MATH 257 Python Hands-On Session

CARE Tutoring

Please sign in to the queue →

Python Quiz 1 (Week 10)

Assessments	
Computational lab lesson	
CL1	Python tutorial
CL2	Working with Vectors
CL3	Matrix Operations
CL4	Slinky: solving linear system of equations
CL5	Intro to Graphs and application to social network
CL6	Discrete Cosine Transforms and Data Compression
CL7	Markov Chains
Computational lab HW	
CHW1	Intro to Python
CHW2	Working with Vectors
CHW3	Matrix operations
CHW4	Linear System of Equations
CHW5	Graphs and Algebraic Graph Theory
CHW6	Coordinate systems and data compression
CHW7	Markov chains

Will cover Labs and associated HWs Lab 1 through 7!

In-Person Resources

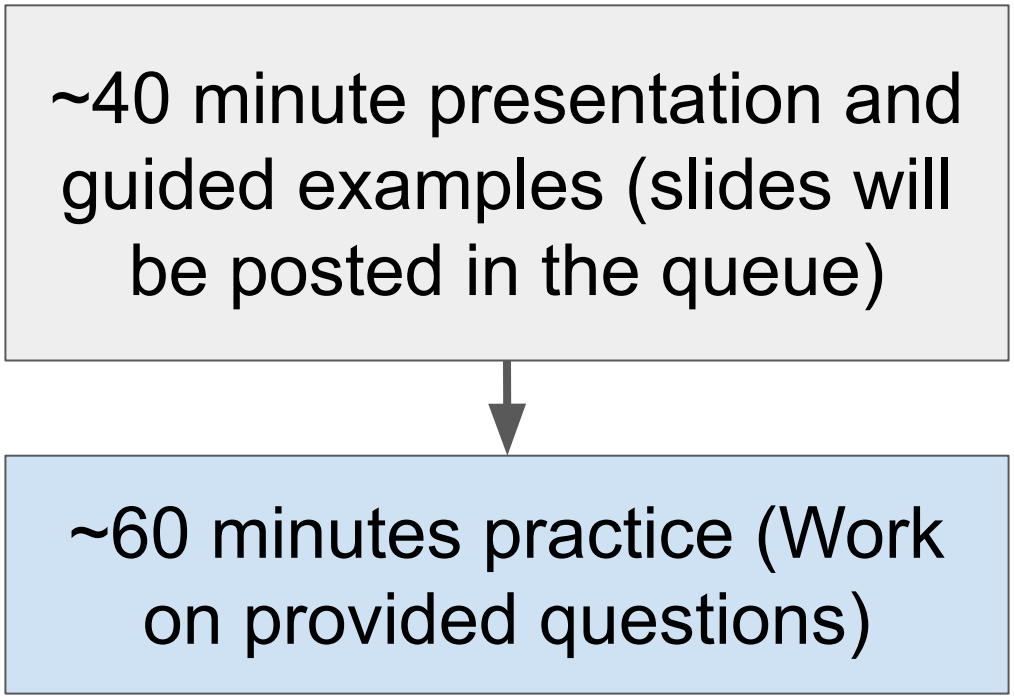
CARE Drop-in tutoring:
7 days a week on the 4th floor of Grainger Library and Zoom!
Sunday - Thursday 12pm-10pm
Friday & Saturday 12-6pm

Utilize CA office hours for MATH 257 posted on Canvas!

Skill	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Python	12pm-10pm	1pm-6pm 8pm-10pm	12pm-3pm 8pm-10pm	12pm-10pm	1pm-10pm	12pm-6pm	12pm-6pm

Subject	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Math 257	12pm-9pm	12pm-2pm 4pm-7pm 8pm-10pm	1pm-3pm 5pm-10pm	12pm-8pm	1pm-9pm	12pm-5pm	1pm-3pm 4pm-6pm

How will this tutoring session work?



This is not CS 101, it's a Linear Algebra class

CARE 1.1. Intro and Motivation

Why do we use Python in this class?

