Summer 2025 CS4641/CS7641 A Homework 1 - Programming Section

Instructor: Dr. Max Mahdi Roozbahani, Dr. Nimisha Roy

Deadline: Friday, September 19th, 11:59 pm ET

- No unapproved extension of the deadline is allowed. For late submissions, please refer to the course website.
- Discussion is encouraged on Ed as part of the Q/A. We encourage whiteboard-level discussions about the homework. **However, all assignments should be done individually.**
- Plagiarism is a *serious offense*. **You are responsible for completing your own work.** You are not allowed to copy and paste, or paraphrase, or submit materials created or published by others, as if you created the materials. All materials submitted must be your own, and you must not collaborate with anyone or share your HW content except the ML instructional team.
- Working with a generative-AI platform *may constitute plagiarism*. In line with the Joyner Heuristic being used by many classes at GT, you should treat collaboration with generative-AI as collaboration with a knowledgeable peer. **Sharing the question or your work verbatim with an AI agent so as to generate an answer to the question is considered academic dishonesty and will be treated the same as any other incident of academic dishonesty.** If you find yourself turning to generative-AI for help answering a question, we suggest that you instead make use of Ed or TA office hours.
- Even using generative-AI for formatting your answers is *not permitted*. **While we understand that \LaTeX~has a learning curve, you may not use any generative-AI platform to improve your writing or LaTeX formatting.** These tools inherently produce output that may include a partial or complete solution to the question, including corrections to work you give it, even if purely prompted for syntactic use. Additionally, you have no custody over the data you give these platforms, which may leak or be used to inform subsequent training. Thus, while you are more than welcome to ask it about LaTeX commands and formatting tips, sharing the question or your answer to a question verbatim with an AI agent, even purely for syntactic use, is considered academic dishonesty and will be treated the same as any other incident of academic dishonesty. If you find yourself turning to generative-AI for help rewording your work to improve the language therein, remember that many of these questions will not be graded on language, but the content of your work. If you wish to improve it nonetheless, you can make use of the Georgia Tech Communications Lab or TA office hours. If you find yourself turning to generative-AI for help reformatting your LaTeX, we suggest that you instead use the resources in the instructions below or TA office hours. Ed is also an appropriate place to ask about LaTeX formatting, so long as your post doesn't reveal answers to a question (or make a private post if your question necessitates revealing answers).
- *All* incidents of suspected dishonesty, plagiarism, or violations of the Georgia Tech Honor Code will be subject to the institute's Academic Integrity procedures. **If we observe any (even small) similarities/plagiarisms detected by Gradescope or our TAs, we will directly report the case to OSI, which may, unfortunately, lead to a very harsh outcome, pending review. Consequences can be severe, including academic probation or dismissal, grade penalties, a 0 grade for assignments concerned, and prohibition from withdrawing from the class.**

Instructions for the assignment

- This assignment consists of warm-up programming questions designed to get you familiar with our programming homework structure.

Using the autograder

- Grads will typically find three assignments on Gradescope and Undergrads will typically find four assignments:
  - "Assignment X Non-programming": Where you will submit the written portion of the assignment.

    - "Assignment X Programming": Where you will submit any .py files and any program outputs as required by the problem.
    - "Assignment X Programming - Bonus for All": Where you will submit any .py files and any program outputs as required for Bonus for All.
    - "Assignment X Programming - Bonus for Undergrad": Where you will submit any .py files and any program outputs as required for Bonus for Undergrand. (Undergrad Only)
- You will submit your code for the autograder in the Assignment 1 Programming section.
- We provided you .py files and we added libraries in those files please DO NOT remove those lines and add your code after those lines. Note that these are the only allowed libraries that you can use for the homework.
- You are allowed to make as many submissions until the deadline as you like. Additionally, note that the autograder tests each function separately, therefore it can serve as a useful tool to help you debug your code if you are not sure of what part of your implementation might have an issue.

Deliverables and Points Distribution

Q7: Programming Warm-Up [5pts total]

Deliverables:

- `warmup.py`
- `env.pkl`

Parts:

- **Setup** [2pts] - *programming*
- **Numpy** [3pts] - *programming*
    - Numpy Basics [2pts]
    - Broadcasting [1pts]

7.1 Setup [2pts]

- **Deliverable:** `env.pkl`

This notebook is tested under python 3.11, and the corresponding packages can be downloaded from miniconda. You may also want to get yourself familiar with several packages:

- jupyter lab: provides a web-based IDE with a built-in debugging functionality for jupyter notebooks.
- numpy: a high performance math library backed by C
- matplotlib: a python plotting libaray

Other packages you may find indispensable in machine learning (and potentially your project) are:

- scikit-learn: provides many classical ML and data analysis algorithms
- pandas: provides many useful tools for organizing and manipulating data
- seaborn: make beatiful plots with less fidgeting in matplotlib
- plotly: another great data visualization package

Please implement the functions that have "raise NotImplementedError", and after you finish the coding, please delete or comment "raise NotImplementedError".

