

## Topics in Computer Science - Bitcoin: Programming the Future of Money - ITCS 4010 & 5010 - Spring 2025 - UNC Charlotte

### Homework 1 - Python Recap (60 Points)

Write Your Name Here: Hritika Kucheriya

Names of collaborators: Google; Stackoverflow; YouTube; w3school

#### Submission Instructions

1. Click the Save button at the top of the Jupyter Notebook.
2. Please make sure to have entered your name above.
3. Select Cell -> All Output -> Clear. This will clear all the outputs from all cells (but will keep the content of all cells).
4. Select Cell -> Run All. This will run all the cells in order, and will take several minutes.
5. Once you've rerun everything, select File -> Download as -> PDF via LaTeX and download a PDF version showing the code and the output of all cells, and save it in the same folder that contains the notebook file.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly.
7. Submit **both** your PDF and the notebook file .ipynb on Gradescope.
8. Make sure your Gradescope submission contains the correct files by downloading it after posting it on Gradescope.

#### 1. Functions (5 Points)

**Disclaimer: Do not use numpy library in this assignment!** The point of this assignment is to implement important basic functions in native Python.

##### a. Dot Product (2 Points)


The function `dotProduct` multiplies two vectors together. Represent vectors as simple lists. Therefore, the product of vector  $a = (a_1 \dots a_n)$  and  $b = (b_1 \dots b_n)$  is given by:

$$\sum_{i=1}^n (a_i b_i)$$

Now write a function `dotProduct` which takes two inputs `vec1`, `vec2` and returns the dot product of the two vectors.

```
def dotProduct(vec1, vec2):
    return sum(a * b for a, b in zip(vec1, vec2))
```

```
vector1 = [1, 2, 3]
vector2 = [4, 5, 6]
dotprod = dotProduct(vector1, vector2)
print("Dot product is:", dotprod)
```

 Dot product is: 32

##### b. Check datastructure (3 Points)

Write a function `isDict` that takes a data structure as an input and returns a Boolean, True if the structure is a dictionary and False otherwise.

```
def isDict(data_structure):
    return isinstance(data_structure, dict)
```

```
data_struct_1 = {'a':123, 'b': 456}
data_struct_2 = [1, 2, 3, 4]
print(isDict(data_struct_1))
print(isDict(data_struct_2))
```

 True  
False

## 2. Introduction to OOP (15 Points)


This exercise introduces basic Object-Oriented Programming (OOP) concepts in Python. You'll create a superclass named `Shapes` along with subclasses for various shapes. The exercise also includes implementing basic mathematical functions such as calculating area and perimeter.

### a. Shapes (3 Points)

Start by defining the `Shapes` class, initializing it with a shape name in the `__init__` method. Implement the `__repr__` method to return a string in the format "`<shape name>` is a shape" (for example, `print(Circle)` should output `Circle is a shape`).

```
class Shapes:
    def __init__(self, name):
        self.name = name
    def __repr__(self):
        return f"{self.name} is a shape"
```

```
pentagon = Shapes("pentagon")
print(pentagon, "\n")
```

 pentagon is a shape

### b. Circle (5 Points)

Define a subclass of `Shapes` – `Circle` and initialize the only input: radius.

`__repr__` method should return the radius of the circle.

Write the method `area` which returns the area of the circle and the method `perimeter` which returns the perimeter of the circle.

```
import math


class Circle(Shapes): # Subclass of Shapes
    def __init__(self, radius):
        super().__init__("Circle") # Call the parent constructor
        self.radius = radius

    def __repr__(self):
        # Return the radius of the circle
        return f"The radius of the circle is {self.radius}"

    def area(self):
        # Calculate and return the area of the circle
        return math.pi * (self.radius ** 2)

    def perimeter(self):
        # Calculate and return the perimeter (circumference) of the circle
        return 2 * math.pi * self.radius
```

```
circle1 = Circle(5)
print(circle1, "\n")
```

 The radius of the circle is 5

```
import math
c1Area = circle1.area()
print("The area of circle1 is", c1Area, "\n")
```

 The area of circle1 is 78.53981633974483

```
c1Per = circle1.perimeter()
print("The perimeter of circle1 is", c1Per, "\n")
```

 The perimeter of circle1 is 31.41592653589793

### ✓ c. Triangle 1 (4 Points)

Define a subclass of Shapes – `Triangle` and initialize the inputs: base and height.

`__repr__` method should return the value of base and height of the triangle in the format - `<base> & <height>`.