- Avoid tedious calculations
- Use larger matrices to process lots of data at once
- Make pretty figures (optional)!

What are we NOT using Python for?

- Doing all of our work for us
- Replacing conceptual understanding of the math
- Complicated, long projects like object-oriented programming

Key competencies

CARE 1.2. Step-by-Step Approach

1) Understand what the question is actually asking you to do
2) Identifying the relevant MATH 257 concepts (i.e. the equations)
    a) Sometimes you should actually do scratchwork on paper first!
3) Implementing those calculations in Python
4) Troubleshooting when things don’t go right
5) (Optional) Visualizing your work

Interpreting Coding Questions

CHW5.3. Six degrees of separation: How are we connected?

It's said that everyone in the world is connected to each other by at most six degrees of separation. For example, you know someone who knows someone who knows someone who knows someone who knows someone who knows someone who knows the president of the United States. Of course, this can never be truly tested, but scientists and mathematicians estimate that it could be much less. A study found that Twitter users are on average connected to each other through a chain of at most 4 people.

You will write a function which checks whether or not every pair of people from a group is connected by **at most** a certain degree of separation. Your function should take in a graph (as an adjacency matrix) and a distance and should then return `True` if every pair of nodes is connected by a walk of **at most** the specified distance, and `False` otherwise.

Hints:

- `np.all(matrix > 0)` will return `True` if all of the entries of `matrix` are positive and `False` otherwise.
- `la.matrix_power(M, n)` will return the `n`-th power of `M`
- What do the entries on the diagonal of an adjacency matrix represent? How do their values relate to the problem?

What is given?

What is asked for?

Which information is relevant?

Interpreting Coding Questions

CHW5.3. Six degrees of separation: How are we connected?

It's said that everyone in the world is connected to each other by at most six degrees of separation. For example, you know someone who knows someone who knows someone who knows someone who knows someone who knows someone who knows the president of the United States. Of course, this can never be truly tested, but scientists and mathematicians estimate that it could be much less. A study found that Twitter users are on average connected to each other through a chain of at most 4 people.

You will write a function which checks whether or not every pair of people from a group is connected by **at most** a certain degree of separation. Your function should take in a graph (as an adjacency matrix) and a distance and should then return `True` if every pair of nodes is connected by a walk of **at most** the specified distance, and `False` otherwise.

Hints:

- `np.all(matrix > 0)` will return `True` if all of the entries of `matrix` are positive and `False` otherwise.
- `la.matrix_power(M, n)` will return the `n`-th power of `M`
- What do the entries on the diagonal of an adjacency matrix represent? How do their values relate to the problem?

Completely useless info

Read this carefully

Relating the Question to MATH 257 Concepts

CARE 1.3. Applying the Math

You should have a **list of topics** covered in the class somewhere
- On the course website, in your notes, in your head, etc.

Look for **keywords** in the question
- Ex. adjacency matrix, linear transformation, basis

What **theorems and operations** do you know that are relevant to the question?
- For the previous example, it asks about walks of a specific length:

Theorem 39. *Let $\mathcal{G}$ be a graph and let A be its adjacency matrix. Then the entry in the i-th row and j-th column of A^ℓ is the number of walks of length ℓ from node j to node i on $\mathcal{G}$.*

Use the hints wisely

CHW5.3. Six degrees of separation: How are we connected?

It's said that everyone in the world is connected to each other by at most six degrees of separation. For example, you know someone who knows someone who knows someone who knows someone who knows someone who knows someone who knows the president of the United States. Of course, this can never be truly tested, but scientists and mathematicians estimate that it could be much less. A study found that Twitter users are on average connected to each other through a chain of at most 4 people.

You will write a function which checks whether or not every pair of people from a group is connected by **at most** a certain degree of separation. Your function should take in a graph (as an adjacency matrix) and a distance and should then return `True` if every pair of nodes is connected by a walk of **at most** the specified distance, and `False` otherwise.