Before you run any jupyter notebook cells, please read environment_setup.md in the environment folder and follow the instructions to set up the environment.

In [1]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################

import sys

sys.path.append("./utilities/")
sys.path.append("warmup.py")

import numpy as np

print("Version information")

print("python: {}".format(sys.version))
print("numpy: {}".format(np.__version__))


%load_ext autoreload
%autoreload 2
```

```
Version information
python: 3.11.13 (main, Jun  5 2025, 13:12:00) [GCC 11.2.0]
numpy: 1.26.2
```

7.1.1 Basics, Imports, and Directories

For the following part, you will need to ensure your notebook runtime is started in the correct directory. You can verify this with the following cell.

In [2]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################

# RUN ME #
import os

os.getcwd()
```

Out[2]:
```
'/app/src/teacher_files'
```

In the cell below, import the `PackageUtils` class from `utils.py`.

In [3]:
```
# YOUR CODE HERE #
```

In the cell below, instantiate an instance of the `PackageUtils` class, then call the `get_packages` method upon that class instance to see what packages are installed in this notebook's runtime environment.

In [5]:
```
# YOUR CODE HERE #
```

In [6]:
```
```

```
anyio==4.1.0
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
```

```
astor==0.8.1
asttokens==2.4.1
async-lru==2.0.4
attrs==23.1.0
autoflake==2.2.1
autopep8==2.0.4
Babel==2.14.0
backports.tarfile==1.2.0
beautifulsoup4==4.12.2
black==23.12.0
bleach==6.1.0
Bottleneck @ file:///croot/bottleneck_1731058641041/work
Brotli @ file:///croot/brotli-split_1736182456865/work
brotlicffi @ file:///croot/brotlicffi_1736182461069/work
build==1.3.0
CacheControl==0.14.3
certifi==2023.11.17
cffi==1.16.0
cfgv==3.4.0
chardet==5.2.0
charset-normalizer==3.3.2
cleo==2.1.0
click==8.1.7
comm==0.2.0
contourpy @ file:///croot/contourpy_1732540045555/work
crashtest==0.4.1
cryptography==45.0.7
cycler @ file:///tmp/build/80754af9/cycler_1637851556182/work
debugpy==1.8.0
decorator==5.1.1
defusedxml==0.7.1
distlib==0.3.8
dulwich==0.22.8
executing==2.0.1
fastjsonschema==2.19.0
filelock==3.13.1
findpython==0.6.3
fonttools @ file:///croot/fonttools_1737039080035/work
fqdn==1.5.1
h11 @ file:///croot/h11_1748442006460/work
httpcore @ file:///croot/httpcore_1748526048470/work
httpx @ file:///croot/httpx_1746747840559/work
identify==2.5.33
idna==3.6
imageio @ file:///croot/imageio_1738159938990/work
importlib_metadata==8.7.0
installer==0.7.0
ipykernel==6.27.1
ipython==8.18.1
ipython_pygments_lexers @ file:///croot/ipython_pygments_lexers_1744753235686/work
ipywidgets==8.1.1
isoduration==20.11.0
isort==5.13.2
jaraco.classes==3.4.0
jaraco.context==6.0.1
jaraco.functools==4.3.0
jedi==0.19.1
jeepney==0.9.0
Jinja2==3.1.2
joblib @ file:///croot/joblib_1754310200728/work
```

```
json5==0.9.14
jsonpointer==2.4
jsonschema==4.20.0
jsonschema-specifications==2023.11.2
jupyter==1.0.0
jupyter-console==6.6.3
jupyter-events==0.9.0
jupyter-lsp==2.2.1
jupyter_client==8.6.0
jupyter_core==5.5.0
jupyter_server==2.12.1
jupyter_server_terminals==0.5.0
jupyterlab==4.0.9
jupyterlab-widgets==3.0.9
jupyterlab_pygments==0.3.0
jupyterlab_server==2.25.2
keyring==25.6.0
kiwisolver @ file:///croot/kiwisolver_1737039087198/work
lazy_loader @ file:///croot/lazy_loader_1718176737906/work
markdown-it-py==3.0.0
