and sustainability communities by using the proven techniques of the open source software community. The project uses proactive measures to enhance the involvement of under-represented communities in its activities.

This award reflects NSF's statutory mission and has been deemed worthy of support through evaluation using the Foundation's intellectual merit and broader impacts review criteria.

This grant was started under NSF Award 1829704 and transitioned to the University of Virginia in November 2021 under the NSF award number 2200409 at the Biocomplexity Institute at the University of Virginia. Administrative support was provided by the BII UVA Global Challenges program. For most students, the program lasted from June 2nd, 2022 until July 29th, 2022. One student was funded for an additional month.

1.2 Teaching Material and Course Websites

Teaching material form previous teaching activities was made available to the students at the beginning of the program. We list here a selection of them. Students were required to find other resources on the internet in case additional material was needed. This served as an exercise to be able to identify relevant information.

Students main goal was to develop new teaching material while at the same time conducting a project making the use of distributed cyberinfrastructure for deep learning simpler. Over time, they contributed to the teaching material.

1.2.1 Books

The following teaching books were used:

- 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

1.2.2 The repository for our report

The repository managing these proceedings is available at

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

Code was mainly developed in

- https://github.com/cloudmesh/cloudmesh-mpi
- https://github.com/cloudmesh/cloudmesh-cc

Students maintained furthermore their own repositories which are listed at the end of these proceedings.

1.2.3 Slack

Some private Slacks that are not open to the community to coordinate activities include

- Private Slack for student collaboration
- Private GC4G slack for event scheduling

1.2.4 Asking for technical help

The many UVA technical help ticket systems include

- [ITS Ticket system](https://virginia.edusupportcenter.com/shp/uva/helpcenter}
- Rivanna Ticket System
- BII Ticket system

1.2.5 Rivanna Information

Information about the Rivanna supercomputer with GPUs at University of Virginia are listed next

- 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/

1.2.6 Git Bash

Students with Windows machines were asked to use GitBash

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

1.2.7 PyCharm

Students were asked to use PyCharm for development.

- PyCharm Professional Edition: https://www.jetbrains.com/pycharm/download
- PyCharm Markdown Manual: <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.

1.2.8 Bibliography Management

Bibliography was collected as bibtex and the following tools were recommended

- 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.

1.2.9 Administrative Links

The following links are only important for Gregor von Laszewski to enable account management.

- 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/

1.3 CONTRIBUTION

1.3.1 Contributors

The contributor list is likely incomplete if your name is missing, please let us know and we add your name by sending mail to `laszewski@gmail.com`.

You can also look at the various github monitoring lists to identify contributors.

Please note that a large amount of information is additionally provided in other collections. The collections can be customized with cyberaide-bookmanager written by Gregor von Laszewski (laszewski@gmail.com) for this purpose.

- Gregor von Laszewski, Jacques P. Fleischer, Jackson Miskill, Alison Lu, Alex Beck, AbdulBaqiy Diyaolu, Robert Knuuti

1.3.2 Calender

- June 2 2022, Program start
- July 28 2022, Program end

1.3.3 Avoid Duplication

Before you do anything, check first if it is already in one of the books. Evaluate if it needs to be expanded in the book, or if a new section is needed. If a new section is needed, please consult with Gregor and ask for approval. We will then also decide if the chapter will be located in reu2022 or in book (both are repositories).

1.3.4 GitHub Insights

Contribution will be determined based on review of commits, and lines contributed in such a fashion that the final lines will be considered. For example let us assume you spent significant time on a section that ia a duplication of what others do, we will then delete that sections and you have not achieved a contribution. Thus, please make sure that you contribute valuable things.

Ask if you have an idea and want confirmation.

To see the contributions, you can look at GitHub Insights:

- Project
- Issues
- Pulse
- Contributors
- Traffic

However, note that the true contributions may not be determined form insights as it may include duplication of lines and other information that is not included in GitHub Insights.

1.3.5 Issues

Cloudmesh contains a convenient program to list all issues of repositories downloaded in the current directory. The command can be installed with

```
$ pip install cloudmesh-git
```

alternatively you can simply install it from source with

```
$ cd cm
$ git clone git@github.com:cloudmesh/cloudmesh-git.git
$ cd cloudmesh-git
$ pip install -e .
$ cd ..
```

To run it cd to the directory where all your git repos are located and say

```
$ cms git issues --repo=. --refresh
```

This will open a Web page that list all issues across all repos in that directory.

In case you like to see all issues only for the REU2022, please check out the GitHub Link.

To get all issues for cloudmesh projects that interest you, you can create an issue list from selected repositories.

For example you can checkout the following

```
$ cd cm
$ cloudmesh-installer get cmd5
$ git clone git@github.com:cloudmesh/yamldb.git
$ git clone git@github.com:cloudmesh/cloudmesh-git.git
$ git clone git@github.com:cloudmesh/cloudmesh-catalog.git
$ git clone git@github.com:cloudmesh/cloudmesh-data.git
$ git clone git@github.com:cloudmesh/cloudmesh-sbatch.git
$ git clone git@github.com:cloudmesh/cloudmesh-mpi.git
$ git clone git@github.com:cloudmesh/cloudmesh-slurm.git
$ git clone git@github.com:cloudmesh/cloudmesh-pi-burn.git
$ git clone git@github.com:cloudmesh/cloudmesh-pi-cluster.git
$ git clone git@github.com:cloudmesh-community/book.git
$ git clone git@github.com:cybertraining-dsc/reu.git
$ git clone git@github.com:cyberaide/bookmanager.git
```

To get the list use

```
$ cms git issues --repo=reu --refresh
```

1.3.6 Teamwork

Learning Objectives

- Principals of Teamwork

---

1. A team works together
2. A team always communicates with each other
3. A canceled meeting for a dasy does not mean that the teamwork stops
4. If you need help you must ask the team
5. If a team member does not want to help you that is not team work
6. If you can not keep up with the team members you need to step up
7. If you are assigned a task and you can not do it you need to ask for help

8. If you spend hours or days on a task that you can not do you need to ask for help
9. In case you know you have an issue with a task you need to communicate that early
10. In case you need to go on vacation you need to ask for permission and do it 3 weeks before the vacation so that reassignment of tasks is possible.
11. In case you do need to go on vacation, you need to provide a concrete plan on (a) how does this work effect the team and how will your work be distributed to the team, (b) what are concrete workhours that you propose to make up the work (c) be aware that any vacation time not only negatively impacts the team time planing, but also your supervisors work

    (d) plan any additional impact on your work, for example you could have other obligations in unrelated projects.

12. A statement such as `This was not my task` if something is not working clearly shows that the spirit of teamwork is not understood. Instead you should have communicated with the person doing the task and helped complete it.
13. A statement such as `I am better then the other team members, they slow me down` is not in the team spirit. In fact practice shows that in advanced project stages such student do not accelerate as they can not benefit from the team and the team does not like to work with them.
14. Research today is always done as team. Not understanding teamwork excludes you from becoming a researcher.

Always ask yourself everyday:

- What have you done to increase teamwork?
- What is a concrete contribution you made for the team?
- Did you help anyone?
- Were you helped by anyone?
- Did you communicate your progress to the team?e

2 DEVELOPMENT ENVIRONMENT

2.1 Install

Learning Objectives

- Learn how to set up a python development computer
- 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 First, make sure you completely uninstall previous versions of Git Bash. If you do not have it installed, you can skip this step.

```
Go to Start Menu > Add Or Remove Programs
```

In the search bar that is located on that page, type Git. Then click on Uninstall and follow the instructions.

To install Git Bash, it is highly recommended to use chocolatey for convenience. If you do not have chocolatey then follow the tutorial at https://chocolatey.org/install. Then run the following command in a Run as Administrator instance of Powershell:

```
$ choco install git.install --params "/GitAndUnixToolsOnPath \
        /Editor:Nano /PseudoConsoleSupport /NoAutoCrlf" -y
```

Otherwise, instead of using chocolatey, download Git Bash from the following link:

- 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 installation.
- Click `Next` to accept the default text editor which is vim,
- Replace the default branch name (`master`) with `main` ,
- Click `Next` and check on to run Git from the command line and 3rd party software,
- Click `Next` and check on to use the OpenSSL library,
- Click `Next` and check on to `Checkout as-is, commit as-is` ,
- Click `Next` and check on to use MinTTY,
- Click `Next` and check on to use the default git pull,
- Click `Next` and check on to use the Git Credential Manager Core,
- Click `Next` and check on enable file system caching, and then
- Click `Next` and check on Enable experiment support for pseudo consoles
- Click `Install` .

A video tutorial on how to install Git and Git Bash on Windows 10 is located at https://youtu.be/HCotEx_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 -rf ~/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 once 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 next.

```
$ 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. Here are the steps to activate the virtual environment and install cloudmesh into the virtual environment called ENV3 which we will use for all of our lessons.

```
$ 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 To remove your virtual environment, execute the following command:

```
$ rm -rf ~/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 To uninstall the virtual environment, execute the following command:

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

2.1.6.3 Update Before starting this process, ensure that python is in the correct path. We will test this 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 WINDOWS

2.2.1 Configuring Windows for Research

Unlike Linux and macOS, Windows runs on a completely different OS. Many coding environments are adapted to Linux, so Windows users must properly configure their machine to prepare it for a project. This is of special importance when working in environments supporting distributed cyberinfrastructure. Here in many cases Linux is required.

2.2.1.1 Setting Up the Python Environment Often you need a specific version of Python. If in doubt, please install the newest one. At time of writing this document it is Python 3.10.5.

Please download and install it from `python.org`. We recommend that you uninstall Anaconda if you used that before and use the verison from python.org instead.. Development is easier when using a native Python installation instead of anaconda/conda.

To uninstall anaconda, press the Windows key and type "Add or remove programs". Then, press Enter and search for `conda` in the "Search this list" box. Remove everything related to anaconda. Note that anaconda may have set some environment variables or added configuration scripts to your `.bashrc` files in case you use gitbash. Please, remove them and make sure your Python version from python.org works as expected.

To code in Python, we recommend using PyCharm and not VSCode.

Pycharm and Git Bash can be installed with the instructions found in *Install.*

Installation may be simplified while using chocolatey.

This includes

- gitbash
- pycharm
- emacs
- docker

Before installing docker however you have to set up the appropriate hypervisor at boot time. PLease let us know how you set them for your machine, so we can add some information here. You will likely have to research it.

Please also know that you MUST uninstall virtualbox before you install docker, as old versions of virtualbox are incompatible with docker and it is just easier to uninstall virtualbox and reinstall it.

Next we summarize the installations using chocolatey.

Before installing anything, we recommend that you read the entire section. Especially when installing docker and if you do not have a brand new computer.

2.2.1.2 Install Chocolatey

To install chocolatey, follow the tutorial at https://github.com/cybertraining-dsc/cybertraining-dsc.github.io/blob/main/content/en/docs/tutorial/reu/chocolatey/index.md

2.2.1.3 Install Git Bash

Git Bash must be installed with specific configurations, as the discrepancy between Windows and other operating systems can cause errors during runtime, if not properly configured. If Git Bash is already installed, uninstall it by pressing the Windows key and typing `Add or remove programs` (and press Enter). Then locate and uninstall Git Bash.

To install Git Bash with chocolatey, issue the following command:

```
$ choco install git.install --params "/GitAndUnixToolsOnPath \
        /Editor:Nano /PseudoConsoleSupport /NoAutoCrlf" -y
```

The `/NoAutoCrlf` sets it so that the files are not downloaded via git with Windows line endings. If it were downloaded with such line endings, then it would cause programming bugs. Programmers should Commit As-Is to avoid conflicts.

2.2.1.4 Install PyCharm, emacs, and Docker

Uninstall PyCharm Community version if already installed on the computer by pressing the Windows key and typing `Add or remove programs` (and press Enter). Then locate and uninstall PyCharm.

The following command installs PyCharm Professional, among other necessary development programs. To install these programs in an easy manner, issue the following command (you must have chocolatey installed):

```
$ choco install pycharm emacs docker-desktop -y
```

PyCharm is advantageous over other IDEs such as VSCode because students receive the professional version of PyCharm for free. Furthermore, PyCharm offers robust features such as Refactor and Inspect Code.

A guide to activating PyCharm with a free professional license is available at https://youtu.be/QPESX-VBnEU

2.2.1.5 Configure PyCharm

2.2.1.5.1 Set hard wrap

Press `Ctrl + Alt + S` in PyCharm and expand the `Editor` menu on the left-hand side. Then, click on `Code Style` and enter `79` in the `Hard wrap at:` box. Also, check the `Wrap on typing` checkbox.

This is done so that the text in files is uniformly indented at 79 columns.

2.2.1.5.2 Makefile Tab configuration and formatting

To change what the Tab key does in a Makefile, open a Makefile in PyCharm and click on `Tab` in the bottom right of the PyCharm interface. If you cannot find the `Tab` button, then click on `View` in the top-right, go to `Appearance`, and make sure `Status Bar` is checked.

After clicking the `Tab` button in the bottom-right, click on `Configure Indents for Makefile...` Tab size should be 4.

If PyCharm fails to render your Makefile correctly, right-click on the Makefile in your open files tabs and click `Override File Type`. If you cannot find `Makefile` in the list, you must install the `Makefile Language plugin` for PyCharm.

2.2.1.6 Preparing for Virtualization

2.2.1.6.1 Docker To enable virtualization for Docker on Windows machines, some preparations must be made. First, if the user has VirtualBox installed it is suggested that they uninstall it and reinstall later if necessary. Some older versions of VirtualBox do not support other virtual images like Windows Subsystem for Linux (WSL).

Next, the BIOS settings must be changed to enable virtualization. To do this, search `Advanced startup` in the Windows Search Bar and click `Restart now`. Click `Troubleshoot` and `Advanced startup options` and then `UEFI Firmware Settings` to get into the BIOS. NOTE: These are not exhaustive instructions because computer brands and hardware differ vastly. The main objective is to get into the BIOS and search for any `Virtualization` or `Hyper V` options in Windows BIOS configuration. For example, Lenovo brand laptops have a `Configure` tab in the BIOS and the virtualization settings must be enabled under that menu. Then, the user must exit the BIOS while saving changes.

Better documentation on enabling virtualization, which is recommended by Docker and created by Berkeley, is located at https://bce.berkeley.edu/enabling-virtualization-in-your-pc-bios.html

Lastly, check Windows features with `Turn Windows features on or off`. For Docker, `Hyper-V` and `Containers` must be enabled.

2.2.1.6.2 WSL WSL is a Linux virtual image designed for Windows. WSL 2 is typically used as opposed to WSL 1. To install, type this into administrative PowerShell:

```
PS> wsl --install
```

To install a particular distribution, use `wsl --install -d <DistroName>` instead. The available distributions can be found with

```
PS> wsl --list --online
```

After WSL is installed, it can be accessed by typing `wsl` in Powershell. More documentation can be found in the Microsoft Official Documentation.

Directories in WSL

WSL creates a Linux environment in your Windows directory. To access your directories with WSL, a special syntax is used. For example, your home directory, typically `C:\Users\USERNAME` and abbreviated to `~` is the following with WSL: `/mnt/c/Users/USERNAME/`. So to change directories to the Desktop in WSL, use this command:

```
$ cd /mnt/c/Users/USERNAME/Desktop
```

where `USERANME` is to be replaced with the name of the user.

2.2.2 Using Git Bash on Windows

Git Bash is the terminal of choice for the Windows operating system. However, it must be properly configured for an optimal Python development experience; for example, Pseudo Console Support must be enabled.

First, try opening Git Bash and entering `wsl` (you must have installed WSL and Git Bash). If it hangs, then a reinstall of Git Bash is necessary. If it does not hang and instead launches into WSL, then Pseudo Console Support is enabled; however, make sure that you picked other necessary options when you installed Git Bash.

To enable Pseudo Console Support, uninstall Git Bash.

If you installed Git with `choco`, then do `choco uninstall git` and `choco uninstall git.install`.

If you did not install Git with `choco` and you instead used the installer wizard from the Git website, then press the Windows key, searching for `Add or remove programs`, searching for `Git`, clicking on it, then clicking `Uninstall` and completing the uninstallation wizard.

Then install Git Bash in a Run as Administrator instance of Powershell by executing the choco command:

```
$ choco install git.install --params "/GitAndUnixToolsOnPath \
        /Editor:Nano /PseudoConsoleSupport /NoAutoCrlf" -y
```

If you do not have chocolatey then follow the tutorial at https://chocolatey.org/install and then run the aforementioned choco command, or forego chocolatey and use the standard Git installer by checking the box that reads `Enable experiment support for pseudo consoles`. Also select the appropriate options that match what is specified in the previously written choco command.

Now you can use `wsl` and other commands that would otherwise require `winpty` prepended to the command.

2.2.2.1 Troubleshooting If an `already installed` message appears when trying to use choco to install Git Bash, such as

```
git.install v2.33.0.2 already installed.
 Use --force to reinstall, specify a version to install, or try upgrade.
```

then try `choco uninstall git`. Then rerun the previously listed `choco install` command. If that does not work, consider the `--force` parameter mentioned in the warning message.

2.2.3 How to Auto-launch SSH Agent on Windows

Copy the following code:

```
env=~/.ssh/agent.env

agent_load_env () { test -f "$env" && . "$env" >| /dev/null ; }

agent_start () {
    (umask 077; ssh-agent >| "$env")
    . "$env" >| /dev/null ; }

agent_load_env

# agent_run_state: 0=agent running w/ key; 1=agent w/o key; 2=agent not running
agent_run_state=$(ssh-add -l >| /dev/null 2>&1; echo $?)

if [ ! "$SSH_AUTH_SOCK" ] || [ $agent_run_state = 2 ]; then
    agent_start
    ssh-add
elif [ "$SSH_AUTH_SOCK" ] && [ $agent_run_state = 1 ]; then
    ssh-add
fi

unset env

alias emacs="C:/ProgramData/chocolatey/bin/emacs.exe"
alias tree="cmd //c tree.com //a //f"
source ~/ENV3/Scripts/activate
cd ~/cm
```

Then, using Git Bash, run `nano ~/.bashrc`, use the down arrow key to ensure you are at the bottom of the file on a new line, and then paste the contents using `Shift + Insert`. Consider, beforehand, even deleting the `.bashrc` if you have a preexisting script that may conflict with the new additions and then creating a new `.bashrc` with the above contents.

Keep in mind that if emacs is not installed using choco, the emacs alias will not function. You can install emacs using choco by running `choco install emacs`. This will not work if you have not installed choco, which you can do by following https://chocolatey.org/install

Additionally, the source command will not work if you have not created a virtual Python environment named `ENV3` in your home dir.

The `cd` will also not function if `cm` dir does not exist in the home dir. If so, then execute `mkdir ~/cm`.

2.2.4 How to Install WSL Ubuntu 22.04 on Command Line for Windows

We assume that you have installed Git Bash with support for pseudo console. If not or you are not sure, we recommend that you reinstall Git Bash with this tutorial:

https://github.com/cybertraining-dsc/reu2022/blob/main/project/git-bash-pseudo-console.md

We also need you to enable Linux line ending support through "NoAutoCrlf" (which is included in the previously linked tutorial) as some of our cloudmesh programs when interacting with wsl need this to be switched on too. After you have installed Git Bash with the right options, you can install WSL Ubuntu 22.04.

Execute each command on Git Bash:

```
curl -o ~/ubuntu-base-22.04-base-amd64.tar.gz https://cdimage.ubuntu.com/ubuntu-base
    ↪ /releases/22.04/release/ubuntu-base-22.04-base-amd64.tar.gz
mkdir -p ~/wsl/ubuntu-22.04/instances
wsl --import ubuntu-22.04 ~/wsl/ubuntu-22.04/instances ~/ubuntu-base-22.04-base-
    ↪ amd64.tar.gz
rm ~/ubuntu-base-22.04-base-amd64.tar.gz
```

Then you can run Ubuntu 22.04 by executing `wsl -d ubuntu-22.04`

If Git Bash freezes, pseudo console support is not enabled. In such a case, execute the `wsl -d ubuntu-22.04` command on Powershell. However you will not have all features you need to run cloudmesh smoothly. Instead we recommend to reinstall Git Bash with the proper options.

2.2.5 How to Kill a Process using Command Line on Windows

To see the list of running processes and their PIDs on Windows, press `Windows key + X` and then press the `T` key. Then, click on the `Details` tab. You may also run `tasklist` on the command line, but with many processes, it may be difficult to sort through.

2.2.5.1 Using PID If using Git Bash, replace all slashes in the command with double-slashes (because Git Bash otherwise interprets them as paths instead of flags). Otherwise, this slash replacement is not necessary in cmd.exe and Powershell.

To kill a process by specifying its PID, execute:

```
taskkill /f /pid PIDNumberGoesHere
```

2.2.5.2 Using Process Name To kill a process by specifying the process name, execute:

```
taskkill /f /im ProcessNameGoesHere
```

For example, you can start an instance of the Windows calculator and then run:

```
taskkill /f /im calculator.exe
```

2.3 HPC RESOURCES

2.3.1 Rivanna

Rivanna is UVA's High Performance Computing (HPC) System. Rivanna is used to run programs that require a lot of memory or computing power. This tutorial will go through the basics of Rivanna, from obtaining access to running specialized jobs.

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

---

2.3.1.1 Links Here is a comprehensive list of useful links on how to navigate the Rivanna system.

- Presentations
  - Knuuti[1]

  - UVA Rivanna presentation, June 8th, 2022 [2]

- Intro to Rivanna
- Rivanna Allocations
- Rivanna Dashboard
- Globus Transfer
- Rivanna SLURM Overview
- Queues Documentation
- Documentation
- Login
- UVA VPN
- Shell access
- JupyterLab
- UVA HPC Support Email

2.3.1.2 Logging in to Rivanna via web interface The user must install Duo Mobile on smartphone to use as an authentication service to approve logins (Figure 1). For security reasons we suggest never saving the password within the browser autofill.

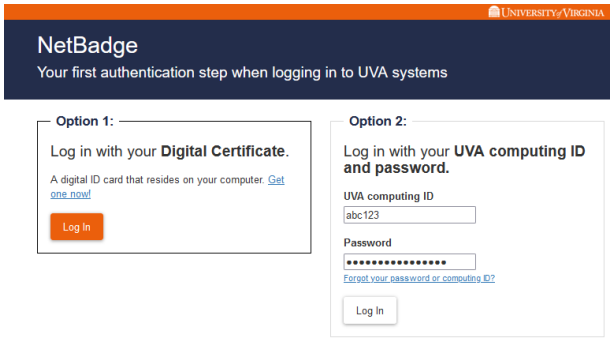


Figure 1: UVA Login

After logging in at https://rivanna-portal.hpc.virginia.edu/, 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. You will be prompted to enter your UVA password (not your ssh-key password). 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$
```

2.3.1.3 Superpod A new resource will be integrated into rivanna. It is an NVIDIA Superpod. The Estimated deployment for testing by the end of this summer 2022.

Hardware Components include:

- 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.

2.3.1.4 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)
```

2.3.1.4.1 Starting interactive job To start interactive jobs on our special partition you can use the following command. Please note that interactive jobs get charged even if you are not typing anything in. Furthermore, you may block valuable time for others. Possibly a better strategy is to use other partitions as they have larger numbers of GUS.

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

```
$ ijob --reservation=bi_fox_dgx --account bii_dsc --partition=bii-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                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     13486      G   /usr/bin/X                         63MiB |
|    0   N/A  N/A     13639      G   /usr/bin/gnome-shell               21MiB |
+-----------------------------------------------------------------------------+
```

2.3.1.5 Activating Rivanna on GitBash Once you have Rivanna installed locally on your computer with your account set up, if you have Cloudmesh installed, you can simply connect to Rivanna on the terminal using the following command. Make sure to run the terminal as Admin or else it won't work.

```
$ cms vpn connect
```

In order to disconnect from Rivanna, simply use the command:

```
$ cms vpn disconnect
```

2.3.1.6 Installing Python 3.10.5 on Rivanna with ENV3 After you log in into Rivanna, there's a chance you're running Rivanna on an outdated version of Python, restricting you from being able to run Python files properly. In order to do so, you'll need a fully to up-to-date version of Python with the ENV3 virtual environment.

2.3.1.6.1 Activate Python3 First Python3, must be activated and can be done by typing in the following:

```
$ ssh rivanna
rivanna$ module load cuda cudnn
rivanna$ module load anaconda
rivanna$ conda create -n ENV3  -c conda-forge python=3.10.5
```

The last command may take a while. After it's done installing, activate the ENV3 virtual environment of Python 3.10.5 by typing in:

```
rivanna$ source activate ENV3
```

2.3.1.6.2 .bashrc Now, you must open your `.bashrc` file. If it doesn't exist, create one. Using any editor, open it and copy and paste the following:

```
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi


PS1="\s-\v\$"
alias vi='vim'
module load cuda cudnn
module load anaconda
source activate ENV3
```

2.3.1.6.3 Installing Cloudmesh into Rivanna Once Python 3.10.5 is installed with the ENV3 virutal environment. Cloudmesh can now be installed by first making a `cm` directory, installing the Cloudmesh Installer through pip and using the installer to get the Cloudmesh files into the `cm` directory. All of this can be done using the following commands:

```
$ mkdir cm
$ cd cm
$ pip install cloudmesh-installer
```

```
$ cloudmesh-installer --ssh get cms
```

If the last command doesn't work due to access rights, type in:

```
$ cloudmesh-installer get cms
```

Additional packages can be similarly installed with

```
$ cloudmesh-installer get cc
$ cloudmesh-installer get sbatch
```

2.3.1.6.4 Example Script for Using GPUs Slurm jobs can be run in Rivanna through `sbatch` scripts. Various arguments such as the job, output, GPU, time, account, etc. can be specified as shown in the example script below.

```
#!/usr/bin/env bash
#SBATCH --job-name=mydemojob
#SBATCH --output=%u-%j.out
#SBATCH --error=%u-%j.err
#SBATCH --partition=gpu
#SBATCH -c 1
#SBATCH --gres="gpu:v100:1"
#SBATCH --mem=4GB
#SBATCH --time=3:00
#SBATCH --account=c4gc

# your automation code here..
python mydemojob.py
```

2.3.1.6.5 How Do You Activate Different GPUs In the `sbatch` scripts, different GPUs can be activated by specifying it through `gres` argument.

```
#SBATCH --gres="gpu:p100:1"
#SBATCH --gres="gpu:v100:1"
#SBATCH --gres="gpu:k80:1"
#SBATCH --gres="gpu:a100:1"
#SBATCH --gres="gpu:a100:1"
#SBATCH --gres="gpu:rtx2080:1"
```

Assignment:

- How can we use a100 with 40 GB vs. 80 GB?
- How can we use the special a100 nodes?

2.3.1.7 Allocations The time you spend on Rivanna is allocated as Service Units (SUs) into the account you're using. When you type in the following command:

```
$ allocations
```

The results show you which accounts you're currently under and the amount of SUs is in balance, reserved, and available as shown below.

```
Account                    Balance        Reserved       Available
-----------------          ---------      ---------      ---------
xyz_community                99999               0           99999
comp4gc                     146527               0          146527
```

You are also able to access the information of all the users under the accounts you're under by typing in the following command:

```
$rivanna allocations -a comp4gc
```

2.3.1.8 SSH Config SSH Config makes it easier for users to connect to Rivanna by specifying arguments in the `config` file instead of typing it in the terminal.

The `.ssh` directory and `config` file can be created using the following commands:

```
$rivanna mkdir -p ~/.ssh && chmod 700 ~/.ssh
$rivanna touch ~/.ssh/config
$rivanna chmod 600 ~/.ssh/config
```

The `config` can then be edited using any terminal text editor of your choice, using the following command. Nano is used in this example.