Hints:

- `np.all(matrix > 0)` will return `True` if all of the entries of `matrix` are positive and `False` otherwise.
- `la.matrix_power(M, n)` will return the `n`-th power of `M`
- What do the entries on the diagonal of an adjacency matrix represent? How do their values relate to the problem?

It tells you to think about matrix powers!

Implementing the Math in Python

Syntax is important

- You must add “`import numpy as np`”
- You must include “`np.`” before all numpy functions
  - Some of them need “`np.linalg.`” if they are from the linear algebra library
  - You can also add “`import numpy.linalg as la`”

Know what the functions are, or how to look them up

- Numpy documentation for linalg
- What data type is the input? The output?
- How many inputs/outputs?
- Do you need to pre-process the output?

```
soln = la.eig(A)

eigVectors = soln[1]

eigVec1 = eigVectors[:,0]
```

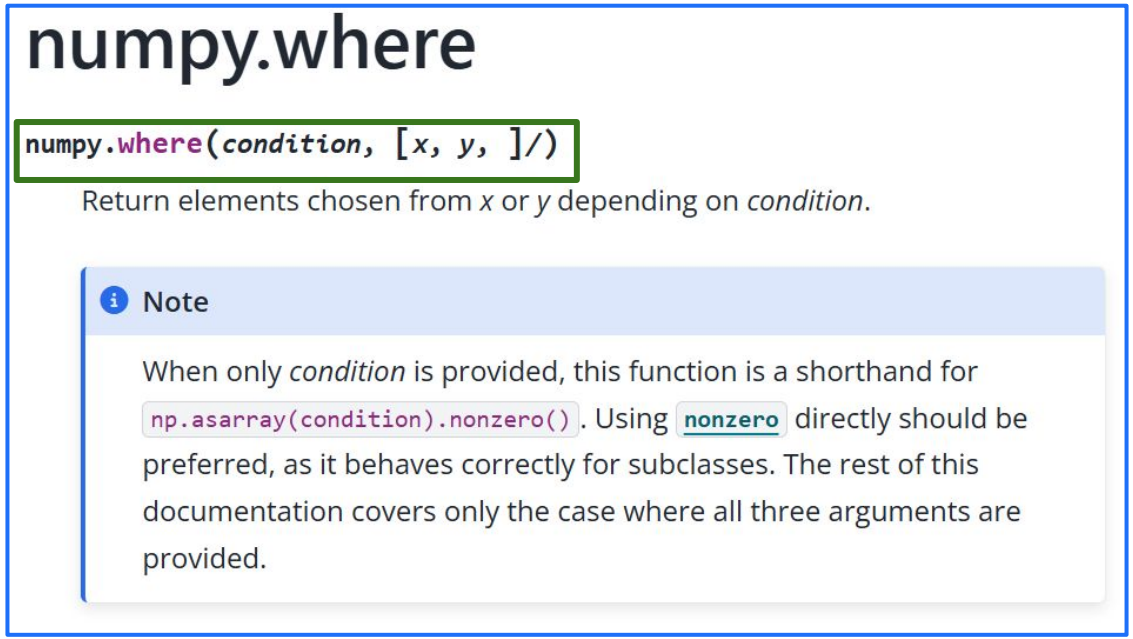
Numpy crash course 1.1: reading documentation

Step 1: Google the function and click on (usually) the first link

Step 2: Look at the syntax

- `Condition` and `x, y` are the inputs
- Read the note! Sometimes it tells you to use another function

Step 3: Look at the types of the inputs and output



Numpy crash course 1.2: reading documentation

Step 3: Look at the types of the inputs and output

Parameters:

condition : *array_like, bool*

Where True, yield *x*, otherwise yield *y*.

x, y : *array_like*

Values from which to choose. *x*, *y* and *condition* need to be broadcastable to some shape.

Returns:

out : *ndarray*

An array with elements from *x* where *condition* is True, and elements from *y* elsewhere.