MarkupSafe==2.1.3
matplotlib==3.10.5
matplotlib-inline==0.1.6
mdurl==0.1.2
mistune==3.0.2
mkl-service==2.4.0
mkl_fft @ file:///io/mkl313/mkl_fft_1730824109137/work
mkl_random @ file:///io/mkl313/mkl_random_1730823916628/work
more-itertools==10.7.0
msgpack==1.1.1
mypy-extensions==1.0.0
nbclient==0.9.0
nbconvert==7.12.0
nbformat==5.9.2
nbqa==1.7.1
nest-asyncio==1.5.8
networkx @ file:///croot/networkx_1756709277056/work
nodeenv==1.8.0
notebook==7.0.6
notebook_shim==0.2.3
numexpr @ file:///croot/numexpr_1752521720723/work
numpy==1.26.2
overrides==7.4.0
packaging==23.2
pandas @ file:///croot/pandas_1756466404410/work/dist/pandas-2.3.2-cp311-cp311-linux_x86_64.whl#sha256=86a7c435b68fb37f2f2717c79ea6522702263461774ddcc8a0b0e
244eb3be162
pandocfilters==1.5.0
parso==0.8.3
pastel==0.2.1
pathspec==0.12.1
patsy @ file:///croot/patsy_1738159930729/work
pbs-installer==2025.8.28
pdf-watermark==2.0.0
pdfkit==1.0.0
pexpect==4.9.0
Pillow==10.1.0
pip @ file:///croot/pip_1756710021410/work
pkginfo==1.12.1.2
platformdirs==4.1.0
poethepoet==0.24.4
```

```
poetry==2.1.4
poetry-core==2.1.3
pre-commit==3.6.0
prometheus-client==0.19.0
prompt-toolkit==3.0.43
psutil==5.9.6
ptyprocess==0.7.0
pure-eval==0.2.2
pyclean==2.7.6
pycodestyle==2.11.1
pycparser==2.21
pyflakes==3.1.0
Pygments==2.17.2
pyparsing @ file:///croot/pyparsing_1731445506121/work
pypdf==3.17.2
pyproject_hooks==1.2.0
PyQt6==6.7.1
PyQt6_sip @ file:///croot/pyqt-split_1753427276959/work/pyqt_sip
PySocks @ file:///work/ci_py311/pysocks_1676822712504/work
python-dateutil==2.8.2
python-json-logger==2.0.7
pytz @ file:///croot/pytz_1752135852232/work
pyupgrade==3.15.0
PyYAML==6.0.1
pyzmq==25.1.2
qtconsole==5.5.1
QtPy==2.4.1
RapidFuzz==3.14.0
referencing==0.32.0
reportlab==4.0.8
requests==2.31.0
requests-toolbelt==1.0.0
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rich==13.7.1
rpds-py==0.13.2
scikit-image @ file:///croot/scikit-image_1750419472910/work
scikit-learn @ file:///croot/scikit-learn_1753427383173/work
scipy @ file:///croot/scipy_1753384405186/work/dist/scipy-1.16.0-cp311-cp311-linux_x86_64.whl#sha256=73630e3030511a4d41e5331c32fe4c71ed2f5470dc5a546981f7931
d7da292b9
seaborn @ file:///croot/seaborn_1749110291192/work
SecretStorage==3.3.3
Send2Trash==1.8.2
setuptools==69.0.2
shellingham==1.5.4
sip @ file:///croot/sip_1738856193618/work
six==1.16.0
sniffio==1.3.0
soupsieve==2.5
stack-data==0.6.3
statsmodels @ file:///croot/statsmodels_1753363813252/work
terminado==0.18.0
threadpoolctl @ file:///croot/threadpoolctl_1719407800858/work
tifffile @ file:///croot/tifffile_1741164537642/work
tinycss2==1.2.1
tokenize-rt==5.2.0
tomli==2.0.1
tomlkit==0.13.3
tornado==6.4
tqdm @ file:///croot/tqdm_1738943501192/work
```

```
traitlets==5.14.0
trove-classifiers==2025.8.26.11
tweet-preprocessor==0.6.0
types-python-dateutil==2.8.19.14
typing_extensions @ file:///croot/typing_extensions_1756280817316/work
tzdata @ file:///croot/python-tzdata_1746123641790/work
unicodedata2 @ file:///croot/unicodedata2_1736541023050/work
uri-template==1.3.0
urllib3==2.1.0
virtualenv==20.25.0
wcwidth==0.2.12
webcolors==1.13
webencodings==0.5.1
websocket-client==1.7.0
wheel==0.45.1
widgetsnbextension==4.0.9
zipp==3.23.0
zstandard==0.24.0
```

7.1.2 Local Testing & Debugging

Optional local tests using a small toy dataset are sometimes provided to aid in debugging. The local tests are all stored in localtests.py