```
$rivanna nano ~/.ssh/config
```

Inside the `config` file, copy and paste the following:

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

2.3.1.9 Links

- ssh Config File[3]
- Roberet Knuuti, Presentation on how to use Rivanna
- UVA Staff, UVA Rivanna presentation, June 8th, 2022

2.3.2 Run Python MPI programs on Rivanna

Learning Objectives

- Learn how to submit Slurm jobs on Rivanna through SBATCH
- Learn how to run SBATCH through Bash files
- Learn how view the queue of running Slurm jobs

2.3.2.1 SBATCH in Bash Files SBATCH is a way to submit jobs onto Rivanna's compute nodes. With any call, there must be specifications for the job. Take this example `mlp_mnist.sh` file, which runs a python program `mlp_mnist.py`.

```
#!/bin/sh
#SBATCH --job-name=mlp_mnist.sh
#SBATCH --output=mlp_mnist_%A.log
#SBATCH --error=mlp_mnist_%A.error
#SBATCH --partition=gpu
#SBATCH --gres=gpu:k80:1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4GB
#SBATCH --time=3:00

echo "# cloudmesh status=running progress=1 pid=$SLURM_JOB_ID"

nvidia-smi
source activate ENV3
python mlp_mnist.py

echo "# cloudmesh status=done progress=100 pid=$SLURM_JOB_ID"
#
```

The name of the `.sh` file, as well as the output and error files are specified with the first three `#SBATCH` lines. The `.log` and `.error` files are additionally marked with `%A`, which will be replaced with the job's unique id.

The `--partition` specifies which queue the job will enter and thus the type of resources it is allowed. More information can be seen in Figure **??**.

`![Slurm partitions](images/partitions.png){#fig:slurm-partitions}`

The `--gres` is used for jobs that use the GPU. The last number specifies the number of cores and the middle specifies which GPU will be used. It can be left blank to use the default. Some GPUs are *A100, V100, K80, P100,* and *P8.*

The `--cpus-per-task` and `--mem` specify the CPUs and memory to be allocated, and the `--time` sets the max time allowed for Rivanna to complete the task, or else it will stop. For `gpu` partitioned jobs, this can be up to three days.

2.3.2.2 Running in SBATCH

To run this bash file `mlp_mnist.sh`, use this command:

```
rivanna $ sbatch mlp_mnist.sh
```

This submits the job to the partition, and it can be watched with either of the following commands:

`watch squeue -u ${USER}` or `squeue -u ${USER}`

These commands pull up all the jobs you have submitted. "Watching" will refresh this list every 2 seconds and can be exited with `Ctrl-C`. When the job has finished, there should be associated `.error` and `.log` files specified with the `#SBATCH` commands.

This specific bash file includes the `nvidia-smi` command which pulls up a chart with the NVIDIA GPU specifications. For example, here are the P8 details.

```
+-----------------------------------------------------------------------------+
| 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 GeForce ...  Off  | 00000000:62:00.0 Off |                  N/A |
| 29%   25C    P8    11W / 250W |      7MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

2.3.2.3 mlp_mnist.sh

```
#!/bin/sh
#SBATCH --job-name=mlp_mnist.sh
#SBATCH --output=mlp_mnist_%A.log
#SBATCH --error=mlp_mnist_%A.error
#SBATCH --partition=gpu
#SBATCH --gres=gpu:k80:1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4GB
```

```
#SBATCH --time=3:00

# A100, V100, K80, P100: set in gres
echo "# cloudmesh status=running progress=1 pid=$SLURM_JOB_ID"

#module purge
#module load singularity

## Assuming that the container has been copied to the user's /scratch directory
#CONTAINERDIR=$HOME
#singularity run --nv $CONTAINERDIR/tensorflow-2.8.0.sif $HOME/experiment/mlp_mnist/
    ↪ mlp_mnist.py

nvidia-smi
# pip install cloudmesh-common
source activate ENV3
which python
python -V
python mlp_mnist.py

echo "# cloudmesh status=done progress=100 pid=$SLURM_JOB_ID"
#
```

2.3.2.3.1 Makefile

```
IMAGE=tensorflow-2.8.0
image:
        cp ${IMAGE}.sif ${IMAGE}-cms.sif
        singularity exec ${IMAGE}-cms.sif pip install cloudmesh-common -U

mlp_mnist:
        cd experiment/mlp_mnist; sbatch mlp_mnist.sh

watch:
        watch squeue -u ${USER}

cat:
        cd experiment/mlp_mnist; cat *.error *.log
```

Time for GPUs

GPU	# Cores	Time to run mlp_mnist.sh
P8	1	16.521 seconds
K80	1	40.519 seconds

GPU	# Cores	Time to run mlp_mnist.sh
A100	1	24.557 seconds
V100	1	21.811 seconds
P100	1	31.927 seconds

2.3.3 Run Python MPI programs on Rivanna

see the book Python MPI

add chapter if not there

2.4 DOCKER

2.4.1 Running Ubuntu Through Cloudmesh and Docker

2.4.1.1 Installing Docker with Chocolatey Docker is a platform that allows users to develop, publish, and run software applications in packages known as containers.

In order to install Docker, first have Chocolatey installed. After it is installed, go on Windows PowerShell and run as an Administrator and type in the following:

```
choco install docker-desktop -y
```

If another host operating system is being used, go on the official Docker Desktop website and download the installation file from there.

Here is the link to the download website: * https://docs.docker.com/get-docker/

Next, make sure open Docker Desktop. Wait for the `Docker Desktop Starting` screen to finish. If it hasn't been done already, create a Docker account on their official website and log on to the account on the application after.

2.4.1.2 Installing Cloudmesh with Git Bash Cloudmesh is a repository that can be downloaded and will be used to run Ubuntu on Docker. In order to download it, first, make sure Git Bash is downloaded so Unix commands can be run on non-Unix platforms through Bash. It's a good idea to download it specifically with Chocolatey and with Pseudo Console Support. Instructions can be found in the Sources section.

After Git Bash is installed, create a directory called `cm` using the command:

```
cd ~
mkdir cm
```

Go into the `cm` directory and create the `cloudmesh-cc` folder using the following commands and let the folder install.

```
cd cm
git clone git@github.com:cloudmesh/cloudmesh-cc.git
```

2.4.1.3 Running Ubuntu with Docker Using Git Bash, go into the docker directory using the following command. This will lead to the `docker/cloudmesh` directory under cloudmesh.

```
cd ~/cm/cloudmesh-cc/docker/cloudmesh
```

Now, install the `Make` tool using the command:

```
choco install make -y
```

This tool controls the generation of executables from the program. It helps builds and installs programs through the use of Makefiles.

After `Make` is installed, it's a good idea to clean out any existing containers and images in Docker that'll interfere with running Ubuntu. This can be done using the command:

```
make cleanall
```

After cleaning everything, Ubuntu and Python will be installed onto a Docker container, an isolated package of software exclusively on Docker. Unix commands are now possible to execute on Docker after installing Ubuntu. This process can take a while. In order to install, type in the following command:

```
make image
```

Wait for everything to install. This is where commands differ based on the user's host operating system. If Windows is used, type in the following command to start up Ubuntu from Git Bash.

```
make wshell
```

If Windows is not used, type in the following command:

```
make shell
```

Now, Ubuntu should be being run through Docker and Unix commands can now be executed. Because everything now takes place in Unix, a different directory now exists under Ubuntu. Due to that, type in the following command to be directed to `cloudmesh-cc` to that.

```
pwd
cd ~
pwd
```

The following output should be produced. The first output shows the home host directory on Windows while the second output shows the Unix host directory.

```
/home/USERNAME
/root
```

At this point, pytests of the various software modules of `cloudmesh-cc` can now be executed without having to worry about the host operating system. For example, in order to test out the `Job` class for `wsl` under the file, type in the following command. All the tests should be able to pass.

```
cd cm/cloudmesh-cc
pytest -v -x --capture=no tests/test_job_wsl.py
```

2.4.1.4 Running Ubuntu Remotely on Rivanna In order to run Unix commands remotely, specifically Rivanna, the University of Virginia's Rivanna main supercomputing platform, first, have it installed. Instructions for installation are provided under the Sources section. A UVA account with a computing ID must be created first.

After Rivanna is installed, open Cisco AnyConnect Secure Mobility Client and connect to the UVA Anywhere VPN.

After that, in any directory, type in the following:

```
ssh-keygen
```

After the prompt shows up, press `ENTER` until all the prompts are through. Then, specifically type in the following:

```
ssh-copy-id COMPID@rivanna.hpc.virginia.edu
```

A prompt will ask if the user wants to continue connecting. Type in "yes". After that, it will ask for the user's Rivanna password, type it in. Once that is done, the user is now connected to Rivanna through Docker.

2.4.1.5 Links

- Get Docker [4]
- Getting Started [5]
- Introduction to Docker [6]
- Setting Up GitBash
- Rivanna Set Up

3 LINUX

3.1 Cloudmesh Project Ramdisk

Learning Objectives

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

THis is an easy assignment to develop a cloudmesh command that sets up a RAMDISK in various operating systems. To facilitate this task we provide the first initial information.

The command should look similar to

```
$ cms ramdisk --... --size=SIZE
```

We recommend that you use python humanize to make it easier to read file sizes such as 1GB (without humanize, the number of bytes is displayed instead of GB, making it hard to comprehend).

You could enhance the program to also integrate it into the OS so it starts up immediately after reboot. However for now a simple on demand program to start and stop the ramdisk is sufficient.

Advanced integration could include

- dynamic ramdisk no reboot needed, but if reboot, ramdisk needs to be set up new
- ramdisk integrated in fstab with reboot
- 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}`')
```

3.1.1 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/
```

Links that could be useful include, but are not limited to

- Information about using /dev/shm is available at <http://ubuntuguide. net/ubuntu-using-ramdisk-for-better-performance-and-fast-response>
- Various methods (in german) are documented at https://wiki.ubuntuusers.de/RAM-Disk_erstellen
- ramfs is an older file system type and is replaced in mostly by tmpfs.
- Windows https://forums.guru3d.com/threads/guide-using-imdisk-to-set-up-ram-disk-s-in-windows-with-no-limit-on-disk-size.356046/

Assignment:

3.2 USEFUL COMMANDS

3.2.1 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. Please coordinate. Create a pull request with your contribution.

3.2.1.1 Introduction to Linux

Please read the following book and contribute to it.

- Linux Book

3.2.1.2 Github

PLease read the following tutorials and improve if needed

- Git Pull Requests
- Git ssh Command
- GitHub

3.2.2 Install pandoc from source

releases are at https://github.com/jgm/pandoc/releases

```
sudo apt purge pandoc
wget https://github.com/jgm/pandoc/releases/download/2.18/pandoc-2.18-1-amd64.deb
sudo dpkg -i pandoc-2.18-1-amd64.deb
```

3.2.3 Emacs

emacs is a command line editor that can be used to write bash scripts which are then used for specific purposes.

In the REU program, we used emacs to create changes in the `.zshrc` and `.zprofile` (for mac/linux) and the `.bashrc` and `.bashprofile` (for Windows).

3.2.3.1 Downloading and Installing

In order to download and install emacs, there are two paths that one can take.

For Mac:

- Go to Aquamacs, which is a website that allows Mac users to download and install emacs.
- After that, follow the GUI installation instructions and then you should be able to launch the application from the command line or from the GUI dashboard.

3.2.3.2 Editing `.zprofile` and `.zshrc` for the REU

The `.zprofile` and `.zshrc` bash scripts are scripts that are run upon starting up a terminal for mac (the command line). In order to set up this for the correct usage, we will be editing these scripts to contain the correct virtual environment so that we can correctly use it.

`.zprofile`

```
# Setting PATH for Python 3.9
# The original version is saved in .zprofile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.9/bin:${PATH}"
export PATH

# Setting PATH for Python 3.10
# The original version is saved in .zprofile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.10/bin:${PATH}"
export PATH

PATH="~/ENV3/bin:${PATH}"
export PATH

source ~/ENV3/bin/activate
```

`.zshrc`

```
PATH="~/ENV3/bin:${PATH}"
export PATH

source ~/ENV3/bin/activate
```

In essence, this is sets up the `.zshrc` and `.zprofile` scripts to have the correct virtual environment.

If you have a different OS, Aquamacs also has other versions to support those OSes, but you could also use vi or another text editor.

4 PYTHON

4.1 Python Data Management

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.

4.1.1 Lists

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

4.1.1.1 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.

4.1.1.2 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.

4.1.1.3 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.

4.1.1.4 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.

4.1.2 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.

4.1.2.1 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.

4.1.2.2 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_dicionary.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)

4.1.2.3 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.

4.1.2.4 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()`.

4.1.3 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.

4.1.3.1 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
```

4.1.3.2 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)
```

4.1.3.3 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.

4.1.3.4 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 2 and Figure 3.

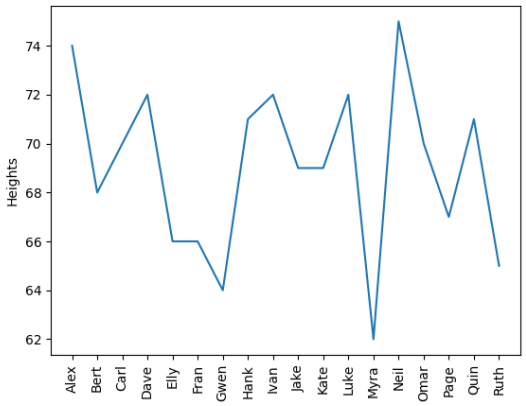


Figure 2: CVS Line plot

Figure 2 is created using the data available here.

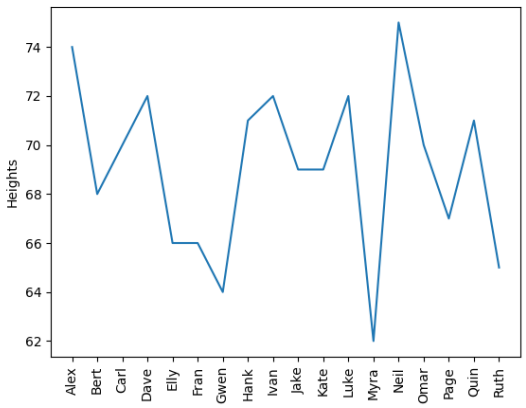


Figure 3: CSV dict line plot

Figure 3 is 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 4:

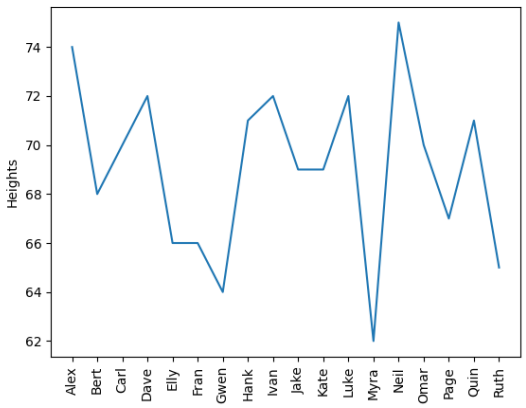


Figure 4: pandas-lineplot

Figure 4 is created using the data available here.

4.1.4 Links

4.1.4.1 Lists

- List Manipulation [7]
- Python List and Array Methods [8]

4.1.4.2 Dictionaries

- Dictionary Manipulation in Python [9]
- Python Dictionary Methods [10]
- Update a dictionary [11]

4.1.4.3 CSV Files

- Create a dataframe [12]
- CSV File Reading and Writing [13]
- CSV File Overview [14]
- Python Built in Read CSV Files [15]
- Extracting Information from a CSV File [16]
- List Index out of Range [17]
- Visualizing Data in CSV [18]

4.2 DATA MANAGEMENT MODULES

4.2.1 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.

4.2.1.1 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
```

4.2.1.2 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
```

4.2.1.3 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
```

4.2.1.4 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
```

4.2.1.5 Links

- Glob Documentation[19]

4.2.2 Mmap

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

4.2.2.1 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 '
```

4.2.2.2 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!
```

4.2.2.3 Links

- mmap documentation[20]

4.2.3 Pickle

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

4.2.3.1 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 ... .'
```

4.2.3.2 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
```

4.2.3.3 Links

- Pickle Documentation[21]

4.2.4 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.

4.2.4.1 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'.

4.2.4.2 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}
```

4.2.4.3 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
```

4.2.4.4 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}
```

4.2.4.5 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()
```

4.2.4.6 Links

- Shelve Documentation[22]
- Python Object Persistence

4.2.5 Yaml Database (yamldb)

Yamldb is a python package that allows a user to store variables on a computer's hard-drive, as opposed to having variables stored in memory.

Yamldb makes use of the yaml data file, and it stores values in the yaml files so that they can be easy to use. The yaml files can be accessed like python dictionaries, which makes the process significantly easier for the user.

The yamldb package is used in the `cloudmesh-cc` section of the `cloudmesh` repository and can be accessed here.

4.2.5.1 Installing and importing Yamldb is easy to install. Simply execute the following command:

```
pip install yamldb
```

Then, to import yamldb to a python file, simply execute the following code:

```
from yamldb import YamlDB
```

Now, following these commands, the computer is set up to run and access yamldb.

4.2.5.2 Using yamldb Yamldb has several methods within itself that allow the user easy access to creating a yaml file and storing things within it. These things can be anything, really. The methods include: `Yamldb()`, `.get()`, `.load()`, `.save()`, and `.search()`. All of these functions can be used. The following is example code of how someone might use yamldb for their own code.

```
from yamldb import YamlDB

filename = 'PATH'
database = YamlDB(filename=filename)

database['queue1'] = 'job1'
database['queue2.job1'] = 'echo hello world'

d = database.get('queue1', default=3)
```

This code can be accessed from GitHub.

Which creates:

```
queue1: job1
queue2:
  job1: echo hello world
```

4.2.5.3 Accessing Values As one can see, these values are easy to access. Let's say, for instance, that we were trying to access the `queue2:job1` string `"echo hello world"`. To do this, we would execute the following:

```
job = database.get('queue2')
print(job)

command = database['queue2'].get('job1')
print(command)
```

This code can be accessed from GitHub.

Which returns:

```
{'job1': 'echo hello world'}
echo hello world
```

4.2.5.4 Save, Load, and Search With the `yamldb` implementation, it is simple to save, load, and search. The following code shows this output.

```
from yamldb import YamlDB

filename = 'PATH'
database = YamlDB(filename=filename)
```

```
database['queue1'] = 'job1'
database['queue2.job1'] = 'echo hello world'

d = database.get('queue1', default=3)

database.load(filename)  # loads the file
database.save(filename)  # saves the file
database.search('queue1') # searches the file for the query
```

This code can be accessed from GitHub.

The `.load()` function loads in the current `.yaml` file that was created originally. The `.save()` function saves the any updates that were made to the `.yaml` file.

Outside of that, this should be the fully implementation of the `yamldb` python package.

The overview of this python package was accessed from here.

4.2.6 Requests with Python

The `requests` library is a python library that allows the user to interact with the internet via HTTP very easily. Typically, users specify the type of request they are making via their programming, but with the `requests` library, it is much easier to program for user enhancement.

4.2.6.1 Installing and Importing The `requests` library can be easily installed from the command line and imported at the top of any python file.

The following script can be used to install `requests`:

```
$ python -m pip install requests
```

Once this has been done, it is simple to import the library into a python file. Simply execute the following:

```
import requests
```

4.2.6.2 Using Requests The `requests` library has many functions that allow it to be utilized in the way it is. Typical functions are `requests.put()`, `requests.head()`, `requests.delete()`, `requests.get()`, and `requests.options()`. What the `requests` library does is it properly encodes everything that is programmed by the user.

A basic example of how requests could be utilized within a program:

```
import requests
r = requests.get("https://api.github.com/events")
r = requests.post('https://httpbin.org/post', data={'key': 'value'})
r = requests.put('https://httpbin.org/put', data={'key': 'value'})
r = requests.delete('https://httpbin.org/delete')
r = requests.head('https://httpbin.org/get')
r = requests.head('https://httpbin.org/get')
```

This code can be accessed from GitHub

For requests, an `r` value of 200 means that the method was a success whereas a 100 means not a success.

In addition to these functions, the `requests` library has a plethora of other functionalities. It can send more data, convert to `json` files, execute `patches`, etc. The documentation for the `requests` library can be found here. This link is also where the examples are drawn from.

4.2.6.3 Links

- Requests Documentation

4.2.7 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.

4.2.7.1 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]"
```

If this command does not work, use this alternative:

```
bash $ pip install --only-binary :all: fastapi[all]
```

However on some systems the `[all]` extensions may not yet be supported. Let us know how to install it there and we add it to our description. We for example found issues in a recent update to fastAPI that had difficulties to be installed on M1 MacOS.

4.2.7.2 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:

```
{"processor": "5950X"}
```

One of the embedded features of FastAPI is its build in documentation framework based on OpenAPI schema, but it also has alternative formats such as json or 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

In case you like to have a different name than `app` you can just use a different variable

```
from fastapi import FastAPI

my_awesome_app = FastAPI()

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

And copy it in a file `main.py` then you would call `uvicorn` like:

```
$ uvicorn main:my_awesome_app --reload
```

4.2.7.3 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}
```

4.2.7.3.1 Path Arguments Not only can you call functions through the path, you can also pass arguments through the path. The arguments can be specified by type in the formal parameter. In this case, we specify that it should be an int. This helps FastAPI validate data, which is done through the package Pydantic.

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

4.2.7.3.2 Query and Search Parameters When you declare other function parameters that are not part of the path parameters, they are automatically interpreted as URL "query" parameters, as seen in the function `get_job()`.

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

In addition, this can be used to search in the FastAPI, as seen in `search_job()`. You can use the URL to search the `jobs` variables like this:

```
http://127.0.0.1:8000/job/?name=Job1
```

Output:

```
"Job1"
```

```
from fastapi import FastAPI

app = FastAPI()
jobs = [{"name": "Job1", "temperature": 0},
        {"name": "Job2", "temperature": 30},
        {"name": "Job3", "temperature": 10}]

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

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

4.2.7.4 Running Through Git bash

```
import requests

result = requests.get('http://127.0.0.1:8000/job/?name=Job1')

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, yielding the output:

```
"Job1"
200
application/json
utf-8
"Job1"
Job1
```

4.2.7.5 Integration with pedantic

Assignment:

- Use pedantic to create a computer that contains temperature and load and implement it
- Use a regular python class to define dynamically access to temperature and proceessor load and contrast the implementation with the one where you use pedantic

4.2.7.6 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
```

4.2.7.7 Running through Docker In most cases, it is preferable to use a container to run the REST service, since there are many dependencies and packages. Here we will show you how to create and export an image for your FastAPI service to Docker.

To start, create a project directory. We will call it `project`. In this folder create a subdirectory called `app`. This can be done through Git Bash.

```
$ mkdir project
$ cd project
$ mkdir app
```

In the `project` directory, create two files `requirements.txt` and `Dockerfile`. `requirements.txt` will have the necessary package dependencies. Depending on what packages are used, this will vary. This is an example of what it would look like.

```
fastapi>=0.68.0,<0.69.0
pydantic>=1.8.0,<2.0.0
uvicorn>=0.15.0,<0.16.0
```

In `Dockerfile`, copy this code:

```
#
FROM python:3.9

#
WORKDIR /code

#
COPY ./requirements.txt /code/requirements.txt

#
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

#
COPY ./app /code/app

#
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "80"]
```

In your `app` folder, add your FastAPI code. In this tutorial we have used the name `main.py`. In addition, create a python file `__init__.py`. This can be left empty.

To build the Docker image, go back to the project directory and type the following code:

```
$ docker build -t myimage .
```

To create a running instance of this image, execute the following code:

```
$ docker run -d --name mycontainer -p 80:80 myimage
```

4.2.7.8 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.

4.2.7.8.1 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.aniyo` 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.

4.2.7.9 Links

- Fastapi First Step[23]
- Path Parameters[24]
- Docker Deployment[25]
- Using Test Client[26]
- Async Tests[27]

4.2.8 Queue

A queue is list that has an order.

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).

4.2.8.1 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 comes out of the data structure first.

FIFO queues are easy to implement:

```
import queue

# 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

4.2.8.2 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:

```
import queue

# this queue is a LIFO queue

lq = queue.LifoQueue()

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

This code can be accesed on GitHub

4.2.8.3 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:

- Python Queues [28]
- Priority Queues in Python[28]
- Queue Documentations[29]

4.3 CHARTS

Charts	Matlotlib	Seaborn	Bokeh	Pandas
barchart	bar	barplot	vbar	plot.bar
grouped barchart	bar	catplot	-	plot.bar
stacked barchart	bar	-	vbar_stack	plot.bar(..)
spline chart	-	-	-	-
multiline chart	plot	lineplot	multi_line	plot.line
compound line chart	-	lineplot	-	plot.area
histogram	hist	histplot	-	plot.hist
linechart	plot	lineplot	line	plot.line
piechart	pie	-	wedge	plot.pie
exploded piechart	pie(...)	-	-	-
donutchart	pie(...)	-	Donut	-
countourplot	contour	kdeplot	-	-
distributionplot	hist	displot	Histogram	-
point chart	-	pointplt	-	plot.scatter
scatterplot	scatter	scatterplot	Scatter	plot.scatter
bubblechart	scatter	-	-	plot.scatter(..)
radar chart	plot	-	-	-
boxplot	boxplot	boxplot	Boxplot	plot.boxplot
pdf export	+	matplotlib	-	matplotlib
png export	+	matplotlib	export_png	matplotlib
svg export	+	matplotlib	export_svg	matplotlib
color palettes	-	color_palette	Color Palettes	-
interactive graph	-	-	+	-

4.3.1 Python Graphics Introduction

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 identfy 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.

4.3.2 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.