- `array_like` just means Python will try to apply to it an array; can be most dtypes
- `ndarray` is a formal numpy array data type

Numpy crash course 1.3: reading documentation

Step 4: If you don't understand, keep scrolling to the examples

```
>>> a = np.array([[0, 1, 2],
...               [0, 2, 4],
...               [0, 3, 6]])
>>> np.where(a < 4, a, -1)  # -1 is broadcast
array([[ 0,  1,  2],
       [ 0,  2, -1],
       [ 0,  3, -1]])
```

- Condition: a < 4
- x: a
- y: -1

- This function replaces any value of a matrix greater than or equal to 4 with -1

Numpy Crash Course 2: ndarrays

- What are they?
  - A type of Object from the **Numpy library**
  - It has its **own attributes and functions** that it is compatible with
  - An ndarray is not the same thing as a list
- Forms it can take:
  - array [n,], column vector [n, 1], row vector [1, n], or matrix [n,m]
- It can have multiple axes, but usually 0 = rows, 1 = columns
  - Only one axis means it is a vector (1D)

```
1  print(np.ones(3))
2  print(np.ones((3,)))
```

```
[1. 1. 1.]
[1. 1. 1.]
```

```
1  print(np.ones((3, 1)))
```

```
[[1.]
 [1.]
 [1.]]
```

```
1  print(np.ones((1, 3)))
```

```
[[1. 1. 1.]]
```

These are all NOT the same thing!

Numpy Crash Course 3: Defining Functions

```
def myFunction(input1, input2):
    a, b, c = input1[0]
    dim2 = np.shape(input2)
    return ______
```

- Input and output data types
- READ THE QUESTION
- Check variable names

The setup code gives the following variables:

Name	Type	Description
power_iteration	function	Helper function that performs power iteration
A	2d numpy array	Adjacency matrix containing character pair interactions
names	list	Names of each character such that the i'th label matches the i'th row/column in the adjacency matrix

Your code snippet should define the following variables:

Name	Type	Description
M	2d numpy array	Markov matrix
G	2d numpy array	Game of Thrones matrix
x	1d numpy array	Eigenvector of the Game of Thrones matrix that corresponds to the largest magnitude eigenvalue
protagonist	string	Character with the largest PageRank

Numpy crash course 4: making loops

Hint: if you don't remember the functions and the syntax, you can almost always write a loop (or nested loops) to replace it!

Ex. Let's try to replace the np.where function in the previous example

[2]:
```
a = np.array([[0, 1, 2],
              [0, 2, 4],
              [0, 3, 6]])
```

[4]:
```
for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        if a[i,j] >= 4:
            a[i,j] = -1
print(a)
```

```
[[ 0  1  2]
 [ 0  2 -1]
 [ 0  3 -1]]
```

All loops iterate over an object like a list, tuple, dictionary, or string

- Each iteration, the temporary variable (ex. i and j) changes
- Use the temporary variable inside the loop to check and modify every entry in a matrix

List of Common Functions

np.linspace

np.zeros (ones)

np.zeros_like (ones_like)

np.shape

np.diag

np.arange

np.dot

@ or la.matmul

np.outer

np.where

np.cos, np.arccos, …

la.solve

la.inv

la.eig

la.svd

la.matrix_power

la.norm

len

la.qr

la.det

la.lstsq

.T

Debugging

Reading error messages

1) Open one of the test result dropdowns and scroll all the way to the bottom
2) Figure out what type of error it is
3) Look for what line in your code raises the error
    a) Don't blindly trust the line number: sometimes it references library code instead of yours

Example

```
  File "<string>", line 8, in connected
  File "/usr/local/lib/python3.12/site-packages/numpy/lib/twodim_base.py", line 211, in eye
    m = zeros((N, M), dtype=dtype, order=order)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: 'tuple' object cannot be interpreted as an integer
```

Troubleshooting

CARE 1.4. Testing and Printing

If you are writing a function, you will not see any printed output unless you CALL the function in your code

- Add in print statements so you have checkpoints for debugging

You can **use test cases** to figure out what is going wrong

- Often if you get partial credit it is because only SOME cases work

Ex. "Function returned False and should not have (Hint: This may be because you're not checking to see whether each connection is connected by paths smaller than the specified degree)"

- This is a math/logic error, not a coding issue!