Write the method `area` that returns the area of the triangle.

```
class Triangle(Shapes): # Subclass of Shapes
    def __init__(self, base, height):
        super().__init__("Triangle") # Call the parent constructor
        self.base = base
        self.height = height


    def __repr__(self):
        # Return the base and height in the format "<base> & <height>"
        return f"{self.base} & {self.height}"

    def area(self):
        # Calculate and return the area of the triangle
        return 0.5 * self.base * self.height
```

```
triangle1 = Triangle(2, 22)
print(triangle1, "\n")
```

 2 & 22

```
tri1Area = triangle1.area()
print("The area of triangle1 is", tri1Area, "\n")
```

 The area of triangle1 is 22.0

### ✓ d. Triangle 2 (3 Points)

Define a subclass of Triangle called `Right_Triangle` that is a right triangle. Write a method `perimeter` that returns the perimeter of the right triangle.

```
import math

class Right_Triangle(Triangle): # Subclass of Triangle
    def __init__(self, base, height):
        super().__init__(base, height) # Call the parent constructor

    def perimeter(self):
        # Calculate the hypotenuse
        hypotenuse = math.sqrt(self.base ** 2 + self.height ** 2)
        # Calculate and return the perimeter of the right triangle
        return self.base + self.height + hypotenuse
```

```
right_tri = Right_Triangle(3,4)
print(right_tri, '\n')
triperimeter = right_tri.perimeter()
print("The perimeter of the triangle is", triperimeter, '\n')
```

 3 & 4

The perimeter of the triangle is 12.0

## 3. Matrices (40 Points)

Create a class Matrix, which allows for the object-oriented programming features in Python to operate on matrices. This is useful to overload the standard arithmetic operators to define matrix addition, multiplication, and inversion. We also allow for functions on matrices (to be

defined as methods on the class), like transpose, print, and so on. In this baseline, we represent a matrix as a list of lists, i.e., a list of rows. E.g., `[[1,2],[2,3]]`.

**Disclaimer: Do not use numpy library in this assignment!**

### ✓ a. Matrix class (5 Points)

Define a class `Matrix`, initialize input matrix, number of rows, and columns (m,n).

Initialize  $m \times n$  matrix to zero matrix (all entries are zero), if no input matrix is passed.

Write `__repr__` method such that the matrix should be printed row wise.

For instance, if the input is `[[1,2],[2,3]]`, it should be printed as:

```
[1,2]
```

```
[2,3]
```

Write a method `__eq__` that overloads `'=='` to check the equality of the two matrices ((including the elements, number of rows and columns). Return True if two matrices are equal.

```
class Matrix:
    def __init__(self, data=None, rows=None, cols=None):
        if data is None:
            # Initialize a zero matrix if no input is provided
            self.data = [[0 for _ in range(cols)] for _ in range(rows)]
            self.rows = rows
            self.cols = cols
        else:
            # Use the provided data to initialize the matrix
            self.data = data
            self.rows = len(data)
            self.cols = len(data[0]) if data else 0

    def __repr__(self):
        # Represent the matrix row-wise
        return "\n".join([str(row) for row in self.data])

    def __eq__(self, other):
        # Overload '==' to check if two matrices are equal
        if self.rows != other.rows or self.cols != other.cols:
            return False
        return self.data == other.data

    def transpose(self):
        # Transpose the matrix by flipping rows and columns
        transposed_data = [[self.data[j][i] for j in range(self.rows)] for i in range(self.cols)]
        return Matrix(transposed_data)
```

```
matrixA = Matrix([[1,2,3], [3,4,6]],2,3)
matrixB = Matrix([[6,3,7], [7,9,12]],2,3)
matrixC = Matrix([[3,3], [7,4], [2,8]],3,2)
matrixD = Matrix([[10,11], [23,21]],2,2)
```

```
print(matrixA == matrixB)
```

```
False
```

### ✓ b. Transpose (5 Points)

Write the method `transpose` that transposes a matrix. It takes the matrix data and flips the rows and columns.

```
matrixA = Matrix([[1, 2, 3], [3, 4, 6]], 2, 3)
print('\nOriginal matrixA:\n', matrixA)
```

```
Original matrixA:
[1, 2, 3]
[3, 4, 6]
```

```
print('\nTranspose of matrixA:\n', matrixA.transpose())
```



```
Transpose of matrixA:
[1, 3]
[2, 4]
[3, 6]
```

## Operator Overloading

We will now overload the standard arithmetic operators. For example, we can add two matrices by adding each of their elements.

### ✓ c. Addition (5 Points)

Overload addition to add two matrices together.

If the argument passed is an integer, it will add that integer to each element of the original matrix.