4.3.2.1 Installation

To install matplotlib, please use the command:

```
$ pip install matplotlib
```

4.3.2.2 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
```

4.3.2.3 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 5.

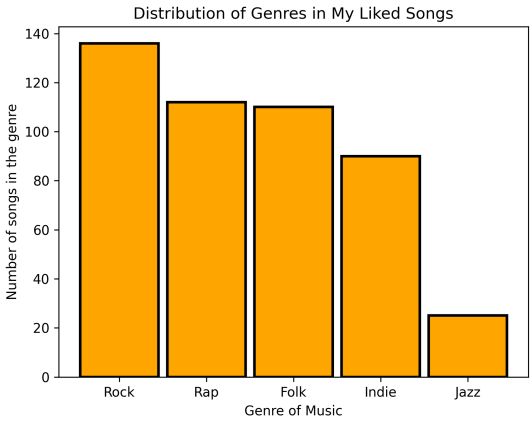


Figure 5: Matplotlib Bar Chart (data from Spotify)

4.3.2.4 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 6.

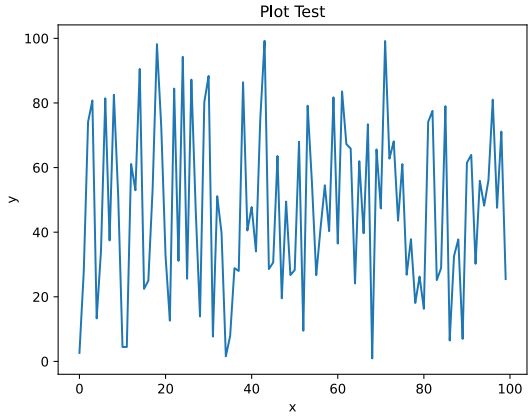


Figure 6: Matplotlib Line Plot (Random Data)

4.3.2.5 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 7.

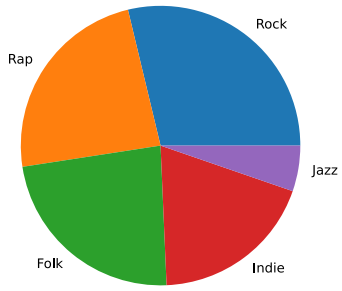


Figure 7: Matplotlib Piecahrt (Genres of Music from Spotify).

4.3.2.6 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 8.

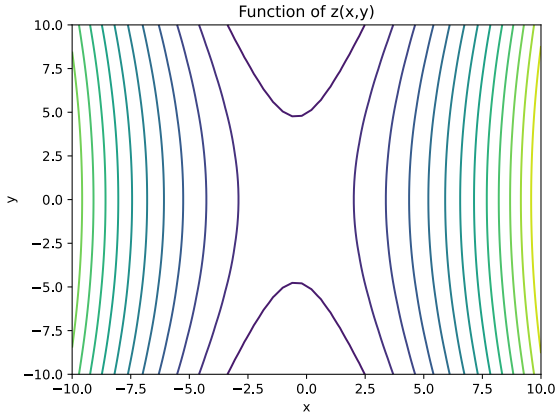


Figure 8: Matplotlib Multivariable Math Equation Shown on a Contour Plot.

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.

4.3.2.7 Annotations

4.3.2.7.1 Titles Titles enable to add a title to the graph. NOte that in most academic publications titles are not used and instead the information is added to a separate caption for the figure. Nevertheless, to add a title to your whole graph in matplotlib, simply type:

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

4.3.2.7.2 XY Labels To se an X and Y Labeel which are needed foa all publisations you can set them with

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

4.3.2.7.3 Legend In case you have multiple graohs you will also want to provide some information about them in a legend:

```
plt.legend()
```

4.3.2.7.4 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`.

```
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()
```

This program can be downloaded from GitHub.

The output of this program is showcased in Figure 9.

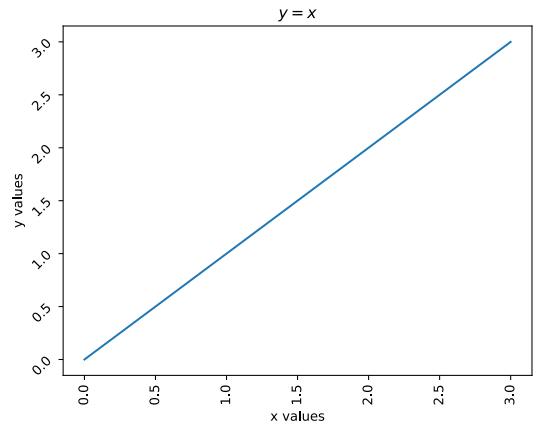


Figure 9: Matplotlib Line Plot with `x-axis` Ticks Rotated by 90 degrees and `y-axis` Ticks Rotated by 45 degrees

4.3.2.8 Export

Saving Chart as files is easy with the command:

```
plt.savefig("fname", dpi=300)
```

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`. Please note that we always want o print the png, SVG, and PDF as dependent on publisher one or the other is preferred. The best way to deal with this is to create a custom save plot function such as

```
import matplotlib.pyplot as plt

def save_plt(name):
  plt.savefig(f'{name}.png', dpi=300)
  plt.savefig(f'{name}.pdf')
  plt.savefig(f'{name}.svg')
  plt.show()
```

This code can be accessed on GitHub

4.3.2.8.1 Display

The `plt.show()` displays the graph on screen:

```
plt.show()
```

4.3.2.9 Links

- https://matplotlib.org/ [30]
- https://matplotlib.org/stable/api/pyplot_summary.html [31]
- https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/ [32]
- https://www.geeksforgeeks.org/bar-plot-in-matplotlib/ [33]

4.3.3 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.

4.3.3.1 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
```

4.3.3.2 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]
```

4.3.3.3 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 10.

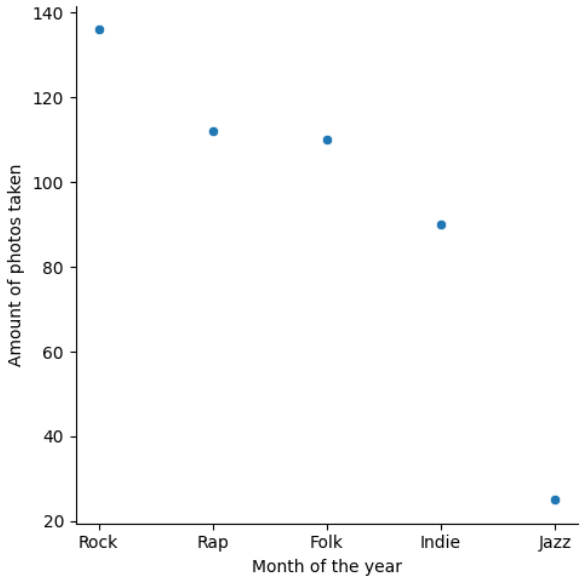


Figure 10: Seaborn Line Plot Created from User Spotify Data

4.3.3.4 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 11.

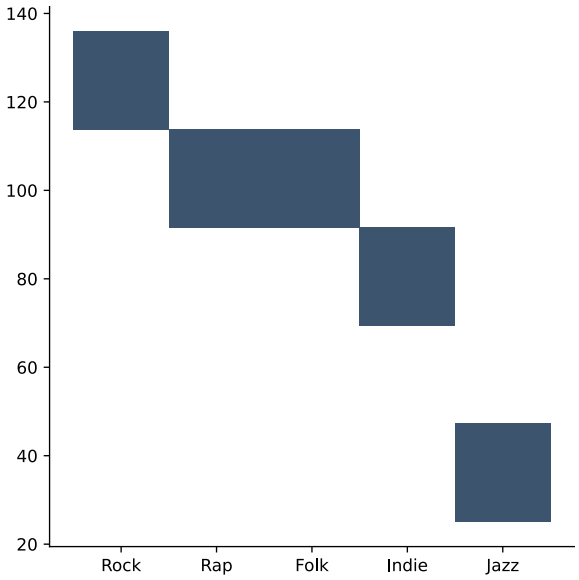


Figure 11: Seaborn Distribution Plot Created from User Spotify Data

4.3.3.5 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 12.

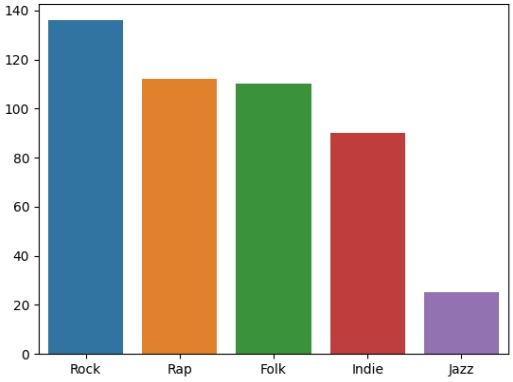


Figure 12: Seaborn Categorical Plot Created from User Spotify Data

4.3.3.6 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 13.

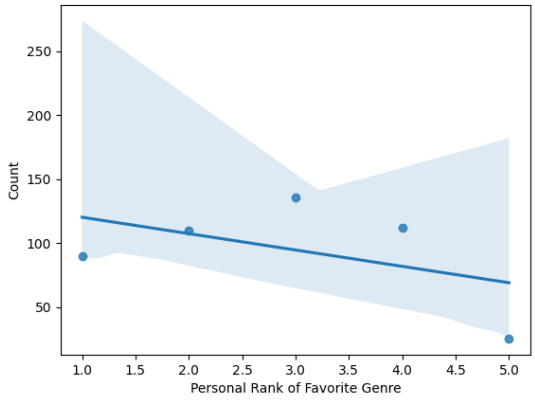


Figure 13: Seaborn Regression Plot Created from User Spotify Data

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

4.3.3.7 Saving Figures

Saving figures created by Seaborn is quite simple as it uses matplotlib.

```
plt.savefig('figure_path/figure_name.png', dpi=300)
```

See the matplotlib section fro a save function saving in svg, pdf, and png.

This program can be downloaded from GitHub

We have included in the matplotlib section a function example that saves to png, svg, and PDF which is often needed for publications.

4.3.3.8 Links

- https://seaborn.pydata.org/api.html [34]
- https://www.geeksforgeeks.org/python-seaborn-tutorial/ [35]
- https://www.geeksforgeeks.org/introduction-to-seaborn-python/ [36]
- https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/ [37]
- https://stackoverflow.com/questions/30336324/seaborn-load-dataset [38]
- https://github.com/mwaskom/seaborn-data/blob/master/planets.csv [39]

4.3.4 Bokeh

Bokeh [40] 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.

4.3.4.1 Installation

To install Bokeh, please use the command:

```
$ pip install bokeh
```

4.3.4.2 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
```

4.3.4.3 Bokeh Plotting `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 14.

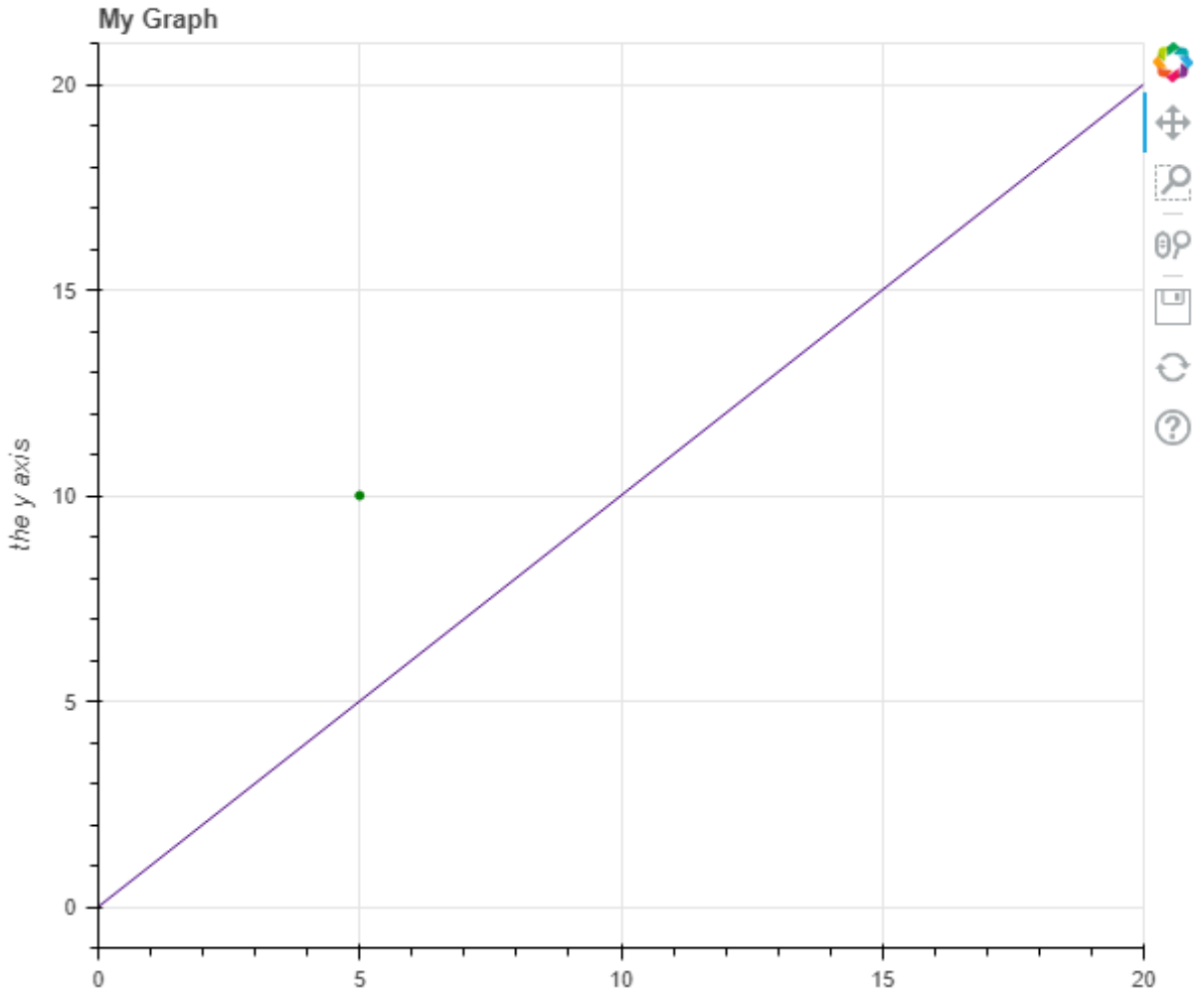


Figure 14: Bokeh Line and Point Plotted on a Chart

4.3.4.4 Annotations The titel can be set with **title**. Bokeh allows setting annotations with the paarmeters **x_axis_label** and **y_axis_label**. To set the range of the s and y axis you can use **x_range** and **y_range**.

4.3.4.5 Dimensions and Color The dimension of the plot can be set with **height** and **width**. The background color can be changed with **background_fill_color**, however, we always want to set it to white for publications.

4.3.4.6 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 15.

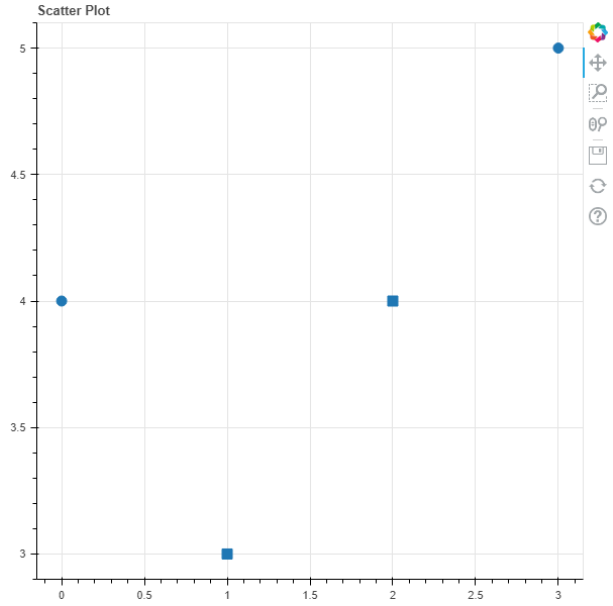


Figure 15: Bokeh Scatter Plot with Various Point Shapes

The list possible marker types and the functions used to create them can be found here.

4.3.4.7 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 The output is shown in Figure 16.

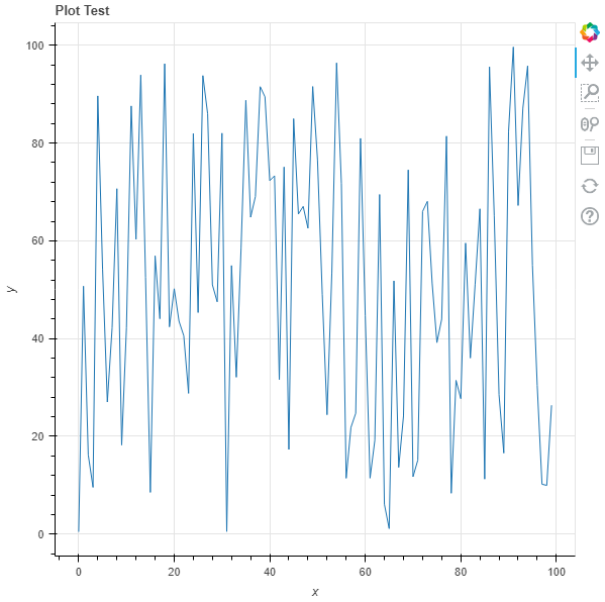


Figure 16: Bokeh Line Plot with random data.

You can find the source code for other types of line plots here: http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html

4.3.4.8 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 17.

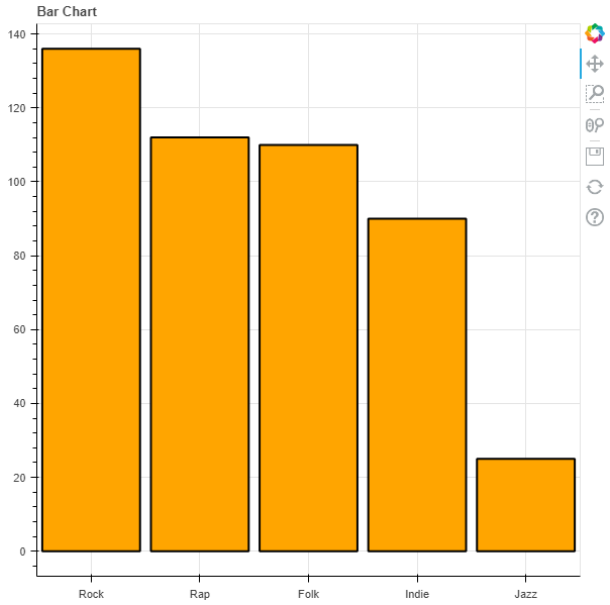


Figure 17: Bokeh Bar CHart with Spotify Data.

4.3.4.9 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.

4.3.4.9.1 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")
```

Assignment:

- find out how to add a dpi to the png image. Set it to 300dpi.
- modify this document and the code accordingly

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.

4.3.4.10 Links

- Bokeh user guide for plotting [41]
- Latest Bokeh Information [42]
- Bokeh Documentation [43]
- Exporting Plots in Bokeh [44]

4.3.5 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.

4.3.5.1 Installation To install Pandas, please use the command:

```
$ pip install pandas
```

4.3.5.2 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
```

4.3.5.3 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 18.

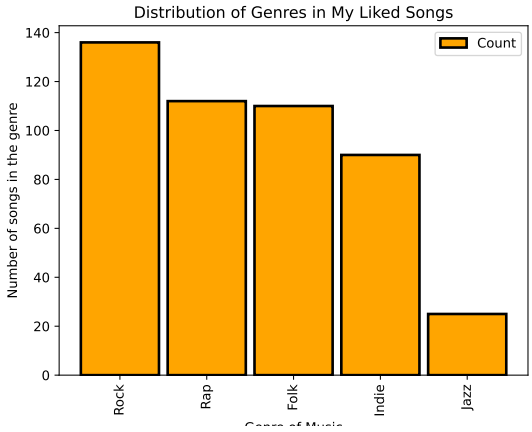


Figure 18: Pandas Bar Chart of Spotify Data

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.

4.3.5.4 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 19.

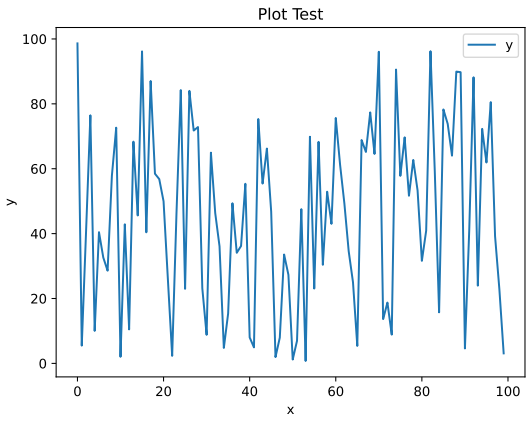


Figure 19: Pandas Line Plot

4.3.5.5 Pie Chart A pie chart is useful for showing a divison 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 20.

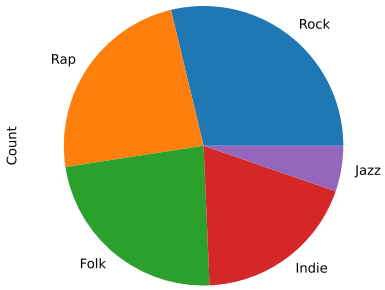


Figure 20: Pandas Pie Chart from Spotify Data

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()`.

4.3.5.6 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()
```

4.3.5.7 Links

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html [45]
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html [46]
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html [47]

4.4 GRAPHS WITH PYTHON

4.4.1 Networkx

`networkx` is a python library that allows a user to create graphs. The library allows the user to create different kinds of graphs, included directed and undirected. With this in mind, the following is a basic overview of the library.

4.4.1.1 Installing and Importing Thankfully, `networkx` can be super easily downloaded. Simply execute the following:

```
$ pip install networkx
```

4.4.1.2 Creating a graph It is super simple to create a graph, add nodes to the graph, and add edges to the graph. For the purposes of this tutorial, we will be working with a synthetic job queueing service. Each node represents a job and the edges are the edges that connect each job.

The following is the code that was created:

```
import networkx as nx

# creating a list of nodes and edges- you have to be extremely careful not to typo
nodes = ['job-1', 'job-2', 'job-3','job-4', 'job-5']

edges = [('job-1', 'job-2'), ('job-2', 'job-3'), ('job-3', 'job-4'),
         ('job-4', 'job-5')]
```

This code can be accessed from GitHub

To create the graph and add the above nodes and edges, `networkx` provides super simple methods.

```
1 import networkx as nx
2
3 graph = nx.Graph()  # creates the graph that we will use
4
5 graph.add_nodes_from(nodes)  # adds the nodes from the dict we created
6 graph.add_edges_from(edges)  # adds the edges from the dict we created
7
8 print(graph)  # prints: "Graph with 5 nodes and 4 edges"
```

This code can be accessed from GitHub

Thus, we have created the graph.

4.4.1.3 Accessing It is necessary to be able to access the elements within the graph. There is a slightly strange way of going about doing this with `networkx`. Rather than being able to directly access the element you input, it will return the adjacent element. Thus, it seems to be necessary to have a dummy head, which makes it so that the first node inserted into the graph can be accessed.

The following looks into this:

```
9 print(graph['job-1'])  # will return {'job-2' : {}}
```

This code can be accessed from GitHub

4.4.1.4 Removing nodes It is very easy to remove nodes and edges using `networkx`. This can be accomplished by executing the following:

```
import networkx as nx

G = nx.Graph()

# creating a list of nodes and edges- you have to be extremely careful not to typo
nodes = ['job-1', 'job-2', 'job-3', 'job-4', 'job-5']

edges = [('job-1', 'job-2'), ('job-2', 'job-3'), ('job-3', 'job-4'),
         ('job-4', 'job-5')]

print(G)  # will return "Graph with 5 nodes and 4 edges".

G.remove('job-1')

print(G)  # will return "Graph with 4 nodes and 3 edges"
```

This code can be accessed on GitHub

4.4.1.5 Colors and Labels It is super simple to add colors and labels, as well as display the networks in a graphic notation. To do so, execute the following code:

```
color_map = []
for n in nodes:
    color_map.append('lightblue')

nx.draw(G, node_color=color_map, with_labels=True)
plt.show()
```

This code can be accessed on GitHub

The `color_map` was created as a list based on the number of nodes in our list of nodes we created before. Thus, it makes it very easy to implement as it allows the user to update individual nodes, if needed.

The code produces the following image:

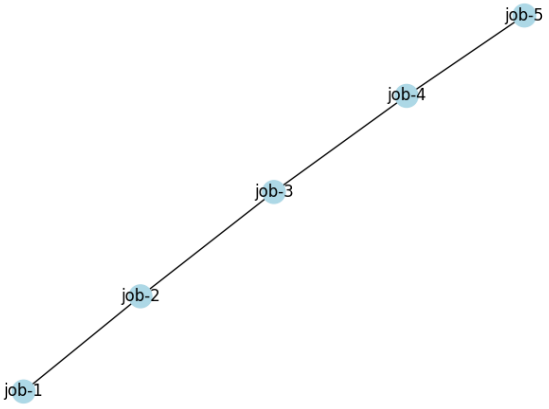


Figure 21: network-image created with networkx python package

There are many other ways to manipulate the nodes and edges. These ways can be accessed from the official documentation

4.4.2 Graphviz

`graphviz` is a library that can be installed on Python which renders graphs from the DOT languages, allowing for visualizations of data structures using undirected or directed graphs consisting of nodes and edges of various shapes and colors.

4.4.2.1 Installation and Importing In order to install `graphviz` on Windows fully, first, have Chocolatey installed. Next, run Command Prompt as an administrator and type in the following:

```
choco install graphviz
```

Next, go onto GitBash and type in the following:

```
$ pip install graphviz
```

4.4.2.2 Creating the graph, adding nodes, and adding edges Creating a basic graph with nodes and edges is very simple using `graphviz`. The following example is a synthetic job queueing service. Each node represents a job and the edges connect the jobs.

There are two types that can be made using `graphviz`. Regular graphs can be made using `Graph()`. They don't have arrows. Directed graphs can be made using `Digraph()`. They do have arrows.

Nodes can be created using `node()` where the variable of the job can be defined and labeled.

Edges can be created either using `edge()` or `edges()`, as used in this example. The commands take in the start and end node variables which will create either one or multiple edges, respectively. Edges can be labeled too.

The following is the code that was created:

```
import graphviz

f = graphviz.Graph('jobs in queues', filename='examples/basic-graphviz.gv')

f.node('job-1', 'ls')
f.node('job-2', 'echo hello world')
f.node('job-3', 'cd ~')
f.node('job-4', 'cd cm/cloudmesh-cc')
f.node('job-5', 'pytest tests')
f.edges([('job-1', 'job-2'), ('job-2', 'job-3'), ('job-3', 'job-4'),
         ('job-4', 'job-5')], )

f.view()
```

This code can be accessed via Github.

Shown here in Figure 22 is the graph produced from the code.

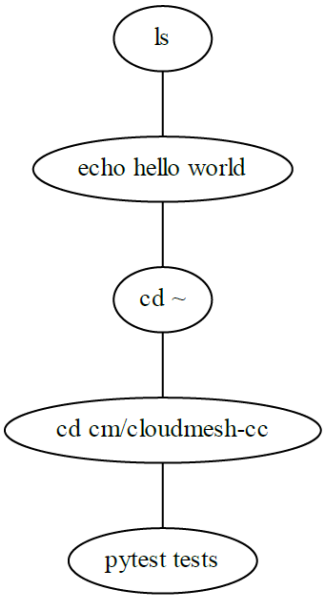


Figure 22: Sequence of Bash Command Path

4.4.2.3 Subgraphs Subgraphs are clusters of nodes and edges that can be created using the `subgraph()` command. However, it's required to have the prefix `'cluster'` in the name of it.

The following code shows the usage of subgraphs by expanding on the last example.