The autograder is the final arbiter

- There are no points associated with passing or failing the local tests, you must still pass the autograder to get points.
- It is possible to fail the local test and pass the autograder.
  - The autograder may have tolerances to account for minor implementation differences.
  - The reverse is also true, as the autograder may cover a larger number of corner cases.
- **You do not need to pass both local and autograder tests to get points, passing the Gradescope autograder is sufficient for credit.**

Work smarter, not harder

- Read the stack trace carefully. Often it will tell you exactly what's wrong.
- Understand what the local-test is doing. That way you can develop your own tests.
- Grow beyond the print statement: embrace a debugger. Jupyter-lab has a built in debugger which allows you to look at data types, set breakpoints, and examine variables. If using a different IDE, look up your IDE's documentation on how to setup a proper debugger.
- Develop incrementally and test frequently, both localy and on Gradescope. Waiting to complete the whole class before testing can make it hard to isolate errors.

For this problem perform the following in the cell below:

- import `WarmupTests` from the `localtests.py`.
- Run the cell and submit `env.pkl`

In [7]:
```
import unittest

# import WarmupTests from the localtests.py in the utilities folder.

# END YOUR CODE ABOVE #
```

In [9]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################
result = unittest.main(
    argv=["ignored", "WarmupTests.test_get_packages"], verbosity=1, exit=False
```

```
)
if not result.result.wasSuccessful():
    sys.exit(1)
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Passed test_get_packages
```

7.2 Numpy Basics [2pts]

The following exercise will familiarize you with the basics of working with Numpy and navigating the numpy documentation

In `warmup.py` you will implement several "one-liners" using functions provided by numpy. No points will be awarded on Gradescope for any use of for loops or list comprehensions. Implement the following functions in `warmup.py`:

- indices_of_k
- argmax_1d
- mean_rows
- sum_squares

You may test your implementation with the below local tests. These local tests only checks the returned values of your implementation and does not check whether your implementation uses loops. Gradescope will check to make sure your implementation does not use loops (for, while, or list comprehensions).

WARNING: Make sure you match the dimensions of the output given in the comments of required function in warmup.py

HINT: Print and see what numpy functions are doing as much as you can!

In [10]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################
import unittest

result = unittest.main(argv=[""], verbosity=1, exit=False)

if not result.result.wasSuccessful():
    sys.exit(1)
```

```
.....
----------------------------------------------------------------------
Ran 5 tests in 0.005s

OK
Correct Values for argmax_1d
Passed test_get_packages
Correct Values for indices_of_k
Correct Values for mean_rows
Correct Values for sum_squares
```

7.3 Broadcasting [1pt]

One of the simplest and most common similarity metrics in ML is the Manhattan Distance or taxicab-distance. The function below takes two lists of N points in D dimensional space (NxD numpy arrays) and computes the Manhattan distance between every possible pair of points. *Hint: you can use this to try creating your own*

unittests

Unfortunately such an implementation is too slow for a large dataset. In `fast_manhattan` leverage the broadcasting properties of numpy to create a faster version in a single line.

In [11]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################

import numpy as np


def slow_manhattan(x, y):
    """
    Args:
        x: N x D numpy array
        y: M x D numpy array
     Return:
        dist: N x M numpy array, where dist[i, j] is the Manhattan distance between
        x[i, :] and y[j, :]
    """
    dist = np.empty((x.shape[0], y.shape[0]))
    for i in range(x.shape[0]):
        for j in range(y.shape[0]):
            d = 0
            for k in range(x.shape[1]):
                d += abs(x[i][k] - y[j][k])
            dist[i][j] = d
    return dist
```

Let's test the speed of this naive implementation:

In [12]:
```
%%timeit

##############################
### DO NOT CHANGE THIS CELL ###
##############################


x = np.random.rand(100, 3)
y = np.random.rand(100, 3)
d = slow_manhattan(x, y)
```

```
18.8 ms ± 318 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Compare this with the vectorized implementation:

In [13]:
```
##############################
### DO NOT CHANGE THIS CELL ###
##############################

import warmup
```

In [14]:
```
%%timeit

##############################
### DO NOT CHANGE THIS CELL ###
##############################
```

```
x = np.random.rand(100, 3)
y = np.random.rand(100, 3)
d = warmup.fast_manhattan(x, y)
```

```
286 µs ± 1.72 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

On the Overleaf LaTeX written portion of this assignment, you will be asked about the relative time and space complexities of the naive approach, `slow_manhattan`, and your code, `fast_manhattan`. You're encouraged to use this notebook to experiment with new data and reason carefully about the time and space complexities of these two algortihms.