Example from the homework

Given a 512×512 image, stored in the variable `image`, you will break the image into smaller 8×8 chunks. Then for each chunk, you will convert to the DCT basis, drop the frequencies with relatively small contribution, then convert back to the standard basis and clip the resulting pixel values so that they lie between 0 and 255. This image is given in gray scale, where each pixel varies from 0 to 255. To gauge which frequencies have a small contribution, use a tolerance of 10% of the largest absolute value in each particular chunk after converting it into the DCT basis.

The setup code provides the helper function `create_dct_basis(n)`, which creates an $n \times n$ matrix whose columns make up the n-dimensional DCT basis.

To construct your solution, write a code snippet that:

- defines the function `compress_chunk(chunk)` that takes as argument a "chunk" of the image, with dimension 8×8, and returns the compressed "chunk".
- make sure your compressed chunk returned from `compress_chunk()` contains only values between 0 and 255, as in the original image. You may find the function `numpy.clip()` useful.
- use your function to construct the variable `compressed`, which contains the compressed `image`.

Example from the homework

The setup code gives the following variable:

Name	Type	Description
`image`	512×512 numpy array	Test image
`create_dct_basis`	function	Creates the n-dimensional DCT basis

Your code snippet should define the following variable:

Name	Type	Description
`compress_chunk`	function	Function to compress an 8×8 chunk
`compressed`	512×512 numpy array	Compressed image

Example from the homework: potential answer

```python
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt

plt.figure()
plt.imshow(image, cmap='gray') # this shows the original image

compressed = image.copy() # make a copy so that you don't modify the original image
D = create_dct_basis(8) # you need to make a basis for an 8x8 chunk

def compress_chunk(chunk):
    change_to_DCT = D.T @ chunk @ D # change chunk into DCT basis
    threshold = 0.1*np.amax(np.abs(change_to_DCT)) # calculate the 10% threshold

    above_threshold = np.where(np.abs(change_to_DCT) < threshold, 0, change_to_DCT) # filter the matrix
    convert_back = D @ above_threshold @ D.T # change chunk back from DCT basis

    return np.clip(convert_back, 0, 255) # clip values outside 0 to 255

for i in range(0, 512, 8):
    for j in range(0, 512, 8):
        image_chunk = image[i:i+8, j:j+8].copy() # create an 8x8 chunk
        compressed[i:i+8, j:j+8] = compress_chunk(image_chunk) # compress that chunk and add it

plt.figure()
plt.imshow(compressed, cmap='gray') # this shows the compressed image
```

Visualization

CARE 1.5. Optional Info about Graphs

We use a library called matplotlib.pyplot for this

- Usually say “import matplotlib.pyplot as plt”
  - Documentation here
- You need to make an Axes and Fig object to plot things
- Most commonly, you use the subplots function to actually make a plot
  - fig, ax = plt.subplots()
  - ax.plot(x, y)
- Everything else is just to make it look fancy

Where else can I get help?

1) **MATH 257 has tons of CA office hours**
    - **See schedule here**
2) **For coding issues, the documentation is available online**
3) **For conceptual issues, ask the TAs!**
4) **CARE Drop-in Tutoring****

****some tutors that list MATH 257 are not comfortable with doing the coding questions**

Now it’s your turn to practice! Open the queue, click the link, and access the PL and Colab!

Please consider also filling out the feedback form on the queue so we can improve this in the future!

List of Common Functions

np.linspace

np.zeros (ones)

np.zeros_like (ones_like)

np.shape

np.diag

np.arange

np.dot

@ or la.matmul

np.outer

np.where

np.cos, np.arccos, …

la.solve

la.inv

la.eig

la.svd

la.matrix_power

la.norm

len

la.qr

la.det

la.lstsq

.T