```
import graphviz

g = graphviz.Digraph('jobs in queues', filename='subgraph-graphviz.gv')

with g.subgraph(name='cluster_1') as s:
    s.node('job-1', 'ls')
    s.node('job-2', 'echo hello world')
    s.node('job-3', 'cd')
    s.node('job-4', 'cd cm/cloudmesh-cc')
    s.node('job-5', 'pytest tests')
    s.edges([('job-1', 'job-2'), ('job-2', 'job-3'), ('job-3', 'job-4'),
             ('job-4', 'job-5')], )

with g.subgraph(name='cluster_2') as s:
    s.node('job-6', 'cd')
    s.node('job-7', 'cd cm')
    s.node('job-8', 'cd cm/cloudmesh-alex')
    s.node('job-9', 'git status')
    s.node('job-10', 'git pull')
    s.edges([('job-6', 'job-7'), ('job-7', 'job-8'), ('job-8', 'job-9'),
             ('job-9', 'job-10')], )

g.edge('start', 'job-1')
g.edge('start', 'job-6')
g.edge('job-5', 'end')
g.edge('job-10', 'end')
g.edge('job-3', 'job-7')
g.edge('job-4', 'job-9')

g.node('start', shape='square')
g.node('end', shape='square')

g.view()
```

This code can be accessed via Github.

Shown here in Figure 23 are the subgraphs produced by the code.

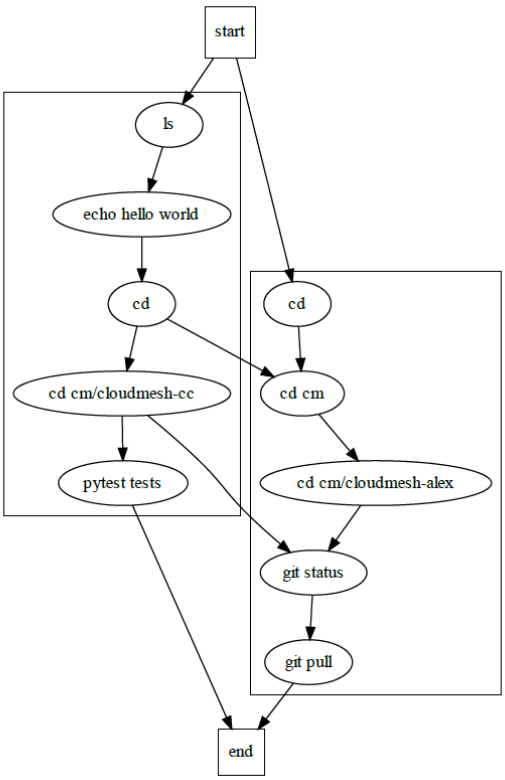


Figure 23: Sequence of Two Bash Different Command Paths

4.4.2.4 Data Structures Rectangular data structures can be created in `graphviz` when the shape of the nodes is set to `'record'`. This specific type of data structure allows for nodes to be clustered together in the same rectangle. The following code shows a diagram of different files and directories.

```
import graphviz

s = graphviz.Digraph('files in directories', filename='structure-graphviz.gv')
s.node_attr={'shape' : 'record'}

s.node('s1', '<d1> cloudmesh-cc | <d2> workflow')
s.node('s2', '{<d1> cloudmesh | <d2> tests}')
s.node('s3', '<d1> contribute | {<d2> graphs |{<d3> graphviz | <d4> networkx}}')

s.edges([('s1:d1', 's2:d2'), ('s1:d2', 's3:d4')])

s.view()
```

This code can be accessed via Github.

Shown here in Figure 24 are the data structures produced by the code.

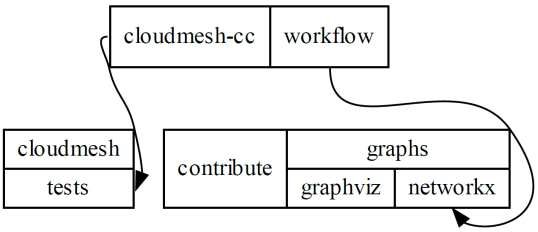


Figure 24: Data Structure of Various Directories and Files

4.4.2.5 Colors and Labels It is super simple to add colors and labels to graphs, nodes, and edges in `graphviz`.

In terms of labels, just add the parameter `label=''` inside the `attr()`, `node()`, or `edge()` commands. The font color of the label can be set using `fontcolor=`.

In terms of color, just add the parameter `color=''` inside the `attr()` or `node()` commands. This will change the color of the perimeter. In order to fill, use the parameter `style='filled'` or set the fill color using `fillcolor=''`.

Gradients can also be added by setting a colon `:` between two different colors. Furthermore, the angle of the gradient can be set using the parameter `gradientangle=''`.

The following code demonstrates many of the features explained.

```
import graphviz

g = graphviz.Digraph('Colors', filename='color-graphviz.gv')
g.attr(bgcolor='red:pink', label='Red Graph', fontcolor='white')

with g.subgraph(name='cluster') as c:
    c.attr(color='cyan', style='filled', label='Cyan Cluster',
           fontcolor='white')
    c.node('n1', 'Orange Node', shape='circle', fillcolor='red:yellow',
           style='filled', gradientangle='90')
    c.node('n2', 'Yellow Node', shape='diamond', color='yellow', style='filled')
    c.edge('n2', 'n1', label='Edge 1')

g.view()
```

This code can be accessed via GitHub.

Shown here in Figure 25 is what the code produced.

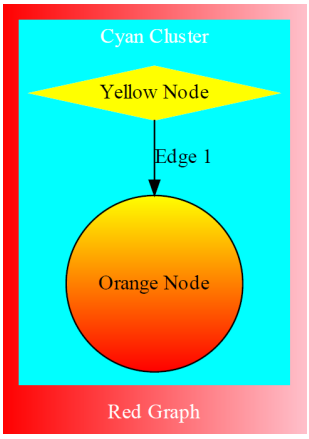


Figure 25: Label Directed Graph with Various Colored Nodes

4.4.2.6 Links

- Graphviz Website
- Graphviz Example
- Graphviz API Reference

4.5 SELECTED CLOUDMESH TOPICS

4.5.1 cms sys command generate

Learning Objectives

- Learn how to create your own cloudmesh commands

Assignment:

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.

4.5.1.1 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 `docopts` 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.

4.5.1.1.1 Example of the CLI Creation

Here is an example command:

```
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.

4.5.1.1.2 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

4.5.2 Cloudmesh StopWatch

Learning Objectives

- Using timers in python
- Using cloudmesh.common.StopWatch

- Create benchmarkes from StopWatch

Improve the documentation of Cloudmesh StopWatch if needed

StopWatch Documentation

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.

4.5.3 How to Set Up Browser Defaults on Windows

4.5.3.1 Changing Default Browser

1. Press the Windows key and type `Default apps`, then press `Enter`.
2. Change `Web browser` to your desired browser.

4.5.3.2 Changing Default Search Engine Changing your search engine depends on the browser you are using. If you are using a fresh install of Windows, then the default browser is Microsoft Edge.

4.5.3.2.1 Microsoft Edge Edge's default search engine is Bing.

1. In Microsoft Edge, click the three dots in the top-right corner and then click `Settings` at the bottom.
2. On the left-hand side, click `Privacy, search, and services`. Scroll all the way to the bottom and click `Address bar and search`.
3. Lastly, change `Search engine used in the address bar` from Bing to Google (or whichever engine you desire).

4.5.3.2.2 Google Chrome Chrome's default search engine is Google.

1. In Chrome, click the three dots in the top-right corner and then click on `Settings`.
2. On the left-hand side, click `Search engine`.
3. Change `Google` to your desired search engine.

4.5.3.2.3 Firefox Firefox's default search engine is Google.

1. In Firefox, click the horizontal three lines (hamburger menu) in the top-right corner.
2. Click on `Search` on the left-hand side.
3. Change `Default Search Engine` from Google to your desired search engine.

4.5.3.2.4 Opera Opera's default search engine is Google.

1. In Opera, click the horizontal three lines with sliders (hamburger menu).
2. Scroll down and click on `Go to full browser settings`.
3. Scroll down to `Search engine` and change `Google Search` to your desired search engine.

4.5.4 CLOUDMESH WORKFLOW

4.5.4.1 Documentation

see cloudmesh.cmd5

- Cloudmesh-CMD File

4.5.4.2 Workflow The workflow class is a class in which users can upload files of jobs (bash and sbatch scripts) and then these jobs are loaded, saved, and run (and in doing so, images are produced of the workflow progression).

This class was implemented with the intention of allowing users to upload jobs that ranged in their complexity and necessary time to complete. Additionally, the python script was created with the intention that jobs could be run on various different operating systems (mac, Windows, and linux) and on different computers, depending on the configuration that the user specified.

For instance, if a user had four jobs that they wanted to run, but they wanted a few of these jobs to be run on a remote computer (like UVa's Rivanna) and a few to be run locally, they could split these tasks up and run them like that. They can run these jobs in parallel (meaning that the remote and local jobs are run at the same time) or they can run these in a topological sort fashion (which means that the workflow will wait for the preceding job to finish before the next job is run).

In this documentation, we will walk through the set up, methods, and use cases for this workflow and will mention the `fastAPI` service that accompanies this as well.

4.5.4.2.1 Set-Up Cloning the repository

In order to be able to accurately access all of the componenets of the workflow (specifically for using the python script for programming), it is necessary to navigate to GitHub and clone the respository so that you can incorporate it into your own python scripts. The command looks like this:

```
git clone <insert the git clone ssh link>
```

In doing so, you can pull the whole cloudmesh-cc repository into your own code development.

Using the repository in your own code

Then, in you python script you would simply do the following:

```
from cloudmesh.cc.workflow import workflow

w = workflow(name='workflow', filename='pathtoyamlyouhave')
```

Yaml files

As you can see in the previous script, the workflow has a parameter called `filename`. This parameter is used to load the data that exists in a file into the overal data structure that exists in the class. Essentially, this parameter is used to bring all of your data into the workflow so that it can be manipulated and changed and used.

There are two options that exist for the instantiation of the workflow. One is that you can pre-build a script that the workflow can use to load in a bunch of data (hence the `filename` parameter from before). The other way is that you can manually add jobs (which will be discussed later on). First, let's discuss the set-up of a proper yaml file for the `workflow` to load in.

The yaml file should set up several parameters for the `workflow` to load in. Rather than explain, we will show an example.

Example

```
workflow:
  nodes:
    start:
      label: start
      kind: local
      user: jacksonmiskill
      host: local
```

```
      status: ready
      progress: 0
      created: '2022-07-20 13:35:35.600053'
      modified: '2022-07-20 13:35:35.600053'
      script: null
      instance: null
      name: start
      parent: []
    end:
      label: end
      kind: local
      user: jacksonmiskill
      host: local
      status: ready
      progress: 0
      created: '2022-07-20 13:35:35.601217'
      modified: '2022-07-20 13:35:35.601217'
      script: null
      instance: null
      name: end
      parent:
      - job-local-2
```

Many of the methods within the `workflow` will update the yaml file with values, like the `created` and `modified` fields. There are several fields that are necessary to fill out before running, however, for the optimal functionality.

First: the name parameter is where the `start` and `end` are above their fields. It is necessary to fill this out with what you would like to run. This also used in order to make sure that the data structure works as intended.

`kind` it is necessary to fill out this field as well. This should specify where you want the job to be run- it gives the `workflow` a flag to look for in deciding which type of job to run.

`name` is it also necessary to go ahead and write in the name of the job that you are attempting to instantiate.

So, a basic script might look like this:

```
workflow:
  nodes:
    a:
      kind: local
      name: a
    b:
      kind: local
```

```
      name: b
```

This would be the most basic way to set this up.

4.5.4.2.2 Methods The most important methods for this class are `jobs`, `load`, `add_job`, `add_dependencies`, `run_parallel`, `run_topo`, `remove_workflow`, `remove_job`, and `status`.

These methods can be found on GitHub and are, for the most part, self-explantory.

Adding new jobs and dependencies

One thing to point out here, however, is that you can manually add jobs and dependencies to the `workflow`, rather than only through the `yaml` file through the `add_job` and `add_dependencies` methods.

To execute this in `python`, simply execute the following

```
from cloudmesh.cc.workflow import Workflow

w = Workflow(name='workflow', filename='pathtoyaml')

w.add_job(name='job-x', label='job-x', kind='local', user='username', status='ready'
    ↪ )

w.add_job(name='job-y', label='job-y', kind='local', user='username', status='ready'
    ↪ )

w.add_dependencies(dependency='x-y')
```

Running jobs

The `run_parallel` and `run_topo` methods are used in order to run the jobs that have been uploaded to the workflow data structure.The way that this works is simple. There are implementations of different kinds of the `job` object. This means that there is a `job` for `slurm`, for `local`, and for `ssh`. When the run method is called, it creates the correct type of job based on the kind of job as specified in the yaml file or in the `add_job` method.

The `run_parallel` method runs the jobs side-by-side, if there are different kinds that are distinguished throughout the jobs. For instance, if there are two kinds of jobs in the workflow- one as `ssh` and one as `local`, then the `run_parallel` method will run them separately and bring them together.

The `run_topo` method creates and runs jobs based on the topological sorting that the sorting method does. This method essentially creates a sort, and will wait for the preceding job to finish before the next job is run.

Removing Jobs

There are `remove_job` and `remove_workflow` methods that can be used to get rid of a single job, or the entirety of a workflow.

In order to utilize `remove_job` is is necessary to specify the name of the job in the parameters.

In order to utilize the `remove_workflow` is is necessary to specify the name of the workflow and then it should function as specified.

4.5.4.3 Queue Data Structure

The queue data structure is a data structure that was built for cloudmesh, a cloud computing platform around which this documentation is based. The queue module is not a queue like the built-in python module. The built-in python module lines up the data in the correct order and implements the push and pop methods correctly.

The queue structure in the cloudmesh GitHub is a data structure that creates jobs, lines up the jobs in a queue, then places that queue into a larger queues data structure. This makes it so that a user could add jobs to a queue, then add that queue to a list of queues and then run those queues.

This module was built with the intent of allowing researchers to run jobs (with are their experimental tests) on multiple devices and report those jobs back.

4.5.4.3.1 Location

The code for this project is located on GitHub. Please feel free to clone and download and use for your own purposes.

4.5.4.3.2 Installation

To install this module, simply download the GitHub repository and utilize the correct import statement that you need in order to make this work in the directory you are working in.

4.5.4.3.3 Contents

In the queue module there are three different classes.

`Job`

The `Job` class is an object that creates the job. There are a few things that the objects holds.

Fields:

It has the fields: name, user, label, command, status, output, output_file, error_file, error, kind, progress, created, and modified. Each of these fields represents necessary information about the job, especially the command. If there is a `.sh` file that you would like to run as opposed to a single command, the command could be to run that `.sh` file!

Methods:

In the `job` module, there are a few methods to assist in leveraging the power of the queue. These methods are: `str`, `dict`, `set`, `update`, `run`, and `get_progress`. The implementations of these methods can be found on GitHub. Please review to understand how to utilize these for your program!

Queue

The `Queue` class is a class that allows users to add jobs and then puts those jobs into a dictionary structure with the name of the job followed by the job itself.

Queues

4.5.4.4 Databases Program

The database program is a selection of different implementations that allow a user to save objects and variables in specific files through which they can access these files and utilize them for the proper use cases.

There are two different database examples that have been created: yamldb and shelve.

4.5.4.4.1 Location

These programs are located in cloudmesh-cc, which is available on GitHub. You can download and utilize these programs to your liking.

4.5.4.4.2 Contents

Inside each of the database programs are many functions that make it easy for a user to utilize the program to their needs.

In yamldb: `__init__`, `save`, `load`, `remove`, `get`, `get_queue`, `get_job`, `__setitem__`, `__str__`, `clear`, `queues`, and `info`.

In shelve: `__init__`, `queues`, `filename`, `shelvename`, `info`, `load`, `save`, `close`, `remove`, `get`, `getitem`, `set`, `__setitem__`, `__str__`, `delete`, `__delitem__`, `clear`, and `len`

4.5.4.4.3 Initialization

The typical use case for these databases is to add it to another object class so that that class can be saved to a database and accessed for testing and for scripting implementation.

In order to do so, the database should be introduced in the initialization of the object:

```
from cloudmesh.cc.db.yamldb.database import Database as QueueDB
# from cloudmesh.cc.db.shelve.database import Database as QueueDB

class Queues:
    def __init__(self, filename=None):
        """
        Initializes the queue structure.
        """
```

```
        if filename is None:
            filename = "~/.cloudmesh/queue/queue.yamldb"

        self.db = QueueDB(filename=filename)
        self.counter = 0
```

This code can be access on GitHub

So now the database can be called from within the object as well as outside the object (i.e. in other programs where a `Queues` object has been created)!.

4.5.4.4.4 Usage It is super simple to use the databases. They are implemented in a way so as to maximize user ease and capability. First of all, it is necessary to use the above methods for each of the two databases that were implemented. These methods allow for ease of access. In the `Queues` class that was implemented, there are a few methods that draw upon this database. We will explore them.

```
    def save(self):
        """
        save the queue to persistent storage
        """
        self.db.save()

    def load(self):
        self.db.load()
```

This code can be accessed on GitHub.

As you can see, the save and load methods have been implemented. After the instantiation of the database in the previous code block, the user can call the database using `self.db` to access it. Loading the database simply pulls the data from the file and loads it in a dictionary-like format. Saving the database allows you to push changes that you have made back into the database.

```
    def info(self):
        return self.db.info()
```

This code can be accessed on GitHub.

This command simply accessed the information provided by the database (ie. what is being stored in the database).

```
@property
    def queues(self):
        return self.db.queues
```

```
    @property
    def config(self):
        return self.db.data['config']
```

This code can be accessed on GitHub.

This code showcases a special way that the database can be used. In this queues class, we have a way to access specific objects that we have stored into the database. In the database, we have set a characteristic to be `queues`, which holds a dictionary of queue names to queues of jobs. In creating this special method, we allow programmers to very quickly and easily grab the queues that they need. This could be extrapolated, too. There could be another method that looks for only a `queue`. It could be implemented like so:

```
    def queue(self, name):
        """
        return the whole queue that the name in the parameter specifies
        """

        queues = self.queues()
        queue = queues[name]
        return queue
```

This code is not accessible anywhere, as it was written as a (untested) pseudo-code specifically for the discussion in this section of the documentation.

As you can see, this would allow a programmer not only access to the whole queues structure, but it would also allow access to a specific queue!

4.5.4.4.5 Extension These are not the only types of files that can be used to save data. There could be other modules out there that exist for the purpose of implementation in another fashion.

4.5.4.5 Workflow The workflow class is a class in which users can upload files of jobs (bash and sbatch scripts) and then these jobs are loaded, saved, and run (and in doing so, images are produced of the workflow progression).

This class was implemented with the intention of allowing users to upload jobs that ranged in their complexity and necessary time to complete. Additionally, the python script was created with the intention that jobs could be run on various different operating systems (mac, Windows, and linux) and on different computers, depending on the configuration that the user specified.

For instance, if a user had four jobs that they wanted to run, but they wanted a few of these jobs to be run on a remote computer (like UVa's Rivanna) and a few to be run locally, they could split these tasks

up and run them like that. They can run these jobs in parallel (meaning that the remote and local jobs are run at the same time) or they can run these in a topological sort fashion (which means that the workflow will wait for the preceding job to finish before the next job is run).

In this documentation, we will walk through the set up, methods, and use cases for this workflow and will mention the `fastAPI` service that accompanies this as well.

4.5.4.5.1 Set-Up Cloning the repository

In order to be able to accurately access all of the componenets of the workflow (specifically for using the python script for programming), it is necessary to navigate to GitHub and clone the respository so that you can incorporate it into your own python scripts. The command looks like this:

```
git clone <insert the git clone ssh link>
```

In doing so, you can pull the whole cloudmesh-cc repository into your own code development.

Using the repository in your own code

Then, in you python script you would simply do the following:

```
from cloudmesh.cc.workflow import workflow

w = workflow(name='workflow', filename='pathtoyamlyouhave')
```

Yaml files

As you can see in the previous script, the workflow has a parameter called `filename`. This parameter is used to load the data that exists in a file into the overal data structure that exists in the class. Essentially, this parameter is used to bring all of your data into the workflow so that it can be manipulated and changed and used.

There are two options that exist for the instantiation of the workflow. One is that you can pre-build a script that the workflow can use to load in a bunch of data (hence the `filename` parameter from before). The other way is that you can manually add jobs (which will be discussed later on). First, let's discuss the set-up of a proper yaml file for the `workflow` to load in.

The yaml file should set up several parameters for the `workflow` to load in. Rather than explain, we will show an example.

Example

```
workflow:
  nodes:
    start:
```

```
      label: start
      kind: local
      user: jacksonmiskill
      host: local
      status: ready
      progress: 0
      created: '2022-07-20 13:35:35.600053'
      modified: '2022-07-20 13:35:35.600053'
      script: null
      instance: null
      name: start
      parent: []
    end:
      label: end
      kind: local
      user: jacksonmiskill
      host: local
      status: ready
      progress: 0
      created: '2022-07-20 13:35:35.601217'
      modified: '2022-07-20 13:35:35.601217'
      script: null
      instance: null
      name: end
      parent:
      - job-local-2
```

Many of the methods within the `workflow` will update the yaml file with values, like the `created` and `modified` fields. There are several fields that are necessary to fill out before running, however, for the optimal functionality.

First: the name parameter is where the `start` and `end` are above their fields. It is necessary to fill this out with what you would like to run. This also used in order to make sure that the data structure works as intended.

`kind` it is necessary to fill out this field as well. This should specify where you want the job to be run- it gives the `workflow` a flag to look for in deciding which type of job to run.

`name` is it also necessary to go ahead and write in the name of the job that you are attempting to instantiate.

So, a basic script might look like this:

```
workflow:
  nodes:
    a:
```

```
      kind: local
      name: a
    b:
      kind: local
      name: b
```

This would be the most basic way to set this up.

4.5.4.5.2 Methods The most important methods for this class are `jobs`, `load`, `add_job`, `add_dependencies`, `run_parallel`, `run_topo`, `remove_workflow`, `remove_job`, and `status`.

These methods can be found on GitHub and are, for the most part, self-explantory.

Adding new jobs and dependencies

One thing to point out here, however, is that you can manually add jobs and dependencies to the `workflow`, rather than only through the `yaml` file through the `add_job` and `add_dependencies` methods.

To execute this in `python`, simply execute the following

```
from cloudmesh.cc.workflow import Workflow

w = Workflow(name='workflow', filename='pathtoyaml')

w.add_job(name='job-x', label='job-x', kind='local', user='username', status='ready'
    ↪ )

w.add_job(name='job-y', label='job-y', kind='local', user='username', status='ready'
    ↪ )

w.add_dependencies(dependency='x-y')
```

Running jobs

The `run_parallel` and `run_topo` methods are used in order to run the jobs that have been uploaded to the workflow data structure.The way that this works is simple. There are implementations of different kinds of the `job` object. This means that there is a `job` for `slurm`, for `local`, and for `ssh`. When the run method is called, it creates the correct type of job based on the kind of job as specified in the yaml file or in the `add_job` method.

The `run_parallel` method runs the jobs side-by-side, if there are different kinds that are distinguished throughout the jobs. For instance, if there are two kinds of jobs in the workflow- one as `ssh` and one as `local`, then the `run_parallel` method will run them separately and bring them together.

The `run_topo` method creates and runs jobs based on the topological sorting that the sorting method does. This method essentially creates a sort, and will wait for the preceding job to finish before the next job is run.

Removing Jobs

There are `remove_job` and `remove_workflow` methods that can be used to get rid of a single job, or the entirety of a workflow.

In order to utilize `remove_job` is is necessary to specify the name of the job in the parameters.

In order to utilize the `remove_workflow` is is necessary to specify the name of the workflow and then it should function as specified.

4.5.4.6 Running Tensorflow on Rivanna

4.5.4.6.1 Table of Contents

- Rivanna
- Running Tensorflow on Rivanna
- Run Python MPI Programs on Rivanna
- Setting Parameters while Running `mnist` Jobs

Learning Objectives

- Learn how to properly install Tensorflow on Rivanna using `conda`
- Learn how to load a singularity module of Tensorflow
- Learn how to run the GPU container image of Tensorflow through `sbatch`

4.5.4.6.2 Activating Rivanna Once you have Rivanna installed locally on your computer with your account set up, if you have Cloudmesh installed, you can simply connect to Rivanna on GitBash using the following commands. Make sure to run the terminal as Admin or else the first command won't work.

```
$ cms vpn connect
$ ssh rivanna
```

4.5.4.6.3 Installing Tensorflow Using Conda Generally in Rivanna, Python is run in a Conda environment. Because of that, Tensorflow can be installed using Conda using the command:

```
$rivanna conda install -n ENV3 tensorflow-gpu cudatoolkit
```

4.5.4.6.4 Loading and Running Singularity Alternatively, you can use Rivanna's built-in containers for running Tensor- flow called Singularity. You can load using the following commands:

```
rivanna$ module load singularity
rivanna$ module load tensorflow/2.8.0
```

This will give you this output which you can run:

```
To execute the default application inside the container, run:
singularity run --nv $CONTAINERDIR/tensorflow-2.8.0.sif
```

You should get instructions on how to now run this Tensorflow container. The `--nv` flag is to load the NVIDIA graphic driver for use of GPU.

4.5.4.6.5 Copying Tensorflow Container Image Before running Tensorflow on any file, you first want to copy the Tensorflow container image into your personal home directory. This can be done by typing in the following commands:

```
rivanna$ module load singularity
rivanna$ module load tensorflow/2.8.0
rivanna$ cd $CONTAINERDIR
rivanna$ cp tensorflow-2.8.0.sif $HOME
```

The first two commands loads you into the Singularity Tensorflow container. You are then directed to the default location of the container image and then you copy the image file into your home directory.

4.5.4.6.6 Running the Singularity Shell Before you run this shell, make sure to install cloudmesh in the image. Run the following commands:

```
rivanna$ singularity shell $HOME/tensorflow-2.8.0.sif
Singularity> pip install cloudmesh-common
```

4.5.4.6.7 Running Slurm Jobs Slurm jobs can be run through the Tensorflow container through `sbatch` scripts. Various arguments such as the job, output, GPU, time, account, etc. can be specified as shown in the example script below.

```
#!/bin/sh
#SBATCH --job-name=mydemojob
#SBATCH --output=%u-%j.out
#SBATCH --error=%u-%j.err
#SBATCH --partition=gpu
#SBATCH --gres=gpu:k80:1
#SBATCH --cpus-per-task=1
#SBATCH --mem=4GB
#SBATCH -t 00:01:00
#SBATCH -A bii_dsc_community

echo "# cloudmesh status=running progress=1 pid=$SLURM_JOB_ID"

module purge
module load singularity

containerdir=$HOME
singularity run --nv $containerdir/tensorflow-2.8.0.sif mydemojob.py

echo " cloudmesh status=done progress=100 pid=$SLURM_JOB_ID"

#
```

4.5.4.6.8 Examples from GitHub Here we demonstrate how to run various examples from our GitHub:

Log into Rivanna

```
$ ssh rivanna
$rivanna: git clone https://github.com/cybertraining-dsc/reu2022.git
$rivanna: cd reu2022/code/deeplearning
$rivanna: bash test1.sh
```

Some users may want to generate modified `ipynb`'s to generate their Python programs. In order to do so, you can simply say:

```
$rivanna: make all
```

Also, if you do not want to modify them, they are already included.

In case you would like to run an individual example, such as `mnist_autoencoder`, you can it in the following way:

```
$rivanna: cd cm/reu2022/code/deeplearning
$rivanna: sbatch --gres:gpu:k80:1 mnist_autoencoder.sh
```

In case you would like to do Papermill, which allows you to run a Python notebook directly without GUI, you can do it as follows:

```
$rivanna: pip install papermill
$rivanna: sbatch --gres:gpu:k80:1 mnist_autoencoder_papermill.sh
```

To see the results, all results are being prepended with a `# csv` string. You can `fgrep` the result as follows:

```
$rivanna: fgrep '# csv' mnist_autoencoder_output.py
```

4.5.4.6.9 Links

- Tensorflow[48]
- Rivanna Tensorflow[49]
- Rivanna Software Container[50]

5 INTRODUCTION TO DEEP LEARNING

5.1 AI

Learning Objectives

- Organizes the AI information

Missing modules, tutorials

- start jupyter on local computer

- start jupyter on rivanna
- graphs: mermaid: https://cybertraining-dsc.github.io/docs/modules/sample/mermaid/
- Update to https://cybertraining-dsc.github.io/docs/modules/mnist-reu/ See also the code in mlcommons

5.1.1 Analytics services

- https://cybertraining-dsc.github.io/docs/modules/cloudmesh-rest/

5.1.2 Reports

- Projects will be deterimined with the help of GRegor. THIs REU everyone has to do two projects.

  1. Using FAsta PY dealing with cyberinfrastructure (THis pays you)
  2. one dealing with AI do apply the first project

- Project Proposal: https://cybertraining-dsc.github.io/docs/report/project-faq/
- https://cybertraining-dsc.github.io/docs/report/other/
- https://cybertraining-dsc.github.io/docs/report/2021/
- https://cybertraining-dsc.github.io/docs/report/2021-reu/

5.1.3 Modules

- Students can pick a variety of modules in case they hav gaps:
  - Over 300 https://cybertraining-dsc.github.io/docs/modules/list/
  - 7 additional MOdules about DevOps https://cybertraining-dsc.github.io/docs/modules/devops/

5.1.4 Course material

https://cybertraining-dsc.github.io/docs/courses/ai-first/

5.1.4.1 Introduction to AI-Driven Digital Transformation

- <ttps://cybertraining-dsc.github.io/docs/modules/ai-first/2021/course_lectures/>

1. Introduction
2. Google Colab
3.

Other material

- https://cybertraining-dsc.github.io/docs/courses/
- 2020: https://cybertraining-dsc.github.io/docs/modules/bigdataapplications/2020/
- 2019 https://cybertraining-dsc.github.io/docs/modules/bigdataapplications/2019/

5.1.5 Publistaions

- https://cybertraining-dsc.github.io/docs/pub/

5.2 OVERVIEW

5.2.1 2022 Lecture on AI-First Deep Learning

Big Data Applications are an important topic that have impact in academia and industry.

An version from an earlier similar lecture is available at the following link

- 2021 Version.

5.2.2 Overview to AI-Driven Digital Transformation

An introductory Lecture with Motivation for Big Data Applications and Analytics Class.

This Lecture is recorded in 8 parts and gives an introduction and motivation for the class. This and other lectures in class are divided into “bite-sized lessons” from 5 to 30 minutes in length; that is why it has 8 parts.

The lecture explains what students might gain from the class even if they end up with different types of jobs from data engineering, software engineering, data science or a business (application) expert. It stresses that we are well into a transformation that impacts industry research and the way life is lived. This transformation is centered on using the digital way with clouds, edge computing and deep learning giving the implementation. This “AI-Driven Digital Transformation” is as transformational

as the Industrial Revolution in the past. We note that deep learning dominates most innovative AI replacing several traditional machine learning methods.

The slides for this course can be found at E534-Fall2020-Introduction

5.2.2.1 Big Data Applications and Analytics (A) This lesson describes briefly the trends driving and consequent of the AI-Driven Digital Transformation. It discusses the organizational aspects of the class and notes the two driving trends are clouds and AI. Clouds are mature and a dominant presence. AI is still rapidly changing and we can expect further major changes. The edge (devices and associated local fog computing) has always been important but now more is being done there.

- Youtube Video: http://youtube.com/watch?v=nCbJ-2kKf0s

5.2.2.2 Big Data Applications and Analytics (B) This lesson goes through the technologies (AI Edge Cloud) from 2008-2020 that are driving the AI-Driven Digital Transformation. we use Hype Cycles and Priority Matrices from Gartner tracking importance concepts from the Innovation Trigger, Peak of Inflated Expectations through the Plateau of Productivity. We contrast clouds and AI.

- Youtube Video: http://youtube.com/watch?v=QN-rPFbUAwc

5.2.2.3 Big Data Applications and Analytics (C)

- This gives illustrations of sources of big data.
- It gives key graphs of data sizes, images uploaded; computing, data, bandwidth trends;
- Cloud-Edge architecture.
- Intelligent machines and comparison of data from aircraft engine monitors compared to Twitter
- Youtube Video: http://youtube.com/watch?v=b_HeufZSEaE

5.2.2.4 Big Data Applications and Analytics (D)

- Multicore revolution
- Overall Global AI and Modeling Supercomputer GAIMSC
- Moores Law compared to Deep Learning computing needs
- Intel and NVIDIA status
- Youtube Video: http://youtube.com/watch?v=NxoOe5XKGyE

5.2.2.5 Big Data Applications and Analytics (E)

- Applications and Analytics
- Cyberinfrastructure, e-moreorlessanything.
- LHC, Higgs Boson and accelerators.
- Astronomy, SKA, multi-wavelength.
- Polar Grid.
- Genome Sequencing.
- Examples, Long Tail of Science.
- Wired's End of Science; the 4 paradigms.
- More data versus Better algorithms.
- Youtube Video: http://youtube.com/watch?v=C_jk161mQ_s

5.2.2.6 Big Data Applications and Analytics (F)

- Clouds, Service-oriented architectures, HPC High Performance Computing, Apace Software
- DIKW process illustrated by Google maps
- Raw data to Information/Knowledge/Wisdom/Decision Deluge from the EdgeInformation/Knowledge/Wisdom/Decision Deluge
- Parallel Computing
- Map Reduce
- Youtube Video: http://youtube.com/watch?v=V_iqzUXCcyg

5.2.2.7 Big Data Applications and Analytics (G) AI grows in importance and industries transform with

- Core Technologies related to
- New "Industries" over the last 25 years
- Traditional "Industries" Transformed; malls and other old industries transform
- Good to be master of Cloud Computing and Deep Learning
- AI-First Industries,
- Youtube Video: http://youtube.com/watch?v=DAJw6mitjao

5.2.2.8 Big Data Applications and Analytics (H)

- Job trends
- Become digitally savvy so you can take advantage of the AI/Cloud/Edge revolution with different jobs
- The qualitative idea of Big Data has turned into a quantitative realization as Cloud, Edge and Deep Learning
- Clouds are here to stay and one should plan on exploiting them
- Data Intensive studies in business and research continue to grow in importance
- Youtube Video: http://youtube.com/watch?v=nsgPI45XJlw

5.2.3 Overview AI-First Engineering Cybertraining

An introductiory Lecture about the class. It was given in 2021 and is still relevant. Introductory Lecture

- Youtube Video: http://youtube.com/watch?v=mjWXGR9NcHU

Assignment: please note the differences between the previous class and write them down and create a pull reuest to add them here.

5.2.3.1 Assignments

- Assignment Github: Get a github.com account
- Assignment Slack: Enroll in our slack (add link here)
- Assignment Github reu2022: add link here
- Post a 2-3 paragraph **formal** Bio in ere REU2022 Github repositorys
- Add links to the Bios and gitrepo here:
  - Name, linkto bio, link to gitrepor

5.2.3.2 Introduction

An introductiory lecture is available:

- Slides: IntroDLOpt: Introduction to Deep Learning and Optimization
- Youtube Video: http://youtube.com/watch?v=PCZZQ93xmGI_002

The slides and videos for deep learning are available:

- Introduction to Deep Learning and Optimization
  - Slides: Introduction to Deep Learning and Optimization

  - Youtube Video: http://youtube.com/watch?v=PCZZQ93xmGI_002

- Overview of Optimization
  - Slides: Overview of Optimization
  - Youtube Video: http://youtube.com/watch?v=kFBqj4U9XyA
- Deep Learning - Some examples
  - Slides: Deep Learning - Some examples
  - Youtube Video: http://youtube.com/watch?v=sJ-9YjgS35w
- Components of Deep Learning Systems
  - Slides: Components of Deep Learning
  - Youtube Video: http://youtube.com/watch?v=z9-OFr__Y-w
- Summary of Types of Deep Learning Systems
  - Slides: Types of Deep Learning Networks: Summary
  - Youtube Video: http://youtube.com/watch?v=4Ma3SFFLhYY

5.2.3.3 Deep Learning Examples, 1 We discuss deep learning examples covering first half of slides DLBasic: Deep Learning - Some examples

- Youtube Video: http://youtube.com/watch?v=whzCJUKJzN4

5.2.3.4 Deep Learning Examples, 2 plus Components We continue with deep learning examples Deep Learning: More Examples and Components

- Youtube Video: http://youtube.com/watch?v=tfDdb8mSL98

5.2.3.5 Deep Learning Networks plus Overview of Optimization Next, we cover two topics

- Deep Learning Networks Types of Deep Learning Networks: Summary
- General Issues in Optimization with presentation Presentation on Optimization
- Youtube Video: http://youtube.com/watch?v=hGUDURDMet8

5.2.3.6 Deep Learning and AI Examples in Health and Medicine 2/3rds of the presentation AI First Scenarios: Health and Medicine are covered in

- Youtube Video: http://youtube.com/watch?v=Efbvy4tAcNg
- the last 1/3rd of the presentation AI First Scenarios: Health and Medicine

as ell as

- AI First Scenarios - Space
- and a start to AI First Scenarios - Energy

are covered in the video

- Youtube Video: http://youtube.com/watch?v=EG6OH1cUN8c

5.2.3.7 Deep Learning and AI Examples

- We finished AI First Scenarios - Energy
- We started AI First Scenarios: Banking and FinTech
- Youtube Video: http://youtube.com/watch?v=_mIBVwLLTmI

5.2.3.8 Deep Learning and AI Examples

- We ended AI First Scenarios: Banking and FinTech
- We started AI Scenarios in Mobility and Transportation Systems
- Youtube Video: http://youtube.com/watch?v=_TbKRO5Tw4o

5.2.3.9 GitHub for the Class project

- We explain how to use GitHub for the class project. A video is available on YouTube. Please note that we only uploaded the relevant portion. The other half of the lecture went into individual comments for each student which we have not published. The comments are included in the GitHub repository.

Note project guidelines are given here

- Youtube Video: http://youtube.com/watch?v=PBMPWjavraU

5.2.3.10 The Final Project

- We described the gidelines of final projects in Slides
- We were impressed by the seven student presentations describing their chosen project and approach.
- Youtube Video: http://youtube.com/watch?v=gUhZdkke9A8

5.2.3.11 Practical Issues in Deep Learning for Earthquakes We used our research on Earthquake forecasting, to illustrate deep learning for Time Series with slides

- Youtube Video: http://youtube.com/watch?v=nQBr2347Xqg

5.2.3.12 Practical Issues in Deep Learning for Earthquakes We continued discussion that illustrated deep learning for Time Series with the same slides as last week

- Youtube Video: http://youtube.com/watch?v=jM2OJYDQE-Y

5.3 APPLICATIONS

5.3.1 Introduction to AI in Health and Medicine

This module discusses AI and the digital transformation for the Health and Medicine Area with a special emphasis on COVID-19 issues. We cover both the impact of COVID and some of the many activities that are addressing it. Parts B and C have an extensive general discussion of AI in Health and Medicine

The complete presentation is available at Google Slides while the videos are a YouTube playlist

5.3.1.1 Introduction (A) This lesson describes some overarching issues including the

- Summary in terms of Hypecycles
- Players in the digital health ecosystem and in particular role of Big Tech which has needed AI expertise and infrastructure from clouds to smart watches/phones
- Views of Pataients and Doctors on New Technology
- Role of clouds. This is essentially assumed throughout presentation but not stressed.
- Importance of Security
- Introduction to Internet of Medical Things; this area is discussed in more detail later in presernta-tion

- Slides
- Youtube Video: http://youtube.com/watch?v=08ZbpK5oe3A

5.3.1.2 Diagnostics (B) This highlights some diagnostic appliocations of AI and the digital transformation. Part C also has some diagnostic coverage – especially particular applications

- General use of AI in Diagnostics
- Early progress in diagnostic imaging including Radiology and Opthalmology
- AI In Clinical Decision Support
- Digital Therapeutics is a recognized and growing activity area
- Slides
- Youtube Video: http://youtube.com/watch?v=7-ULGwApFzM

5.3.1.3 Examples (C) This lesson covers a broad range of AI uses in Health and Medicine

- Flagging Issues requirng urgent attentation and more generally AI for Precision Merdicine
- Oncology and cancer have made early progress as exploit AI for images. Avoiding mistakes and diagnosing curable cervical cancer in developing countries with less screening.
- Predicting Gestational Diabetes
- cardiovascular diagnostics and AI to interpret and guide Ultrasound measurements
- Robot Nurses and robots to comfort patients
- AI to guide cosmetic surgery measuring beauty
- AI in analysis DNA in blood tests
- AI For Stroke detection (large vessel occlusion)
- AI monitoring of breathing to flag opioid-induced respiratory depression.
- AI to relieve administration burden including voice to text for Doctor's notes
- AI in consumer genomics
- Areas that are slow including genomics, Consumer Robotics, Augmented/Virtual Reality and Blockchain
- AI analysis of information resources flags probleme earlier
- Internet of Medical Things applications from watches to toothbrushes

- Slides
- Youtube Video: http://youtube.com/watch?v=Sp9h_-bf_I8

5.3.1.4 Impact of Covid-19 (D) This covers some aspects of the impact of COVID -19 pandedmic starting in March 2020

- The features of the first stimulus bill
- Impact on Digital Health, Banking, Fintech, Commerce – bricks and mortar, e-commerce, groceries, credit cards, advertising, connectivity, tech industry, Ride Hailing and Delivery,
- Impact on Restaurants, Airlines, Cruise lines, general travel, Food Delivery
- Impact of working from home and videoconferencing
- The economy and
- The often positive trends for Tech industry
- Slides
- Youtube Video: http://youtube.com/watch?v=E6EIZr6L7x4

5.3.1.5 Covid-19 and Recession (E) This is largely outdated as centered on start of pandemic induced recession. and we know what really happenmed now. Probably the pandemic accelerated the transformation of industry and the use of AI.

- Slides
- Youtube Video: http://youtube.com/watch?v=fGdyC3SVA1s

5.3.1.6 Tackling Covid-19 (F) This discusses some of AI and digital methods used to understand and reduce impact of COVID-19

- Robots for remote patient examination
- computerized tomography scan + AI to identify COVID-19
- Early activities of Big Tech and COVID
- Other early biotech activities with COVID-19
- Remote-work technology: Hopin, Zoom, Run the World, FreeConferenceCall, Slack, GroWrk, Webex, Lifesize, Google Meet, Teams
- Vaccines

- Wearables and Monitoring, Remote patient monitoring
- Telehealth, Telemedicine and Mobile Health
- Slides
- Youtube Video: http://youtube.com/watch?v=iPDNxRS7JmI

5.3.1.7 Data and Computational Science and Covid-19 (G) This lesson reviews some sophisticated high performance computing HPC and Big Data approaches to COVID

- Rosetta volunteer computer to analyze proteins
- COVID-19 High Performance Computing Consortium
- AI based drug discovery by startup Insilico Medicine
- Review of several research projects
- Global Pervasive Computational Epidemiology for COVID-19 studies
- Simulations of Virtual Tissues at Indiana University available on nanoHUB
- Slides
- Youtube Video: http://youtube.com/watch?v=02emPIxf_IA

5.3.1.8 Screening Drug and Candidates (H) A major project involving Department of Energy Supercomputers

- General Structure of Drug Discovery
- DeepDriveMD Project using AI combined with molecular dynamics to accelerate discovery of drug properties
- Slides
- Youtube Video: http://youtube.com/watch?v=tbZ_ObdOoFA

5.3.1.9 Areas for Covid19 Study and Pandemics as Complex Systems (I)

- Slides
- Possible Projects in AI for Health and Medicine and especially COVID-19
- Pandemics as a Complex System
- AI and computational Futures for Complex Systems
- Youtube Video: http://youtube.com/watch?v=F6vykp-gKoI

5.3.2 Mobility (Industry)

This section discusses the mobility in Industry

1. Industry being transformed by a) Autonomy (AI) and b) Electric power
2. Established Organizations can't change
    - General Motors (employees: 225,000 in 2016 to around 180,000 in 2018) finds it hard to compete with Tesla (42000 employees)
    - Market value GM was half the market value of Tesla at the start of 2020 but is now just 11% October 2020
    - GM purchased Cruise to compete
    - Funding and then buying startups is an important "transformation" strategy
3. Autonomy needs Sensors Computers Algorithms and Software
    - Also experience (training data)
    - Algorithms main bottleneck; others will automatically improve although lots of interesting work in new sensors, computers and software
    - Over the last 3 years, electrical power has gone from interesting to "bound to happen"; Tesla's happy customers probably contribute to this
    - Batteries and Charging stations needed

- Summary Slides
- Youtube Video: http://youtube.com/watch?v=1UYAq4VfpJ4_002
- Full Slide Deck

5.3.2.1 Introduction (A)

- Futures of Automobile Industry, Mobility, and Ride-Hailing
- Self-cleaning cars
- Medical Transportation
- Society of Automotive Engineers, Levels 0-5
- Gartner's conservative View
- Youtube Video: http://youtube.com/watch?v=cUn4FbJdkjY

5.3.2.2 Self Driving AI (B)

- Image processing and Deep Learning
- Examples of Self Driving cars
- Road construction Industry
- Role of Simulated data
- Role of AI in autonomy
- Fleet cars
- 3 Leaders: Waymo, Cruise, NVIDIA
- Youtube Video: http://youtube.com/watch?v=LgS05R23q3M

5.3.2.3 General Motors View (C)

- Talk by Dave Brooks at GM, “AI for Automotive Engineering”
- Zero crashes, zero emission, zero congestion
- GM moving to electric autonomous vehicles
- Youtube Video: http://youtube.com/watch?v=pybTudmAod0

5.3.2.4 Self Driving Snippets (D)

- Worries about and data on its Progress
- Tesla’s specialized self-driving chip
- Some tasks that are hard for AI
- Scooters and Bikes
- Youtube Video: http://youtube.com/watch?v=-YzXTfeVZMw

5.3.2.5 Electrical Power (E)

- Rise in use of electrical power
- Special opportunities in e-Trucks and time scale
- Future of Trucks
- Tesla market value
- Drones and Robot deliveries; role of 5G

- Robots in Logistics
- Youtube Video: http://youtube.com/watch?v=EwPsb_izTTo

Space and Energy

This section discusses the space and energy.

1. Energy sources and AI for powering Grids.
2. Energy Solution from Bill Gates
3. Space and AI

- Full Slide Deck

5.3.2.6 Energy (A)

- Distributed Energy Resources as a grid of renewables with a hierarchical set of Local Distribution Areas
- Electric Vehicles in Grid
- Economics of microgrids
- Investment into Clean Energy
- Batteries
- Fusion and Deep Learning for plasma stability
- AI for Power Grid, Virtual Power Plant, Power Consumption Monitoring, Electricity Trading
- Youtube Video: http://youtube.com/watch?v=rdGWniEHpT8
- Slides

5.3.2.7 Clean Energy startups from Bill Gates (B)

- 26 Startups in areas like long-duration storage, nuclear energy, carbon capture, batteries, fusion, and hydropower …
- The slide deck gives links to 26 companies from their website and pitchbook which describes their startup status (#employees, funding)
- It summarizes their products
- Youtube Video: http://youtube.com/watch?v=rgq99JSKoe8
- Slides

5.3.2.8 Space (C)

- Space supports AI with communications, image data and global navigation
- AI Supports space in AI-controlled remote manufacturing, imaging control, system control, dynamic spectrum use
- Privatization of Space - SpaceX, Investment
- 57,000 satellites through 2029
- Youtube Video: http://youtube.com/watch?v=bpXNLFa7piY
- Slides

5.3.3 AI In Banking

This section discusses AI in Banking

In this lecture, AI in Banking is discussed. Here we focus on the transition of legacy banks towards AI based banking, real world examples of AI in Banking, banking systems and banking as a service.

- Slides

5.3.3.1 The Transition of legacy Banks (A)

1. Types of AI that is used
2. Closing of physical branches
3. Making the transition
4. Growth in Fintech as legacy bank services decline

- Youtube Video: http://youtube.com/watch?v=h5xR8uisfSc

5.3.3.2 FinTech (B)

1. Fintech examples and investment
2. Broad areas of finance/banking where Fintech operating

- Youtube Video: http://youtube.com/watch?v=k0o74_e7xec

5.3.3.3 Neobanks (C)

1. Types and Examples of neobanks
2. Customer uptake by world region

3. Neobanking in Small and Medium Business segment
4. Neobanking in real estate, mortgages
5. South American Examples

- Youtube Video: http://youtube.com/watch?v=DMS_h_qkpDk

5.3.3.4 The System (D)

1. The Front, Middle, Back Office
2. Front Office: Chatbots
3. Robo-advisors
4. Middle Office: Fraud, Money laundering
5. Fintech
6. Payment Gateways (Back Office)
7. Banking as a Service

- Youtube Video: http://youtube.com/watch?v=V6KuN0yZQws

5.3.3.5 Examples (E)

1. Credit cards
2. The stock trading ecosystem
3. Robots counting coins
4. AI in Insurance: Chatbots, Customer Support
5. Banking itself
6. Handwriting recognition
7. Detect leaks for insurance

- Youtube Video: http://youtube.com/watch?v=0uWDWfeLLuY

5.3.3.6 Banking as a Service (E)

1. Banking Services Stack
2. Business Model
3. Several Examples
4. Metrics compared among examples
5. Breadth, Depth, Reputation, Speed to Market, Scalability

- Youtube Video: http://youtube.com/watch?v=UYY0-OJltqQ

5.3.4 Transportation Systems

This section discusses the transportation systems

1. The ride-hailing industry highlights the growth of a new “Transportation System” TS a. For ride-hailing TS controls rides matching drivers and customers; it predicts how to position cars and how to avoid traffic slowdowns b. However, TS is much bigger outside ride-hailing as we move into the “connected vehicle” era c. TS will probably find autonomous vehicles easier to deal with than human drivers
2. Cloud Fog and Edge components
3. Autonomous AI was centered on generalized image processing
4. TS also needs AI (and DL) but this is for routing and geospatial time-series; different technologies from those for image processing

- Slides
- Youtube Video: http://youtube.com/watch?v=1UYAq4VfpJ4_002

5.3.4.1 Introduction (A)

1. “Smart” Insurance
2. Fundamentals of Ride-Hailing

- Youtube Video: http://youtube.com/watch?v=aRBi94QTyvQ

5.3.4.2 Components of a Ride-Hailing System (B)

1. Transportation Brain and Services
2. Maps, Routing,
3. Traffic forecasting with deep learning

- Youtube Video: http://youtube.com/watch?v=obUYIj4cNao

5.3.4.3 Different AI Approaches in Ride-Hailing (C)

1. View as a Time Series: LSTM and ARIMA
2. View as an image in a 2D earth surface - Convolutional networks
3. Use of Graph Neural Nets
4. Use of Convolutional Recurrent Neural Nets
5. Spatio-temporal modeling
6. Comparison of data with predictions

7. Reinforcement Learning
8. Formulation of General Geospatial Time-Series Problem

- Youtube Video: http://youtube.com/watch?v=k9qFE5WdQWA

5.3.5 Commerce

This section discusses Commerce

- Slides

5.3.5.1 The Old way of doing things (A)

1. AI in Commerce
2. AI-First Engineering, Deep Learning
3. E-commerce and the transformation of “Bricks and Mortar”

- Youtube Video: http://youtube.com/watch?v=w07nShBHKRY

5.3.5.2 AI in Retail (B)

1. Personalization
2. Search
3. Image Processing to Speed up Shopping
4. Walmart

- Youtube Video: http://youtube.com/watch?v=Kqgbbrnaf1g

5.3.5.3 The Revolution that is Amazon (C)

1. Retail Revolution
2. Saves Time, Effort and Novelity with Modernized Retail
3. Looking ahead of Retail evolution

- Youtube Video: http://youtube.com/watch?v=M1TQp1CcjZk

5.3.5.4 DLMalls e-commerce (D)

1. Amazon sellers
2. Rise of Shopify
3. Selling Products on Amazon

- Youtube Video: http://youtube.com/watch?v=kaZaxn5aHaE

5.3.5.5 Recommender Engines, Digital media (E)

1. Spotify recommender engines
2. Collaborative Filtering
3. Audio Modelling
4. DNN for Recommender engines

- Youtube Video: http://youtube.com/watch?v=pY1KZu7EOn0

5.4 CLOUD COMPUTING

5.4.1 Cloud Computing

Cloud Computing

- Full Slide Deck
- Youtube Video: http://youtube.com/watch?v=FV0r-Yf6zNk

5.4.1.1 Defining the Cloud

5.4.1.1.1 Definition Basic definition of cloud and two very simple examples of why virtualization is important

1. How clouds are situated wrt HPC and supercomputers
2. Why multicore chips are important
3. Typical data center

Youtube Video: http://youtube.com/watch?v=8OE3oOVDmlQ

5.4.1.1.2 Service-oriented architectures Service-oriented architectures: Software services as Message-linked computing capabilities

1. The different aaS's: Network, Infrastructure, Platform, Software
2. The amazing services that Amazon AWS and Microsoft Azure have
3. Initial Gartner comments on clouds (they are now the norm) and evolution of servers; serverless and microservices
4. Gartner hypecycle and priority matrix on Infrastructure Strategies

- Youtube Video: http://youtube.com/watch?v=mzkFZApl_-4

5.4.1.1.3 Market Share

1. How important are they?
2. How much money do they make?

- Youtube Video: http://youtube.com/watch?v=jYfoNxlROqU

5.4.1.2 Virtualization

Virtualization Technologies, Hypervisors and the different approaches

1. KVM Xen, Docker and Openstack

- Youtube Video: http://youtube.com/watch?v=Tqsx-sEPY6M

5.4.1.3 Cloud Infrastructure

Comments on trends in the data center and its technologies

1. Clouds physically across the world
2. Green computing
3. Fraction of world's computing ecosystem in clouds and associated sizes
4. An analysis from Cisco of size of cloud computing

- Youtube Video: http://youtube.com/watch?v=6ncp-UTYk-4

Gartner hypecycle and priority matrix on Compute Infrastructure

1. Containers compared to virtual machines
2. The emergence of artificial intelligence as a dominant force

- Youtube Video: http://youtube.com/watch?v=R0Sk1rAS20M

5.4.1.4 Cloud Software

HPC-ABDS with over 350 software packages and how to use each of 21 layers

1. Google's software innovations
2. MapReduce in pictures
3. Cloud and HPC software stacks compared
4. Components need to support cloud/distributed system programming

- Youtube Video: http://youtube.com/watch?v=gMAA9y8khMM

5.4.1.5 Cloud Applications Clouds in science where area called cyberinfrastructure; the science usage pattern from NIST

1. Artificial Intelligence from Gartner

- Youtube Video: http://youtube.com/watch?v=sAbYkZHD81U

Characterize Applications using NIST approach

1. Internet of Things
2. Different types of MapReduce

- Youtube Video: http://youtube.com/watch?v=YaPg-OOlkdw

5.4.1.6 Parallel Computing Analogies

5.4.1.6.1 Parallel Computing in pictures

1. Some useful analogies and principles

- Youtube Video: http://youtube.com/watch?v=iqVHvQg851s

5.4.1.6.2 Real Parallel Computing Single Program/Instruction Multiple Data SIMD SPMD

1. Big Data and Simulations Compared
2. What is hard to do?

- Youtube Video: http://youtube.com/watch?v=7gD2yyTKatM

5.4.1.7 Storage Cloud data approaches

1. Repositories, File Systems, Data lakes

- Youtube Video: http://youtube.com/watch?v=NcIbKQ-AChA

5.4.1.8 HPC and Clouds The Branscomb Pyramid

1. Supercomputers versus clouds
2. Science Computing Environments

- Youtube Video: http://youtube.com/watch?v=QTxYKhpVtDw

5.4.1.9 Comparison of Data Analytics with Simulation Structure of different applications for simulations and Big Data

1. Software implications
2. Languages

- Youtube Video: http://youtube.com/watch?v=6WmWE_7iB3w

5.4.1.10 The Future Gartner cloud computing hypecycle and priority matrix 2017 and 2019

1. Hyperscale computing
2. Serverless and FaaS
3. Cloud Native
4. Microservices
5. Update to 2019 Hypecycle

- Youtube Video: http://youtube.com/watch?v=1xH5ow5kOhc

Security and Blockchain

- Youtube Video: http://youtube.com/watch?v=9AmVHEXv7gY

Fault Tolerance

- Youtube Video: http://youtube.com/watch?v=VnkkCix3yEE

5.5 COLAB

5.5.1 Google Colab

Google colab is a valuable tool for students that are not able to set up their compures for deep learning or have access to GPU's. It provides zero install access to GPUs through the Web browser.

We provide a lecture plus four Python notebooks

- First DL: Deep Learning MNIST Example Spring
- Welcome To Colaboratory
- Google Colab: A gentle introduction to Google Colab for Programming
- Python Warm Up
- MNIST Classification on Google Colab
- Youtube Video: http://youtube.com/watch?v=r-aD0oHVAdA

5.5.2 Python On Colab

Python Exercise on Google Colab

5.5.2.1 Python Exercise on Google Colab

- Open In Colab
- View in Github
- Download
- Python Textbook

In this exercise, we will take a look at some basic Python Concepts needed for day-to-day coding.

- Youtube Video: http://youtube.com/watch?v=x1ICvWDlvB0

Check the installed Python version.

```
! python --version
```

Python 3.7.6

5.5.2.2 Simple For Loop

```
for i in range(10):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

5.5.2.3 List

```
list_items = ['a', 'b', 'c', 'd', 'e']
```

5.5.2.3.1 Retrieving an Element

```
list_items[2]

'c'
```

5.5.2.3.2 Append New Values

```
list_items.append('f')
list_items

['a', 'b', 'c', 'd', 'e', 'f']
```

5.5.2.3.3 Remove an Element

```
list_items.remove('a')
list_items

['b', 'c', 'd', 'e', 'f']
```

- Youtube Video: http://youtube.com/watch?v=oudT4sRIuwU

5.5.2.4 Dictionary

```
dictionary_items = {'a':1, 'b': 2, 'c': 3}
```

5.5.2.4.1 Retrieving an Item by Key

```
dictionary_items['b']

2
```

5.5.2.4.2 Append New Item with Key

```
dictionary_items['c'] = 4
dictionary_items

{'a': 1, 'b': 2, 'c': 4}
```

5.5.2.4.3 Delete an Item with Key

```
del dictionary_items['a']
dictionary_items

{'b': 2, 'c': 4}
```

5.5.2.5 Comparators

```
x = 10
y = 20
z = 30
x > y

False

x < z

True

z == x

False

if x < z:
  print("This is True")

This is True

if x > z:
  print("This is True")
else:
  print("This is False")
```

```
This is False
```

- Youtube Video: http://youtube.com/watch?v=GKU6-SNZGQc

5.5.2.6 Arithmetic

```
k = x * y * z
k

6000

j = x + y + z
j

60

m = x -y
m

-10

n = x / z
n

0.3333333333333333
```

5.5.2.7 Numpy

5.5.2.7.1 Create a Random Numpy Array

```
import numpy as np
a = np.random.rand(100)
a.shape

(100,)
```

5.5.2.7.2 Reshape Numpy Array

```
b = a.reshape(10,10)
b.shape

(10, 10)
```

5.5.2.7.3 Manipulate Array Elements

```
c = b * 10
c[0]

array([3.33575458, 7.39029235, 5.54086921, 9.88592471, 4.9246252 ,
       1.76107178, 3.5817523 , 3.74828708, 3.57490794, 6.55752319])

c = np.mean(b,axis=1)
c.shape

10

print(c)

[0.60673061 0.4223565  0.42687517 0.6260857  0.60814217 0.66445627
  0.54888432 0.68262262 0.42523459 0.61504903]
```

5.5.3 Deep Learning Colab Examples

5.5.3.1 Distributed Training for MNIST Distributed Training for MNIST

- Open In Colab
- Open in GitHub
- Download

In this lesson we discuss in how to create a simple IPython Notebook to solve an image classification problem with Multi Layer Perceptron with LSTM.

5.5.3.1.1 Pre-requisites Install the following Python packages

1. cloudmesh-installer

2. cloudmesh-common

```
pip3 install cloudmesh-installer
pip3 install cloudmesh-common
```

5.5.3.2 Sample MLP + LSTM with Tensorflow Keras

5.5.3.2.1 Import Libraries

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, SimpleRNN, InputLayer, LSTM,
    ↪ Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist
from cloudmesh.common.StopWatch import StopWatch
```

5.5.3.2.2 Download Data and Pre-Process

```
StopWatch.start("data-load")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
StopWatch.stop("data-load")


StopWatch.start("data-pre-process")
num_labels = len(np.unique(y_train))


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


image_size = x_train.shape[1]
x_train = np.reshape(x_train,[-1, image_size, image_size])
x_test = np.reshape(x_test,[-1, image_size, image_size])
```

```
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
StopWatch.stop("data-pre-process")

input_shape = (image_size, image_size)
batch_size = 128
units = 256
dropout = 0.2
```

5.5.3.2.3 Define Model Here we use the Tensorflow distributed training components to train the model in multiple CPUs or GPUs. In the Colab instance multiple GPUs are not supported. Hence, the training must be done in the device type 'None' when selecting the 'runtime type' from Runtime menu. To run with multiple-GPUs no code change is required. Learn more about distributed training.

```
StopWatch.start("compile")
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():
  model = Sequential()
  # LSTM Layers
  model.add(LSTM(units=units,
                     input_shape=input_shape,
                     return_sequences=True))
  model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=True))
  model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=False))
  # MLP Layers
  model.add(Dense(units))
  model.add(Activation('relu'))
  model.add(Dropout(dropout))
  model.add(Dense(units))
  model.add(Activation('relu'))
  model.add(Dropout(dropout))
  # Softmax_layer
  model.add(Dense(num_labels))
  model.add(Activation('softmax'))
  model.summary()
  plot_model(model, to_file='rnn-mnist.png', show_shapes=True)

  print("Number of devices: {}".format(strategy.num_replicas_in_sync))

  model.compile(loss='categorical_crossentropy',
                optimizer='sgd',
```

```
                metrics=['accuracy'])
StopWatch.stop("compile")

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 28, 256)           291840
_________________________________________________________________
lstm_7 (LSTM)                (None, 28, 256)           525312
_________________________________________________________________
lstm_8 (LSTM)                (None, 256)               525312
_________________________________________________________________
dense_6 (Dense)              (None, 256)               65792
_________________________________________________________________
activation_6 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               65792
_________________________________________________________________
activation_7 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_8 (Activation)    (None, 10)                0
=================================================================
Total params: 1,476,618
Trainable params: 1,476,618
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task
    ↪ :0/device:GPU:0',)
Number of devices: 1
INFO:tensorflow:Reduce to /job:localhost/replica:0/task:0/device:CPU:0 then
    ↪ broadcast to ('/job:localhost/replica:0/task:0/device:CPU:0',).
INFO:tensorflow:Reduce to /job:localhost/replica:0/task:0/device:CPU:0 then
    ↪ broadcast to ('/job:localhost/replica:0/task:0/device:CPU:0',).
INFO:tensorflow:Reduce to /job:localhost/replica:0/task:0/device:CPU:0 then
    ↪ broadcast to ('/job:localhost/replica:0/task:0/device:CPU:0',).
INFO:tensorflow:Reduce to /job:localhost/replica:0/task:0/device:CPU:0 then
    ↪ broadcast to ('/job:localhost/replica:0/task:0/device:CPU:0',).
```

5.5.3.2.4 Train

```
StopWatch.start("train")
model.fit(x_train, y_train, epochs=30, batch_size=batch_size)
StopWatch.stop("train")

Epoch 1/30
469/469 [==============================] - 7s 16ms/step - loss: 2.0427 - accuracy:
    ↪ 0.2718
Epoch 2/30
469/469 [==============================] - 7s 16ms/step - loss: 1.6934 - accuracy:
    ↪ 0.4007
Epoch 3/30
469/469 [==============================] - 7s 16ms/step - loss: 1.2997 - accuracy:
    ↪ 0.5497
...
Epoch 28/30
469/469 [==============================] - 8s 17ms/step - loss: 0.1175 - accuracy:
    ↪ 0.9640
Epoch 29/30
469/469 [==============================] - 8s 17ms/step - loss: 0.1158 - accuracy:
    ↪ 0.9645
Epoch 30/30
469/469 [==============================] - 8s 17ms/step - loss: 0.1098 - accuracy:
    ↪ 0.9661
```

5.5.3.2.5 Test

```
StopWatch.start("evaluate")
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
StopWatch.stop("evaluate")

StopWatch.benchmark()

79/79 [==============================] - 3s 9ms/step - loss: 0.0898 - accuracy:
    ↪ 0.9719

Test accuracy: 97.2%

+------------------+----------+---------+---------+--------------------+-------+--------------+--
    ↪
```

```
| Name              | Status   |    Time |     Sum | Start               | tag   |
    ↪ Node          | User   | OS    | Version                         |
|-------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
| data-load         | failed   |   0.473 |   0.473 | 2021-03-07 11:34:03 |       |
    ↪ b39e0899c1f8 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| data-pre-process | failed   |   0.073 |   0.073 | 2021-03-07 11:34:03 |       |
    ↪ b39e0899c1f8 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| compile           | failed   |   0.876 |   7.187 | 2021-03-07 11:38:05 |       |
    ↪ b39e0899c1f8 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| train             | failed   | 229.341 | 257.023 | 2021-03-07 11:38:44 |       |
    ↪ b39e0899c1f8 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| evaluate          | failed   |   2.659 |   4.25  | 2021-03-07 11:44:54 |       |
    ↪ b39e0899c1f8 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
+-------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
```

5.5.3.2.6 References

1. Advance Deep Learning with Keras
2. Distributed With Tensorflow
3. Keras with Tensorflow Distributed Training

5.5.3.3 MLP + LSTM with MNIST on Google Colab MLP + LSTM with MNIST on Google Colab

- Open In Colab
- Open in GitHub
- Download

In this lesson we discuss in how to create a simple IPython Notebook to solve an image classification problem with Multi Layer Perceptron with LSTM.

5.5.3.3.1 Pre-requisites Install the following Python packages

1. cloudmesh-installer
2. cloudmesh-common

```
pip3 install cloudmesh-installer
pip3 install cloudmesh-common
```

5.5.3.3.2 Sample MLP + LSTM with Tensorflow Keras

5.5.3.3.3 Import Libraries

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, SimpleRNN, InputLayer, LSTM,
    ↪ Dropout
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist
from cloudmesh.common.StopWatch import StopWatch
```

5.5.3.3.4 Download Data and Pre-Process

```
StopWatch.start("data-load")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
StopWatch.stop("data-load")


StopWatch.start("data-pre-process")
num_labels = len(np.unique(y_train))


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


image_size = x_train.shape[1]
x_train = np.reshape(x_train,[-1, image_size, image_size])
x_test = np.reshape(x_test,[-1, image_size, image_size])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
StopWatch.stop("data-pre-process")

input_shape = (image_size, image_size)
batch_size = 128
units = 256
```

```
dropout = 0.2
```

5.5.3.3.5 Define Model

```
StopWatch.start("compile")
model = Sequential()
# LSTM Layers
model.add(LSTM(units=units,
                     input_shape=input_shape,
                     return_sequences=True))
model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=True))
model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=False))
# MLP Layers
model.add(Dense(units))
model.add(Activation('relu'))
model.add(Dropout(dropout))
model.add(Dense(units))
model.add(Activation('relu'))
model.add(Dropout(dropout))
# Softmax_layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='rnn-mnist.png', show_shapes=True)


model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
StopWatch.stop("compile")

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 28, 256)           291840
_________________________________________________________________
lstm_1 (LSTM)                (None, 28, 256)           525312
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               525312
```

```
_________________________________________________________________
dense (Dense)                (None, 256)               65792
_________________________________________________________________
activation (Activation)      (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
activation_1 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 1,476,618
Trainable params: 1,476,618
Non-trainable params: 0
```

5.5.3.3.6 Train

```
StopWatch.start("train")
model.fit(x_train, y_train, epochs=30, batch_size=batch_size)
StopWatch.stop("train")

469/469 [==============================] - 378s 796ms/step - loss: 2.2689 - accuracy
    ↪ : 0.2075
```

5.5.3.3.7 Test

```
StopWatch.start("evaluate")
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
StopWatch.stop("evaluate")

StopWatch.benchmark()

79/79 [==============================] - 1s 7ms/step - loss: 2.2275 - accuracy:
    ↪ 0.3120
```

```
Test accuracy: 31.2%

+------------------+----------+--------+--------+---------------------+-------+--------------+----
    ↪
| Name             | Status   |   Time |    Sum | Start               | tag   | Node
    ↪          | User   | OS    | Version                           |
|------------------+----------+--------+--------+---------------------+-------+--------------+----
    ↪
| data-load        | failed   |  0.61  |  0.61  | 2021-02-21 21:35:06 |       | 9810
    ↪ ccb69d08 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| data-pre-process | failed   |  0.076 |  0.076 | 2021-02-21 21:35:07 |       | 9810
    ↪ ccb69d08 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| compile          | failed   |  6.445 |  6.445 | 2021-02-21 21:35:07 |       | 9810
    ↪ ccb69d08 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| train            | failed   | 17.171 | 17.171 | 2021-02-21 21:35:13 |       | 9810
    ↪ ccb69d08 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| evaluate         | failed   |  1.442 |  1.442 | 2021-02-21 21:35:31 |       | 9810
    ↪ ccb69d08 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
+------------------+----------+--------+--------+---------------------+-------+--------------+----
    ↪
```

5.5.3.3.8 References Orignal Source to Source Code

5.5.3.4 MNIST Classification on Google Colab MNIST Classification on Google Colab

- Open In Colab
- Open in GitHub
- Download

In this lesson we discuss in how to create a simple IPython Notebook to solve an image classification problem. MNIST contains a set of pictures

5.5.3.4.1 Import Libraries Note: https://python-future.org/quickstart.html

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
```

```
from keras.utils import to_categorical, plot_model
from keras.datasets import mnist
```

5.5.3.4.2 Warm Up Exercise

5.5.3.4.3 Pre-process data Load data

First we load the data from the inbuilt mnist dataset from Keras Here we have to split the data set into training and testing data. The training data or testing data has two components. Training features and training labels. For instance every sample in the dataset has a corresponding label. In Mnist the training sample contains image data represented in terms of an array. The training labels are from 0-9.

Here we say x_train for training data features and y_train as the training labels. Same goes for testing data.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
    ↪ mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

5.5.3.4.4 Identify Number of Classes As this is a number classification problem. We need to know how many classes are there. So we'll count the number of unique labels.

```
num_labels = len(np.unique(y_train))
```

####Convert Labels To One-Hot Vector

Read more on one-hot vector.

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

5.5.3.4.5 Image Reshaping The training model is designed by considering the data as a vector. This is a model dependent modification. Here we assume the image is a squared shape image.

```
image_size = x_train.shape[1]
input_size = image_size * image_size
```

5.5.3.4.6 Resize and Normalize The next step is to continue the reshaping to a fit into a vector and normalize the data. Image values are from 0 - 255, so an easy way to normalize is to divide by the maximum value.

```
x_train = np.reshape(x_train, [-1, input_size])
x_train = x_train.astype('float32') / 255
x_test = np.reshape(x_test, [-1, input_size])
x_test = x_test.astype('float32') / 255
```

5.5.3.4.7 Create a Keras Model Keras is a neural network library. The summary function provides tabular summary on the model you created. And the plot_model function provides a grpah on the network you created.

```
# Create Model
# network parameters
batch_size = 4
hidden_units = 64

model = Sequential()
model.add(Dense(hidden_units, input_dim=input_size))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_6 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

Assignment: include image

5.5.3.4.8 Compile and Train A keras model need to be compiled before it can be used to train the model. In the compile function, you can provide the optimization that you want to add, metrics you expect and the type of loss function you need to use.

Here we use adam optimizer, a famous optimizer used in neural networks.

The loss funtion we have used is the categorical_crossentropy.

Once the model is compiled, then the fit function is called upon passing the number of epochs, traing data and batch size.

The batch size determines the number of elements used per minibatch in optimizing the function.

Note: Change the number of epochs, batch size and see what happens.

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=1, batch_size=batch_size)

469/469 [==============================] - 3s 7ms/step - loss: 0.3647 - accuracy:
    ↪ 0.8947




<tensorflow.python.keras.callbacks.History at 0x7fe88faf4c50>
```

5.5.3.4.9 Testing Now we can test the trained model. Use the evaluate function by passing test data and batch size and the accuracy and the loss value can be retrieved.

MNIST_V1.0|Exercise: Try to observe the network behavior by changing the number of epochs, batch size and record the best accuracy that you can gain. Here you can record what happens when you change these values. Describe your observations in 50-100 words.

```
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))

79/79 [==============================] - 0s 4ms/step - loss: 0.2984 - accuracy:
    ↪ 0.9148

Test accuracy: 91.5%
```

5.5.3.4.10 Final Note This programme can be defined as a hello world programme in deep learning. Objective of this exercise is not to teach you the depths of deep learning. But to teach you basic concepts that may need to design a simple network to solve a problem. Before running the whole code, read all the instructions before a code section.

5.5.3.4.11 Assignments

1. Solve Exercise MNIST_V1.0.

5.5.3.4.12 References Orignal Source to Source Code

5.5.3.5 MNIST With PyTorch MNIST With PyTorch

- Open In Colab
- Open in GitHub
- Download

In this lesson we discuss in how to create a simple IPython Notebook to solve an image classification problem with Multi Layer Perceptron with PyTorch.

5.5.3.5.1 Import Libraries

```
import numpy as np
import torch
import torchvision
import matplotlib.pyplot as plt
from torchvision import datasets, transforms
from torch import nn
from torch import optim
from time import time
import os
from google.colab import drive
```

5.5.3.5.2 Pre-Process Data Here we download the data using PyTorch data utils and transform the data by using a normalization function. PyTorch provides a data loader abstraction called a `DataLoader` where we can set the batch size, data shuffle per batch loading. Each data loader expecte a Pytorch Dataset. The DataSet abstraction and DataLoader usage can be found here

```
# Data transformation function
transform = transforms.Compose([transforms.ToTensor(),
```

```
                                transforms.Normalize((0.5,), (0.5,)),
                                ])

# DataSet
train_data_set = datasets.MNIST('drive/My Drive/mnist/data/', download=True, train=
    ↪ True, transform=transform)
validation_data_set = datasets.MNIST('drive/My Drive/mnist/data/', download=True,
    ↪ train=False, transform=transform)

# DataLoader
train_loader = torch.utils.data.DataLoader(train_data_set, batch_size=32, shuffle=
    ↪ True)
validation_loader = torch.utils.data.DataLoader(validation_data_set, batch_size=32,
    ↪ shuffle=True)
```

5.5.3.5.3 Define Network Here we select the matching input size compared to the network definition. Here data reshaping or layer reshaping must be done to match input data shape with the network input shape. Also we define a set of hidden unit sizes along with the output layers size. The `output_size` must match with the number of labels associated with the classification problem. The hidden units can be chosesn depending on the problem. `nn.Sequential` is one way to create the network. Here we stack a set of linear layers along with a softmax layer for the classification as the output layer.

```
input_size = 784
hidden_sizes = [128, 128, 64, 64]
output_size = 10

model = nn.Sequential(nn.Linear(input_size, hidden_sizes[0]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[0], hidden_sizes[1]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[1], hidden_sizes[2]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[2], hidden_sizes[3]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[3], output_size),
                      nn.LogSoftmax(dim=1))


print(model)

Sequential(
  (0): Linear(in_features=784, out_features=128, bias=True)
  (1): ReLU()
```

```
  (2): Linear(in_features=128, out_features=128, bias=True)
  (3): ReLU()
  (4): Linear(in_features=128, out_features=64, bias=True)
  (5): ReLU()
  (6): Linear(in_features=64, out_features=64, bias=True)
  (7): ReLU()
  (8): Linear(in_features=64, out_features=10, bias=True)
  (9): LogSoftmax(dim=1)
)
```

5.5.3.5.4 Define Loss Function and Optimizer Read more about Loss Functions and Optimizers supported by PyTorch.

```
criterion = nn.NLLLoss()
optimizer = optim.SGD(model.parameters(), lr=0.003, momentum=0.9)
```

5.5.3.5.5 Train

```
epochs = 5

for epoch in range(epochs):
    loss_per_epoch = 0
    for images, labels in train_loader:
        images = images.view(images.shape[0], -1)

        # Gradients cleared per batch
        optimizer.zero_grad()

        # Pass input to the model
        output = model(images)
        # Calculate loss after training compared to labels
        loss = criterion(output, labels)

        # backpropagation
        loss.backward()

        # optimizer step to update the weights
        optimizer.step()

        loss_per_epoch += loss.item()
    average_loss = loss_per_epoch / len(train_loader)
    print("Epoch {} - Training loss: {}".format(epoch, average_loss))
```

```
Epoch 0 - Training loss: 1.3052690227402808
Epoch 1 - Training loss: 0.33809808635317695
Epoch 2 - Training loss: 0.22927882223685922
Epoch 3 - Training loss: 0.16807103878669521
Epoch 4 - Training loss: 0.1369301250545995
```

5.5.3.5.6 Model Evaluation Similar to training data loader, we use the validation loader to load batch by batch and run the feed-forward network to get the expected prediction and compared to the label associated with the data point.

```
correct_predictions, all_count = 0, 0
# enumerate data from the data validation loader (loads a batch at a time)
for batch_id, (images,labels) in enumerate(validation_loader):
  for i in range(len(labels)):
    img = images[i].view(1, 784)
    # at prediction stage, only feed-forward calculation is required.
    with torch.no_grad():
        logps = model(img)

    # Output layer of the network uses a LogSoftMax layer
    # Hence the probability must be calculated with the exponential values.
    # The final layer returns an array of probabilities for each label
    # Pick the maximum probability and the corresponding index
    # The corresponding index is the predicted label
    ps = torch.exp(logps)
    probab = list(ps.numpy()[0])
    pred_label = probab.index(max(probab))
    true_label = labels.numpy()[i]
    if(true_label == pred_label):
      correct_predictions += 1
    all_count += 1

print(f"Model Accuracy {(correct_predictions/all_count) * 100} %")

Model Accuracy 95.95 %
```

5.5.3.5.7 References

1. Torch NN Sequential
2. Handwritten Digit Recognition Using PyTorch — Intro To Neural Networks
3. MNIST Handwritten Digit Recognition in PyTorch

5.5.3.6 MNIST-AutoEncoder Classification on Google Colab

MNIST with AutoEncoder: Classification on Google Colab

- Open In Colab
- Open in GitHub
- Download

5.5.3.6.1 Prerequisites

Install the following packages

```
! pip3 install cloudmesh-installer
! pip3 install cloudmesh-common
```

5.5.3.6.2 Import Libraries

```
import tensorflow as tf
from keras.layers import Dense, Input
from keras.layers import Conv2D, Flatten
from keras.layers import Reshape, Conv2DTranspose
from keras.models import Model
from keras.datasets import mnist
from keras.utils import plot_model
from keras import backend as K

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

5.5.3.6.3 Download Data and Pre-Process

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

image_size = x_train.shape[1]
x_train = np.reshape(x_train, [-1, image_size, image_size, 1])
x_test = np.reshape(x_test, [-1, image_size, image_size, 1])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

input_shape = (image_size, image_size, 1)
batch_size = 32
kernel_size = 3
```

```
latent_dim = 16
hidden_units = [32, 64]

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
    ↪ mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

5.5.3.6.4 Define Model

```
inputs = Input(shape=input_shape, name='encoder_input')
x = inputs
x = Dense(hidden_units[0], activation='relu')(x)
x = Dense(hidden_units[1], activation='relu')(x)

shape = K.int_shape(x)

# generate latent vector
x = Flatten()(x)
latent = Dense(latent_dim, name='latent_vector')(x)

# instantiate encoder model
encoder = Model(inputs,
                latent,
                name='encoder')
encoder.summary()
plot_model(encoder,
           to_file='encoder.png',
           show_shapes=True)


latent_inputs = Input(shape=(latent_dim,), name='decoder_input')
x = Dense(shape[1] * shape[2] * shape[3])(latent_inputs)
x = Reshape((shape[1], shape[2], shape[3]))(x)
x = Dense(hidden_units[0], activation='relu')(x)
x = Dense(hidden_units[1], activation='relu')(x)

outputs = Dense(1, activation='relu')(x)

decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
plot_model(decoder, to_file='decoder.png', show_shapes=True)

autoencoder = Model(inputs,
                    decoder(encoder(inputs)),
```

```
                    name='autoencoder')
autoencoder.summary()
plot_model(autoencoder,
           to_file='autoencoder.png',
           show_shapes=True)

autoencoder.compile(loss='mse', optimizer='adam')

Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 28, 28, 1)]       0
_________________________________________________________________
dense_2 (Dense)              (None, 28, 28, 32)        64
_________________________________________________________________
dense_3 (Dense)              (None, 28, 28, 64)        2112
_________________________________________________________________
flatten_1 (Flatten)          (None, 50176)             0
_________________________________________________________________
latent_vector (Dense)        (None, 16)                802832
=================================================================
Total params: 805,008
Trainable params: 805,008
Non-trainable params: 0
_________________________________________________________________
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 16)]              0
_________________________________________________________________
dense_4 (Dense)              (None, 50176)             852992
_________________________________________________________________
reshape (Reshape)            (None, 28, 28, 64)        0
_________________________________________________________________
dense_5 (Dense)              (None, 28, 28, 32)        2080
_________________________________________________________________
dense_6 (Dense)              (None, 28, 28, 64)        2112
_________________________________________________________________
dense_7 (Dense)              (None, 28, 28, 1)         65
=================================================================
Total params: 857,249
Trainable params: 857,249
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 28, 28, 1)]       0
_________________________________________________________________
encoder (Functional)         (None, 16)                805008
_________________________________________________________________
decoder (Functional)         (None, 28, 28, 1)         857249
=================================================================
Total params: 1,662,257
Trainable params: 1,662,257
Non-trainable params: 0
```

5.5.3.6.5 Train

```
autoencoder.fit(x_train,
                x_train,
                validation_data=(x_test, x_test),
                epochs=1,
                batch_size=batch_size)

1875/1875 [==============================] - 112s 60ms/step - loss: 0.0268 -
    ↪ val_loss: 0.0131

<tensorflow.python.keras.callbacks.History at 0x7f3ecb2e0be0>
```

5.5.3.6.6 Test

```
x_decoded = autoencoder.predict(x_test)

79/79 [==============================] - 7s 80ms/step - loss: 0.2581 - accuracy:
    ↪ 0.9181

Test accuracy: 91.8%
```

5.5.3.6.7 Visualize

```
imgs = np.concatenate([x_test[:8], x_decoded[:8]])
imgs = imgs.reshape((4, 4, image_size, image_size))
imgs = np.vstack([np.hstack(i) for i in imgs])
plt.figure()
plt.axis('off')
plt.title('Input: 1st 2 rows, Decoded: last 2 rows')
plt.imshow(imgs, interpolation='none', cmap='gray')
plt.savefig('input_and_decoded.png')
plt.show()
```

5.5.3.7 MNIST-CNN Classification on Google Colab MNIST with Convolutional Neural Networks: Classification on Google Colab

- Open In Colab
- Open in GitHub
- Download

5.5.3.7.1 Prerequisites Install the following packages

```
! pip3 install cloudmesh-installer
! pip3 install cloudmesh-common
```

5.5.3.7.2 Import Libraries

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout
from keras.layers import Conv2D, MaxPooling2D, Flatten, AveragePooling2D
from keras.utils import to_categorical, plot_model
from keras.datasets import mnist
```

5.5.3.7.3 Download Data and Pre-Process

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

num_labels = len(np.unique(y_train))

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

image_size = x_train.shape[1]
x_train = np.reshape(x_train,[-1, image_size, image_size, 1])
x_test = np.reshape(x_test,[-1, image_size, image_size, 1])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

input_shape = (image_size, image_size, 1)
print(input_shape)
batch_size = 128
kernel_size = 3
pool_size = 2
filters = 64
dropout = 0.2

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
    ↪ mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
(28, 28, 1)
```

5.5.3.7.4 Define Model

```
model = Sequential()
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu',
                 input_shape=input_shape,
                 padding='same'))
model.add(MaxPooling2D(pool_size))
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu',
                 input_shape=input_shape,
                 padding='same'))
model.add(MaxPooling2D(pool_size))
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
```

```
                 activation='relu',
                 padding='same'))
model.add(MaxPooling2D(pool_size))
model.add(Conv2D(filters=filters,
                 kernel_size=kernel_size,
                 activation='relu'))
model.add(Flatten())
model.add(Dropout(dropout))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='cnn-mnist.png', show_shapes=True)

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 28, 28, 64)        640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 1, 1, 64)          36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 112,074
Trainable params: 112,074
Non-trainable params: 0
_________________________________________________________________
```

5.5.3.7.5 Train

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
# train the network
model.fit(x_train, y_train, epochs=10, batch_size=batch_size)

469/469 [==============================] - 125s 266ms/step - loss: 0.6794 - accuracy
    ↪ : 0.7783

<tensorflow.python.keras.callbacks.History at 0x7f35d4b104e0>
```

5.5.3.7.6 Test

```
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))

79/79 [==============================] - 6s 68ms/step - loss: 0.0608 - accuracy:
    ↪ 0.9813

Test accuracy: 98.1%
```

5.5.3.8 MNIST-LSTM Classification on Google Colab MNIST-LSTM Classification on Google Colab

- Open In Colab
- Open in GitHub
- Download

5.5.3.8.1 Pre-requisites Install the following Python packages

1. cloudmesh-installer
2. cloudmesh-common

```
pip3 install cloudmesh-installer
pip3 install cloudmesh-common
```

5.5.3.8.2 Sample LSTM with Tensorflow Keras

5.5.3.8.3 Import Libraries

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, SimpleRNN, InputLayer, LSTM
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist
from cloudmesh.common.StopWatch import StopWatch
```

5.5.3.8.4 Download Data and Pre-Process

```
StopWatch.start("data-load")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
StopWatch.stop("data-load")


StopWatch.start("data-pre-process")
num_labels = len(np.unique(y_train))


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


image_size = x_train.shape[1]
x_train = np.reshape(x_train,[-1, image_size, image_size])
x_test = np.reshape(x_test,[-1, image_size, image_size])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
StopWatch.stop("data-pre-process")

input_shape = (image_size, image_size)
batch_size = 128
units = 256
dropout = 0.2
```

5.5.3.8.5 Define Model

```
StopWatch.start("compile")
model = Sequential()
model.add(LSTM(units=units,
                     input_shape=input_shape,
                     return_sequences=True))
model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=True))
model.add(LSTM(units=units,
                     dropout=dropout,
                     return_sequences=False))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='rnn-mnist.png', show_shapes=True)


model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
StopWatch.stop("compile")

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 28, 256)           291840
_________________________________________________________________
lstm_4 (LSTM)                (None, 28, 256)           525312
_________________________________________________________________
lstm_5 (LSTM)                (None, 256)               525312
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 1,345,034
Trainable params: 1,345,034
Non-trainable params: 0
```

5.5.3.8.6 Train

```
StopWatch.start("train")
model.fit(x_train, y_train, epochs=1, batch_size=batch_size)
StopWatch.stop("train")

469/469 [==============================] - 378s 796ms/step - loss: 2.2689 - accuracy
    ↪ : 0.2075
```

5.5.3.8.7 Test

```
StopWatch.start("evaluate")
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
StopWatch.stop("evaluate")

StopWatch.benchmark()

79/79 [==============================] - 22s 260ms/step - loss: 1.9646 - accuracy:
    ↪ 0.3505

Test accuracy: 35.0%

+------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
| Name             | Status   |    Time |     Sum | Start               | tag   |
    ↪ Node         | User   | OS    | Version                         |
|------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
| data-load        | failed   |   0.354 |   0.967 | 2021-02-18 15:27:21 |       |
    ↪ 351ef0f61c92 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| data-pre-process | failed   |   0.098 |   0.198 | 2021-02-18 15:27:21 |       |
    ↪ 351ef0f61c92 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| compile          | failed   |   0.932 |   2.352 | 2021-02-18 15:27:23 |       |
    ↪ 351ef0f61c92 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| train            | failed   | 377.842 | 377.842 | 2021-02-18 15:27:26 |       |
    ↪ 351ef0f61c92 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| evaluate         | failed   |  21.689 |  21.689 | 2021-02-18 15:33:44 |       |
    ↪ 351ef0f61c92 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
+------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
```

5.5.3.8.8 References Orignal Source to Source Code

5.5.3.9 MNIST-MLP Classification on Google Colab MNIST-MLP Classification on Google Colab

- Open In Colab
- Open in GitHub
- Download

In this lesson we discuss in how to create a simple IPython Notebook to solve an image classification problem with Multi Layer Perceptron.

5.5.3.9.1 Pre-requisites Install the following Python packages

1. cloudmesh-installer
2. cloudmesh-common

```
pip3 install cloudmesh-installer
pip3 install cloudmesh-common
```

5.5.3.9.2 Sample MLP with Tensorflow Keras

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import time

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.utils import to_categorical, plot_model
from keras.datasets import mnist
#import pydotplus
from keras.utils.vis_utils import model_to_dot
#from keras.utils.vis_utils import pydot


from cloudmesh.common.StopWatch import StopWatch

StopWatch.start("data-load")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
StopWatch.stop("data-load")
```

```
num_labels = len(np.unique(y_train))


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


image_size = x_train.shape[1]
input_size = image_size * image_size


x_train = np.reshape(x_train, [-1, input_size])
x_train = x_train.astype('float32') / 255
x_test = np.reshape(x_test, [-1, input_size])
x_test = x_test.astype('float32') / 255

batch_size = 128
hidden_units = 512
dropout = 0.45

model = Sequential()
model.add(Dense(hidden_units, input_dim=input_size))
model.add(Activation('relu'))
model.add(Dropout(dropout))
model.add(Dense(hidden_units))
model.add(Activation('relu'))
model.add(Dropout(dropout))
model.add(Dense(hidden_units))
model.add(Activation('relu'))
model.add(Dropout(dropout))
model.add(Dense(hidden_units))
model.add(Activation('relu'))
model.add(Dropout(dropout))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='mlp-mnist.png', show_shapes=True)

StopWatch.start("compile")
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
StopWatch.stop("compile")
StopWatch.start("train")
model.fit(x_train, y_train, epochs=5, batch_size=batch_size)
StopWatch.stop("train")
```

```
StopWatch.start("test")
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
StopWatch.stop("test")

StopWatch.benchmark()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/
    ↪ mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               401920
_________________________________________________________________
activation (Activation)      (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_3 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_4 (Activation)    (None, 10)                0
=================================================================
Total params: 1,195,018
Trainable params: 1,195,018
Non-trainable params: 0
```

```
_________________________________________________________________
Epoch 1/5
469/469 [==============================] - 14s 29ms/step - loss: 0.7886 - accuracy:
    ↪ 0.7334
Epoch 2/5
469/469 [==============================] - 14s 29ms/step - loss: 0.1981 - accuracy:
    ↪ 0.9433
Epoch 3/5
469/469 [==============================] - 14s 29ms/step - loss: 0.1546 - accuracy:
    ↪ 0.9572
Epoch 4/5
469/469 [==============================] - 14s 29ms/step - loss: 0.1302 - accuracy:
    ↪ 0.9641
Epoch 5/5
469/469 [==============================] - 14s 29ms/step - loss: 0.1168 - accuracy:
    ↪ 0.9663
79/79 [==============================] - 1s 9ms/step - loss: 0.0785 - accuracy:
    ↪ 0.9765

Test accuracy: 97.6%

+-----------+----------+--------+--------+---------------------+-------+--------------+--------+--
    ↪
| Name      | Status   |   Time |    Sum | Start               | tag   | Node
    ↪         | User   | OS    | Version                          |
|-----------+----------+--------+--------+---------------------+-------+--------------+--------+--
    ↪
| data-load | failed   |  0.549 |  0.549 | 2021-02-15 15:24:00 |       | 6609095905
    ↪ d1 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| compile   | failed   |  0.023 |  0.023 | 2021-02-15 15:24:01 |       | 6609095905
    ↪ d1 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| train     | failed   | 69.1   | 69.1   | 2021-02-15 15:24:01 |       | 6609095905
    ↪ d1 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| test      | failed   |  0.907 |  0.907 | 2021-02-15 15:25:10 |       | 6609095905
    ↪ d1 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
+-----------+----------+--------+--------+---------------------+-------+--------------+--------+--
    ↪
```

5.5.3.9.3 References Orignal Source to Source Code

5.5.3.10 MNIST-RMM Classification on Google Colab MNIST with Recurrent Neural Networks: Classification on Google Colab

- Open In Colab

- Open in GitHub
- Download

5.5.3.10.1 Prerequisites

Install the following packages

```
! pip3 install cloudmesh-installer
! pip3 install cloudmesh-common
```

5.5.3.10.2 Import Libraries

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, SimpleRNN
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.datasets import mnist
from cloudmesh.common.StopWatch import StopWatch
```

5.5.3.10.3 Download Data and Pre-Process

```
StopWatch.start("data-load")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
StopWatch.stop("data-load")


StopWatch.start("data-pre-process")
num_labels = len(np.unique(y_train))


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)


image_size = x_train.shape[1]
x_train = np.reshape(x_train,[-1, image_size, image_size])
x_test = np.reshape(x_test,[-1, image_size, image_size])
```

```
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
StopWatch.stop("data-pre-process")

input_shape = (image_size, image_size)
batch_size = 128
units = 256
dropout = 0.2
```

5.5.3.10.4 Define Model

```
StopWatch.start("compile")
model = Sequential()
model.add(SimpleRNN(units=units,
                    dropout=dropout,
                    input_shape=input_shape, return_sequences=True))
model.add(SimpleRNN(units=units,
                    dropout=dropout,
                    return_sequences=True))
model.add(SimpleRNN(units=units,
                    dropout=dropout,
                    return_sequences=False))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
plot_model(model, to_file='rnn-mnist.png', show_shapes=True)


model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
StopWatch.stop("compile")

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 28, 256)           72960
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 28, 256)           131328
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 256)               131328
_________________________________________________________________
dense (Dense)                (None, 10)                2570
```

```
_________________________________________________________________
activation (Activation)      (None, 10)                0
=================================================================
Total params: 338,186
Trainable params: 338,186
Non-trainable params: 0
```

5.5.3.10.5 Train

```
StopWatch.start("train")
model.fit(x_train, y_train, epochs=1, batch_size=batch_size)
StopWatch.stop("train")

469/469 [==============================] - 125s 266ms/step - loss: 0.6794 - accuracy
    ↪ : 0.7783

<tensorflow.python.keras.callbacks.History at 0x7f35d4b104e0>
```

5.5.3.10.6 Test

```
StopWatch.start("evaluate")
loss, acc = model.evaluate(x_test, y_test, batch_size=batch_size)
print("\nTest accuracy: %.1f%%" % (100.0 * acc))
StopWatch.stop("evaluate")

StopWatch.benchmark()

79/79 [==============================] - 7s 80ms/step - loss: 0.2581 - accuracy:
    ↪ 0.9181

Test accuracy: 91.8%

+------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
| Name             | Status   |    Time |     Sum | Start               | tag   |
    ↪ Node           | User   | OS    | Version                         |
|------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
| data-load        | failed   |   0.36  |   0.36  | 2021-02-18 15:16:12 |       | 8
    ↪ f16b3b1f784 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| data-pre-process | failed   |   0.086 |   0.086 | 2021-02-18 15:16:12 |       | 8
    ↪ f16b3b1f784 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
```

```
| compile          | failed   |   0.51  |   0.51  | 2021-02-18 15:16:12 |       | 8
    ↪ f16b3b1f784 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| train            | failed   | 126.612 | 126.612 | 2021-02-18 15:16:13 |       | 8
    ↪ f16b3b1f784 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
| evaluate         | failed   |   6.798 |   6.798 | 2021-02-18 15:18:19 |       | 8
    ↪ f16b3b1f784 | collab | Linux | #1 SMP Thu Jul 23 08:00:38 PDT 2020 |
+------------------+----------+---------+---------+---------------------+-------+--------------+--
    ↪
```

5.5.3.10.7 References

Orignal Source to Source Code

5.6 NLP

5.6.1 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

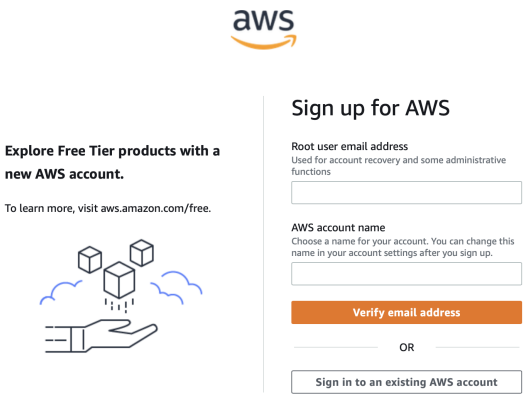
In this documentation we provide a summary of steps that give you quickly access to the service.

5.6.1.1 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:

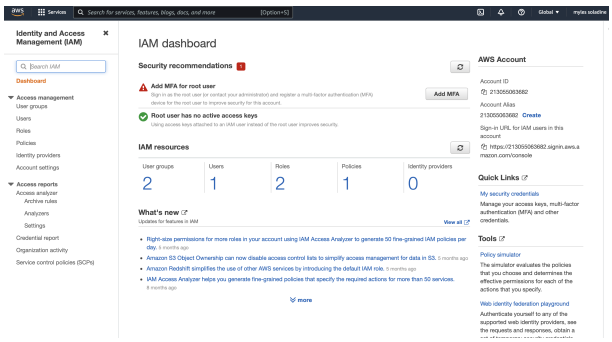
- https://portal.aws.amazon.com/billing/signup#/start/email

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`:



5.6.1.1.1 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



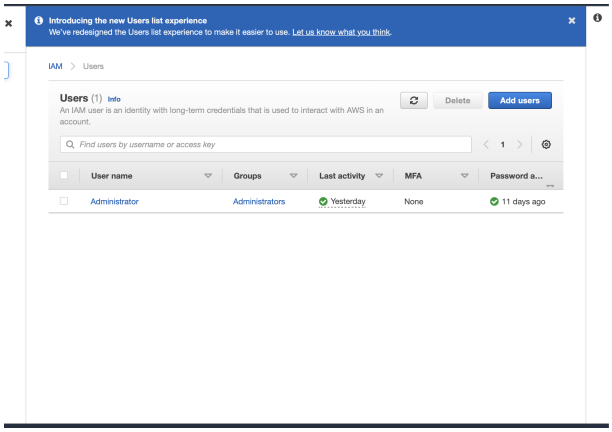
Here you can access secret keys and set permissions. At the top of this page search for text translate.

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

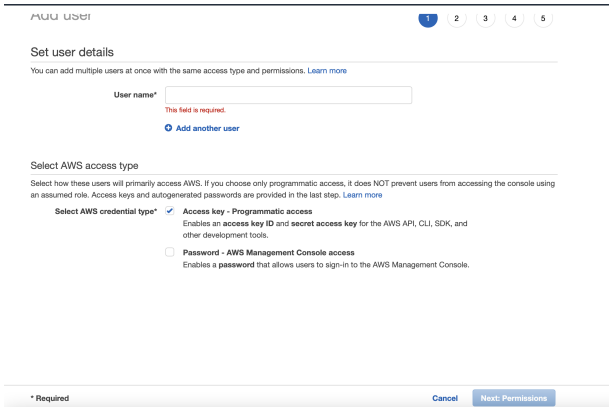
After clicking users you can create a user to run credentials.

From here click add user.



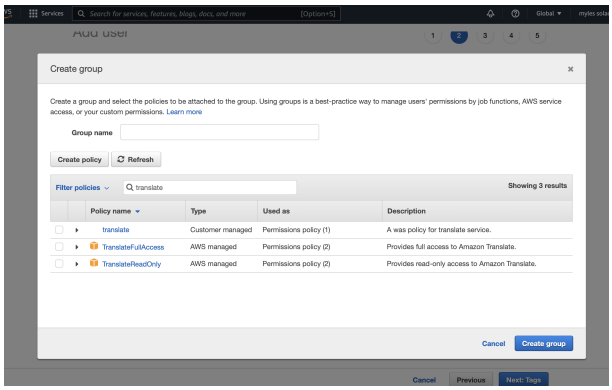
You will be required to add a name.

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



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` .

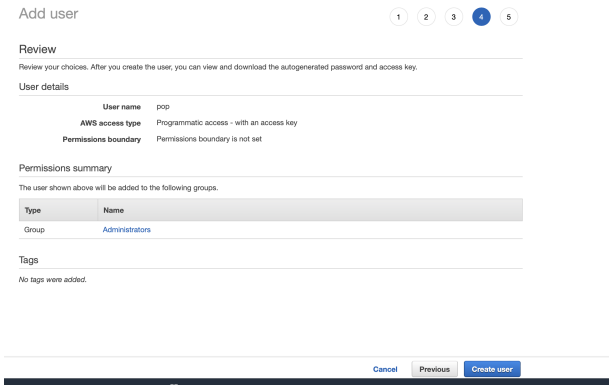
You can then click create group.



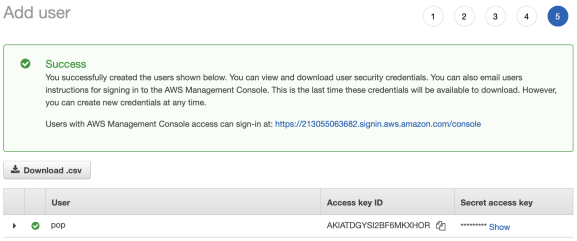
Next step is tags. this step is meant for users that want to give optional tags to their project.

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.



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



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 enviorment

```
$ 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`:

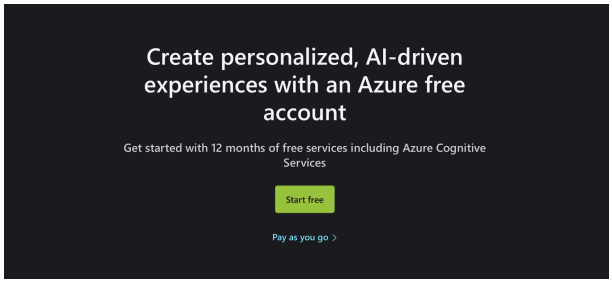
From here you can get started working in the command line found here: CLI Start

5.6.2 Azure Translate

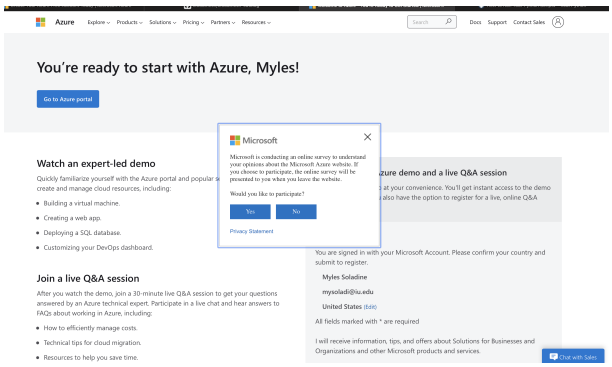
Here are steps to get started with the azure translate example in cloudmesh nlp.

First, navigate tot his link to sign uo with an azure account: https://azure.microsoft.com/en-us/free/cognitive-services/

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

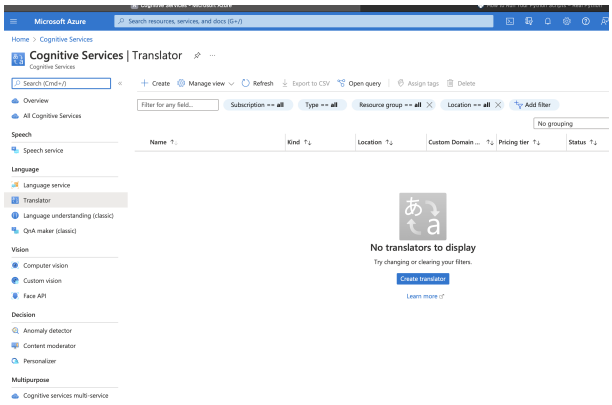


furthermore, after setting up an account with an email and billing you will be prompted to the console screen.



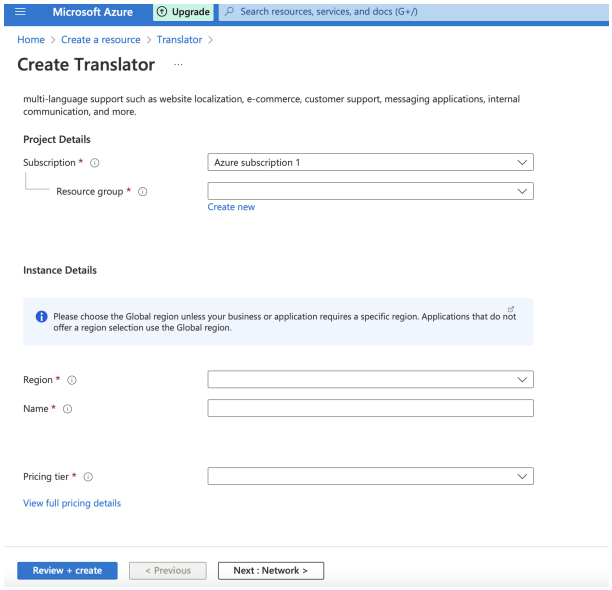
from here, you will click the top left link to go to the console.

You can either type translators in the search bar or follow this link: https://portal.azure.com/#blade/Microsoft_Azure_Pro

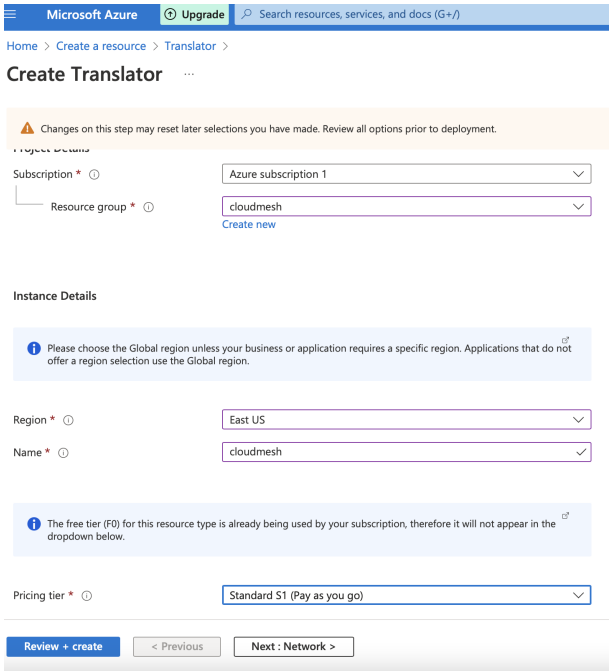


From here you will click the create translator button.

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

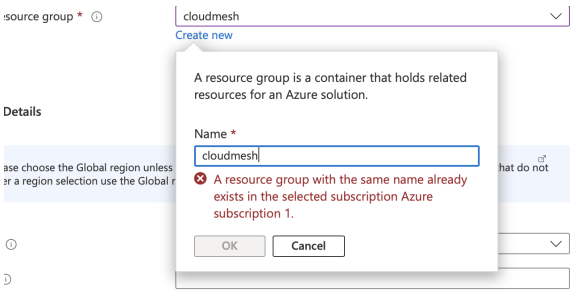


This what the form should look like filled out with the proper applied information.

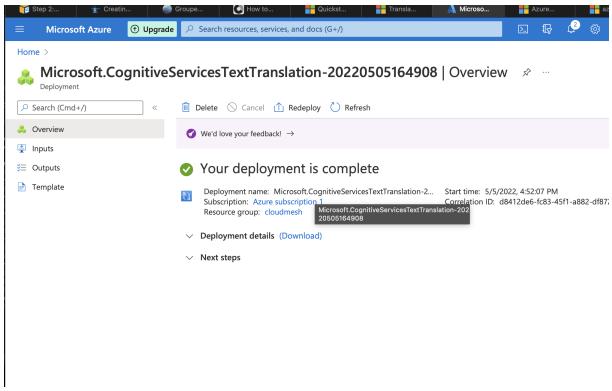


Resource groups are mostly irrelevant they are for bigger scaled projects where multiple people are working. Title this something for the project since it is required for an endpoint.

After this scroll down and hit ‘Create’



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



5.6.2.1 Installing and Starting Azure Translate through the command line

5.6.2.1.1 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
```

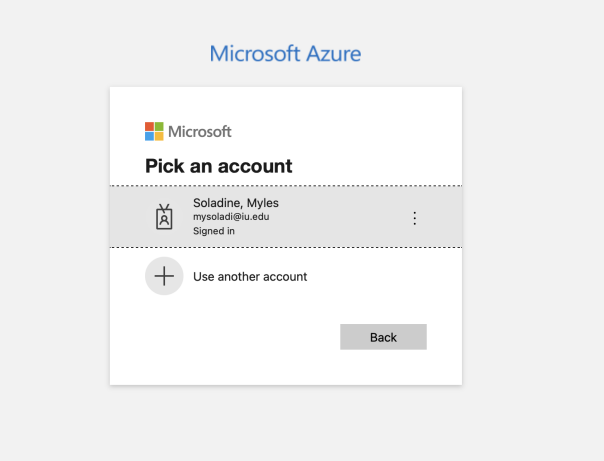
This installs the azure command line interface in the current directory.

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.

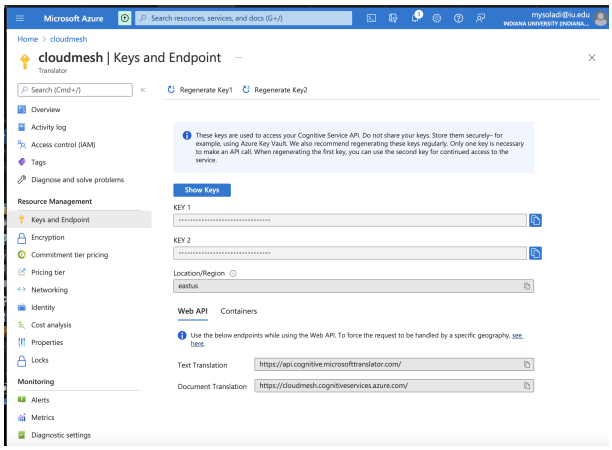
information about your account will be displayed on the command line.

```
[
  {
    "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).



Region is also highlighted in this form.

This will return

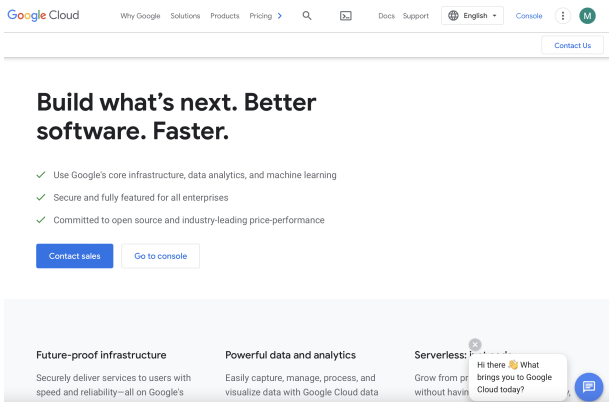
```
[{"detectedLanguage":{"language":"en","score":1.0},"translations":[{"text":"Hallo
    ↪ Welt","to":"de"}]}]%
```

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.

5.6.3 Google Translation Service

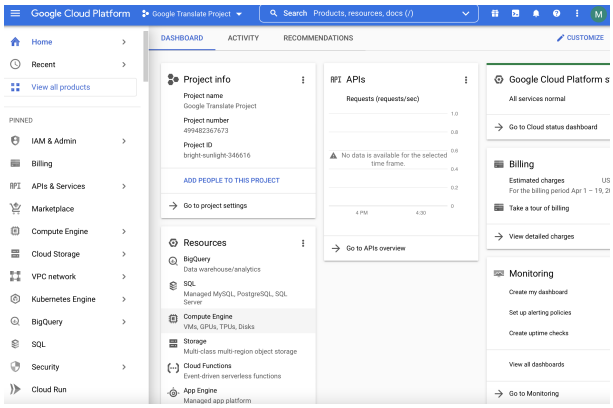
In order to get started using google translate there are steps for setup. Navigate to the link

- https://cloud.google.com



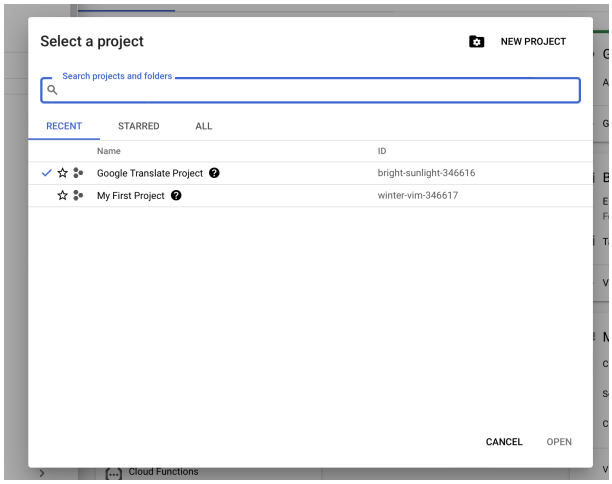
This is the homepage for Google cloud. Will need to activate your console with a Google account and billing.

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.

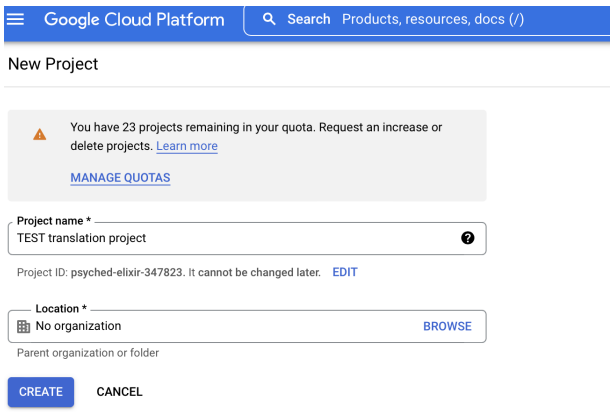


Here is the console for all google projects. The next step is to make a new project by selecting the dropdown at the top left.

The project creation page will look like this:



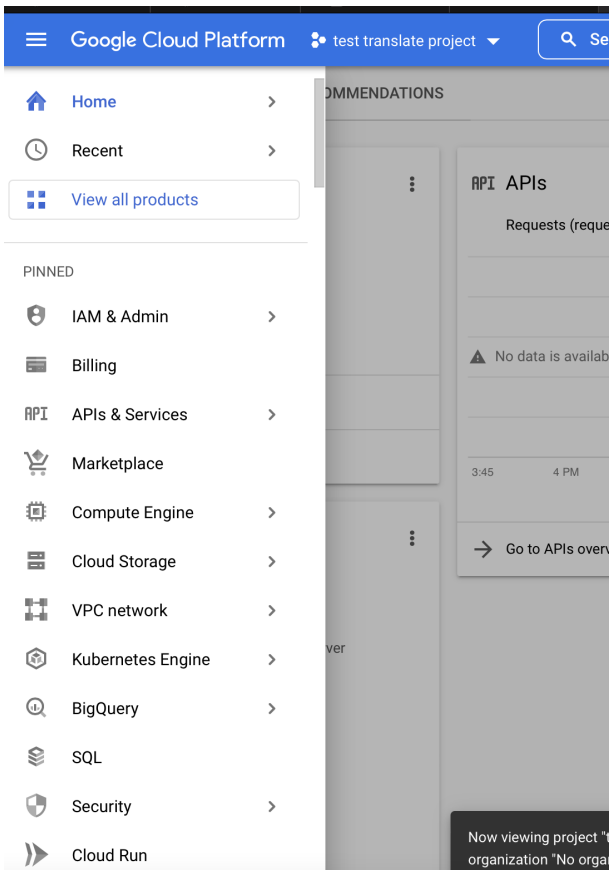
Then, click create new project.



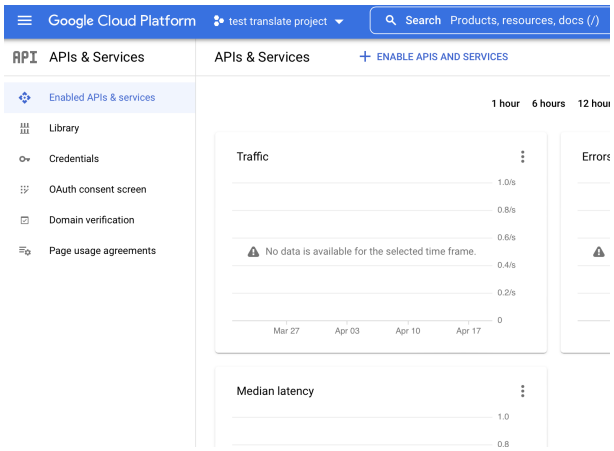
Give your project a title.

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

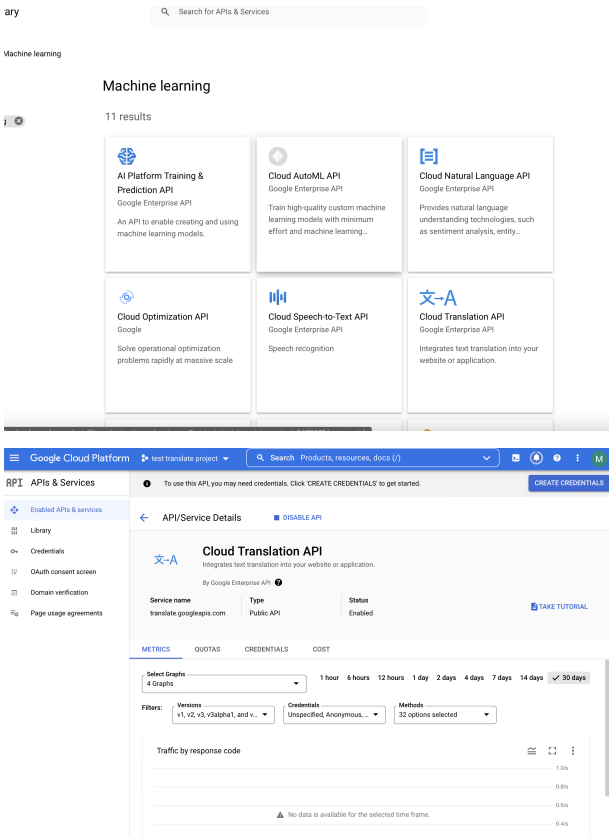
From here we need to activate the api for language translation. Select the api tab in the sidebar to the left.



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



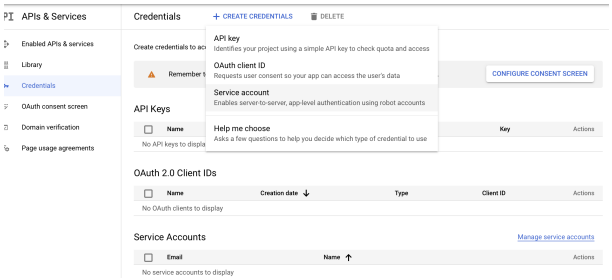
Scroll down to machine learning. Here you will see a cloud translation api. Click the api and enable it.



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

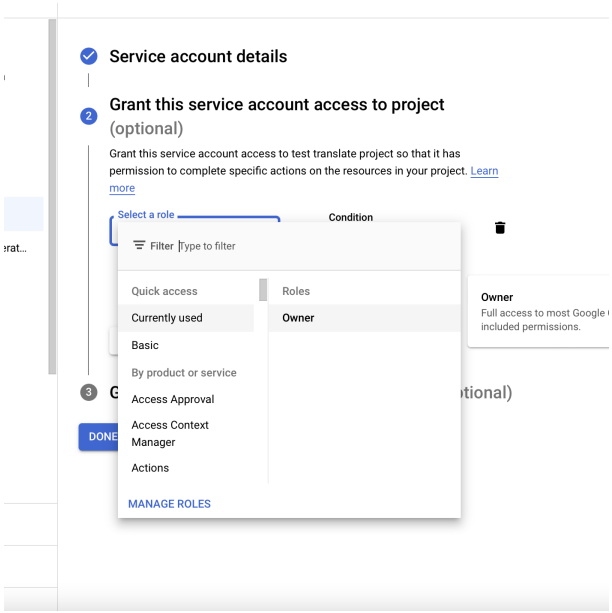
For each project you will have to enable credentials. this is a vital part to the continuation of the project.

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

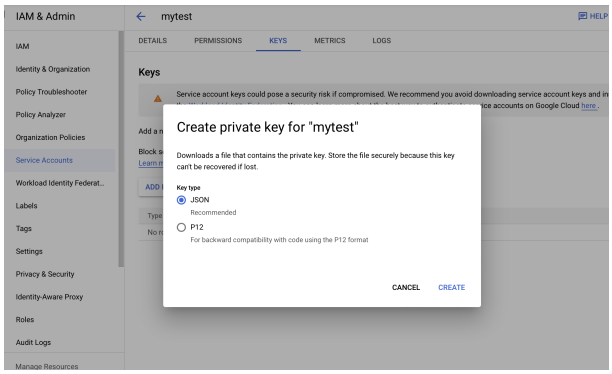


The next step is to title your service account. You will proceed and click the role owner.

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.



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 enviroment 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 requirments.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.

5.6.4 Implementation of A Hybrid Cloud natural Language Example

5.6.4.1 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
```

5.6.4.2 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
```

5.6.4.3 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

Provider	Google	aws
English, Australian	en-AU	en-AU
English, British	en-GB	en-GB
English, India	en-IN	en-IN
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

Provider	Google	aws
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

5.7 TIMESERIES

5.7.1 DL Timeseries

Learning Objectives

- Learn how to use deep learning for time series analysis

---

create and document a 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.

6 MNIST

tensorflow: M1

```
python -m pip install tensorflow-macos
```

tensorflow mac

```
python -m pip install tensorflow
```

see if previously also works on inten macs

6.1 Setting Parameters while Running `mnist` Jobs

6.1.1 Having Cloudmesh Installed

First, make sure Cloudmesh is installed properly on your local device. This can be done with the following command:

```
$ pip install cloudmesh-installer
$ mkdir cm
$ cd cm
$ cloudmesh-installer --ssh get cc
```

6.1.2 Installing the `reu2022` folder

Next, make sure the `reu2022` folder is installed. While you're in the `cm` folder, do the following commands to install the folder and be directed to the `mnist` folder.

```
$ git clone https://github.com/cybertraining-dsc/reu2022.git
$ cd reu2022/code/deeplearning/mnist
```

6.1.3 Setting the `user`, `host` parameters

Now, in order to run these jobs, the following parameters must be set: `user`, `host`, `cpu`, `gpu`, and `device`. As a disclaimer, no special characters are allowed while naming these parameters. Specifically no dashes (`-`) are allowed either. These can be replaced with underscores (`_`) or periods (`.`).

Setting the `user` and `host` parameters is relatively easy. Make sure you're in the `mnist` in `reu2022`. To set the `user` parameter, type in the following:

```
$ cms set user='user'
```

`'user'` can be what you want to call yourself as long it doesn't reveal your real identity. You can use your first name, a nickname, a random keyword, etc. For example:

```
$ cms set user='alex'
```

To set the `host` parameter, type in the following:

```
$ cms set host='host'
```

`'host'` is used to specify what type on device you're using. You can specify whether you're using a laptop or desktop and what operating system you're running on. For example:

```
$ cms set host='win11_laptop'
```

6.1.4 Setting the cpu and gpu parameters

When you run these `mnist` jobs, you'll be using either your CPU or GPU to run them. However, you'll have to input these parameters, you'll need to know what model you CPU or GPU is. This can be done with the following command:

```
$ python getinfo.py
```

The following output is produced on this specific device:

```
CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
GPU: none
```

After you receive these results, you want to shorten your CPU/GPU names to a single term with no spaces, dashes, nor special characters. In this case, the CPU name will be shortened to `i7_1165G7`.

If your device doesn't have a GPU, set device to `'cpu'` with the command:

```
$ cms set device='cpu'
```

If your device does have a GPU, set device to `'gpu'` with the command:

```
$ cms set device='gpu'
```

6.2 Setting Parameters while Running `mnist` Jobs

6.2.1 Table of Contents

- Rivanna
- Running Tensorflow on Rivanna
- Run Python MPI Programs on Rivanna
- Setting Parameters while Running `mnist` Jobs

Learning Objectives

- Learn how to set user parameters using `cms`
- Learn how to properly name the `user`, `host`, `cpu`, `gpu`, and `device` parameters
- Learn how run the `mnist` files and view the status of their progress

---

6.2.2 Having Cloudmesh Installed

First, make sure Cloudmesh is installed properly on your local device. This can be done with the following command:

```
$ pip install cloudmesh-installer
$ mkdir cm
$ cd cm
$ cloudmesh-installer --ssh get cc
```

6.2.3 Installing the `reu2022` folder

Next, make sure the `reu2022` folder is installed. While you're in the `cm` folder, do the following commands to install the folder and be directed to the `mnist` folder.

```
$ git clone https://github.com/cybertraining-dsc/reu2022.git
$ cd reu2022/code/deeplearning/mnist
```

6.2.4 Setting the `user`, `host` parameters

Now, in order to run these jobs, the following parameters must be set: `user`, `host`, `cpu`, `gpu`, and `device`. As a disclaimer, no special characters are allowed while naming these parameters. Specifically no dashes (`-`) are allowed either. These can be replaced with underscores (`_`) or periods (`.`).

Setting the `user` and `host` parameters is relatively easy. Make sure you're in the `mnist` in `reu2022`. To set the `user` parameter, type in the following:

```
$ cms set user='user'
```

`'user'` can be what you want to call yourself as long it doesn't reveal your real identity. You can use your first name, a nickname, a random keyword, etc. For example:

```
$ cms set user='alex'
```

To set the `host` parameter, type in the following:

```
$ cms set host='host'
```

`'host'` is used to specify what type on device you're using. You can specify whether you're using a laptop or desktop and what operating system you're running on. For example:

```
$ cms set host='win11_laptop'
```

6.2.5 Setting the `cpu` and `gpu` parameters

When you run these `mnist` jobs, you'll be using either your CPU or GPU to run them. However, you'll have to input these parameters, you'll need to know what model you CPU or GPU is. This can be done with the following command:

```
$ python getinfo.py
```

The following output is produced on this specific device:

```
CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
GPU: none
```

After you receive these results, you want to shorten your CPU/GPU names to a single term with no spaces, dashes, nor special characters. In this case, the CPU name will be shortened to `i7_1165G7`.

If your device doesn't have a GPU, set device to `'cpu'` with the command:

```
$ cms set device='cpu'
```

If your device does have a GPU, set device to `'gpu'` with the command:

```
$ cms set device='gpu'
```

6.2.6 Running the `mnist` Files

In the `mnist` folder located in the `reu2022` folder, there are various `mnist` files that can be run. Each test comes in three file formats, which are in `.ipynb`, `py`, and `.sh` files. On Rivanna, you want to run the `.sh`. The following example will run `mlp_mnist.sh`.

```
rivanna $ sbatch mlp_mnist.sh
```

This will submit a Slurm job to run the `.sh` file and which will be put in the queue that can be viewed using the following command:

```
rivanna $ squeue -u $USER
```

6.3 MNIST tensorflow rivanna

Use gregors MNIST program and run on rivanna, improve documentation and make sure everyone can reproduce it

If you have a GPU on your machine also run it there.

Compare with everyone the runtime, and use python graphics to create the image

6.4 Run Python MPI programs on Rivanna

see the book Python MPI

add chapter if not there

6.5 MNIST pyTorch Rivanna

Find a mnist pytorch implementation, make it runnable on rivanna, compare runtime to tensorflow

run it on your local machine

7 BIOS

7.1 Bios

In this section we provide a number of bios from a subset of students that have contributed to this effort in IEEE format.

7.1.1 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.

Alex worked in research during the summer of 2022 at the UVA Biocomplexity Institute's Computing for Global Challenges program under Dr. Gregor von Laszewski where he gained experience in programming and data science. Prior to that, he had 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.

7.1.2 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 participated in the Biocomplexity Institute's C4GC REU program. Her interests include computer architecture, machine learning, and quantum computing alongside quantum mechanics.

7.1.3 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 worked with 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.

7.1.4 Jacques Fleischer

Figure 26: Jacques Fleischer

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.

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 to find high performance computing solutions using Raspberry Pis.

7.1.5 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.

7.1.6 Abdulbaqiy Diyaolu

AbdulBaqiy Diyaolu is an undergraduate Computer Science and Mathematics Major from Mississippi Valley State University. AbdulBaqiy is currently a presidential scholar at such university.

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 partially participated in the Biocomplexity Research Program at UVA where he used his skills in support of different researches.

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.

7.1.7 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.

7.1.8 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.

7.1.9 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.

8 REFERENCES

[1] "Slurm Tutorial." Web Page, Jul-2022 [Online]. Available: https://docs.google.com/presentation/d/1Xt4kOtQpvl1JTDETJkOS-OMi8csZsJJw03xGus3QLA0/edit#slide=id.gd2ae9a35bb_0_0

[2] U. R. Computing, "Introduction to Rivanna at the RC Learning Portal." Web Page, Jun-2022 [Online]. Available: https://learning.rc.virginia.edu/notes/rivanna-intro

[3] Linuxize, "Using the SSH Config File." Web Page; Linuxize, Feb-2021 [Online]. Available: https://linuxize.com/post/using-the-ssh-config-file

[4] "Get Docker from the Docker documentation." Web Page, Jul-2022 [Online]. Available: https://docs.docker.com/get-docker

[5] "Orientation and setup." Web Page, Jul-2022 [Online]. Available: https://docs.docker.com/get-started

[6] P. Srivastav, "A Docker Tutorial for Beginners." Web Page, Jan-2016 [Online]. Available: https://docker-curriculum.com

[7] M. Galarnyk, "Python Lists and List Manipulation." Web Page at Towards Data Science; Towards Data Science, Nov-2019 [Online]. Available: https://towardsdatascience.com/python-basics-6-lists-and-list-manipulation-a56be62b1f95

[8] "Python List Array Methods." Web Page, Jul-2022 [Online]. Available: https://www.w3schools.com/python/python_ref_list.asp

[9] "Dictionary Manipulation in Python." Web Page at Python For Beginners, Aug-2020 [Online]. Available: https://www.pythonforbeginners.com/dictionary/dictionary-manipulation-in-python

[10] "Python Dictionary Methods." Web Page, Jul-2022 [Online]. Available: https://www.w3schools.com/python/python_ref_dictionary.asp

[11] "Python Dictionary update() Method." Web Page, Jul-2022 [Online]. Available: https://www.w3schools.com/python/ref_dictionary_update.asp

[12] "Creating a dataframe using CSV files." Web Page at Geeks for Geeks, Feb-2022 [Online]. Available: https://www.geeksforgeeks.org/creating-a-dataframe-using-csv-files

[13] "CSV File Reading and Writing." Web Page, Jul-2022 [Online]. Available: https://docs.python.org/3/library/csv.html#examples

[14] "CSV Files Overview." Web Page, Jul-2016 [Online]. Available: https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html

[15] "Built-in Functions." Web Page, Jul-2022 [Online]. Available: https://docs.python.org/3/library/functions.html#open

[16] "Python CSV File Reading and Writing." Web Page at ProTech, Jul-2022 [Online]. Available: https://www.protechtraining.com/blog/post/python-for-beginners-reading-manipulating-csv-files-737#extracting-information-from-a-csv-file

[17] "List Index Out of Range." Web Page at Stack Overflow, Jul-2022 [Online]. Available: https://stackoverflow.com/questions/13039392/csv-list-index-out-of-range

[18] "Visualize data from CSV file in Python." Web Page Geeks for Geeks, Mar-2021 [Online]. Available: https://www.geeksforgeeks.org/visualize-data-from-csv-file-in-python

[19] "glob Filename Pattern Matching PyMOTW 3." Web Page, Oct-2021 [Online]. Available: https://pymotw.com/3/glob/index.html

[20] "mmap Memory-map Files PyMOTW 3." Web Page, Oct-2021 [Online]. Available: https://pymotw.com/3/mmap/index.html

[21] "pickle Object Serialization PyMOTW 3." Web Page, Oct-2021 [Online]. Available: https://pymotw.com/3/pickle/index.html

[22] “shelve Persistent Storage of Objects — PyMOTW 3.” Web Page, Oct-2021 [Online]. Available: https://pymotw.com/3/shelve/index.html

[23] “First Steps - FastAPI.” Web Page, Jul-2022 [Online]. Available: https://fastapi.tiangolo.com/tutorial/first-steps

[24] “Path Parameters - FastAPI.” Web Page, Jul-2022 [Online]. Available: https://fastapi.tiangolo.com/tutorial/path-params

[25] “FastAPI in Containers - Docker - FastAPI.” Web Page, Jul-2022 [Online]. Available: https://fastapi.tiangolo.com/deployment/docker

[26] “Testing - FastAPI.” Web Page, Jul-2022 [Online]. Available: https://fastapi.tiangolo.com/tutorial/testing/#using-testclient

[27] “Async Tests - FastAPI.” Web Page, Jul-2022 [Online]. Available: https://fastapi.tiangolo.com/advanced/async-tests

[28] “A Thread-Safe FIFO Implementation of a Pythin Queue.” Web Page, Oct-2021 [Online]. Available: https://pymotw.com/3/queue/index.html

[29] “Priority Queue in Python.” Web Page, May-2022 [Online]. Available: https://www.geeksforgeeks.org/priority-queue-in-python

[30] “Matplotlib Visualization with Python.” Web Page, Jul-2022 [Online]. Available: https://matplotlib.org

[31] “Matplotlib 3.5.2 Documentation.” Web Page, Jun-2022 [Online]. Available: https://matplotlib.org/stable/api/pyplot_summary.html

[32] “What Is Matplotlib In Python? How to use it for plotting?” Web Page at Active State, Jul-2022 [Online]. Available: https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting

[33] “Bar Plot in Matplotlib.” Web Page at Geeks for Geeks, Mar-2021 [Online]. Available: https://www.geeksforgeeks.org/bar-plot-in-matplotlib

[34] “API reference seaborn 0.11.2 documentation.” Web Page, Jul-2022 [Online]. Available: https://seaborn.pydata.org/api.html

[35] “Python Seaborn Tutorial - GeeksforGeeks.” GeeksforGeeks, Mar-2022 [Online]. Available: https://www.geeksforgeeks.org/python-seaborn-tutorial

[36] "Introduction to Seaborn - Python - GeeksforGeeks." GeeksforGeeks, Jun-2020 [Online]. Available: https://www.geeksforgeeks.org/introduction-to-seaborn-python

[37] "Plotting graph using Seaborn | Python - GeeksforGeeks." GeeksforGeeks, Jul-2022 [Online]. Available: https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python

[38] "Seaborn load_dataset." Stack Overflow, Jul-2022 [Online]. Available: https://stackoverflow.com/questions/30336324/seaborn-load-dataset

[39] mwaskom, "seaborn-data." GitHub, Jul-2022 [Online]. Available: https://github.com/mwaskom/seaborn-data/blob/master/planets.csv

[40] "Bokeh." Web Page, May-2022 [Online]. Available: http://docs.bokeh.org

[41] "Plotting with basic glyphs." Web Page, May-2022 [Online]. Available: http://docs.bokeh.org/en/latest/docs/user_guide/plotting.html

[42] "Bokeh documentation." Web Page, May-2022 [Online]. Available: http://docs.bokeh.org/en/latest

[43] "Bokeh Figure Documentation." Web Page, May-2022 [Online]. Available: https://docs.bokeh.org/en/latest/docs/reference/plotting/figure.html

[44] "Bokeh Exporting Plots." Web Page, May-2022 [Online]. Available: https://docs.bokeh.org/en/latest/docs/user_guide/export.html

[45] "Pandas DataFrame Plot Bar." Web Page, Jun-2022 [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html

[46] "Pandas Dataframe Plot Line." Web Page, Jun-2022 [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html

[47] "Pandas Dataframe Plot Pie." Web Page, Jun-2022 [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html

[48] "TensorFlow Anaconda documentation." Web Page, Jul-2022 [Online]. Available: https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow

[49] "TensorFlow on Rivanna the University of Virginia Research Computing Portal." Web Page, Jul-2022 [Online]. Available: https://www.rc.virginia.edu/userinfo/rivanna/software/tensorflow

[50] "Software Containers the University of Virginia Research Computing Portal." Web Page, Jul-2022 [Online]. Available: https://www.rc.virginia.edu/userinfo/rivanna/software/containers