If the argument passed is a matrix (with the same dimensions as the original), then the new matrix will be the element-by-element addition.

```
class Matrix:
    def __init__(self, data=None, rows=None, cols=None):
        if data is None:
            self.data = [[0 for _ in range(cols)] for _ in range(rows)]
            self.rows = rows
            self.cols = cols
        else:
            self.data = data
            self.rows = len(data)
            self.cols = len(data[0]) if data else 0

    def __repr__(self):
        # Return the matrix in a readable row-wise format
        return "\n".join([str(row) for row in self.data])

    def __eq__(self, other):
        if self.rows != other.rows or self.cols != other.cols:
            return False
        return self.data == other.data

    def transpose(self):
        transposed_data = [[self.data[j][i] for j in range(self.rows)] for i in range(self.cols)]
        return Matrix(transposed_data)

    def __add__(self, other):
        if isinstance(other, Matrix):
            if self.rows != other.rows or self.cols != other.cols:
                raise ValueError("Matrix dimensions must match for addition.")
            added_data = [[self.data[i][j] + other.data[i][j] for j in range(self.cols)] for i in range(self.rows)]
            return Matrix(added_data)
        elif isinstance(other, int):
            added_data = [[self.data[i][j] + other for j in range(self.cols)] for i in range(self.rows)]
            return Matrix(added_data)
        else:
            raise TypeError("Unsupported type for addition.")

# Example usage
matrixA = Matrix([[1, 2, 3], [3, 4, 6]], 2, 3)
matrixB = Matrix([[6, 3, 7], [7, 9, 12]], 2, 3)
```

```
zero = Matrix([[0, 0, 0], [0, 0, 0]], 2, 3)
print(matrixA + zero == matrixA)
print('\nMatrix addition:\n', matrixA + matrixB)
print('\nMatrix addition:\n', matrixA + 5)
```



True

```
Matrix addition:
[7, 5, 10]
[10, 13, 18]
```

```
Matrix addition:
[6, 7, 8]
[8, 9, 11]
```

### ✓ d. Multiplication (5 Points)

Overload multiplication for matrix multiplication such that if the multiplier is a scalar, then scalar multiplication should be performed with the original matrix otherwise it is matrix multiplication to be performed.

```
def __mul__(self, other):
    if isinstance(other, Matrix):
        # Matrix multiplication
        if self.cols != other.rows:
            raise ValueError("Matrix dimensions are not compatible for multiplication.")
        result = [[sum(self.data[i][k] * other.data[k][j] for k in range(self.cols)) for j in range(other.cols)] for i in range(self.rows)]
        return Matrix(result)
    elif isinstance(other, (int, float)):
        # Scalar multiplication
        result = [[self.data[i][j] * other for j in range(self.cols)] for i in range(self.rows)]
        return Matrix(result)
    else:
        raise TypeError("Unsupported type for multiplication.")

def __rmul__(self, other):
    # Reverse multiplication to handle scalar * matrix
    return self.__mul__(other)

Matrix.__mul__ = __mul__
Matrix.__rmul__ = __rmul__

# Example Matrices
matrixA = Matrix([[1, 2, 3], [3, 4, 6]], 2, 3)
matrixC = Matrix([[3, 3], [7, 4], [2, 8]], 3, 2)
matrixD = Matrix([[10, 11], [23, 21]], 2, 2)
```

```
print((matrixA * matrixC) * matrixD == matrixA * (matrixC * matrixD))

print('Matrix Multiplication:\n', matrixA * matrixC)
k = 2
print('\nScalar Multiplication:\n', k * matrixA)
```

```
True
Matrix Multiplication:
[23, 35]
[49, 73]

Scalar Multiplication:
[2, 4, 6]
[6, 8, 12]
```

### ✓ e. Subtraction (2 Points)

Observe that for two matrices A and B of the same dimensions,  $A - B = A + (-1)B$ . Hence use multiplication and addition for overloading matrix subtraction.

```
def __sub__(self, other):
    if isinstance(other, Matrix):
        # Subtraction is equivalent to addition with the negative of the other matrix
        return self + (-1 * other)
    else:
        raise TypeError("Unsupported type for subtraction.")

Matrix.__sub__ = __sub__
```

```
print(matrixA - matrixA == zero)
print('\nMatrix Subtraction:\n', matrixA - matrixB)
```

```
True

Matrix Subtraction:
[-5, -1, -4]
[-4, -5, -6]
```

### ✓ f. Comparator (8 Points)

Overload the comparison operators `<` and `>`. For matrices A and B,  $A < B$  if every element in A is strictly less than every element in B.  $A > B$  if every element in A is strictly greater than B.

```
def __lt__(self, other):
    if isinstance(other, Matrix):
        # Check if every element of self is less than every element of other
        return all(self.data[i][j] < other.data[i][j] for i in range(self.rows) for j in range(self.cols))
    else:
        raise TypeError("Unsupported type for comparison.")

def __gt__(self, other):
    if isinstance(other, Matrix):
        # Check if every element of self is greater than every element of other
        return all(self.data[i][j] > other.data[i][j] for i in range(self.rows) for j in range(self.cols))
    else:
        raise TypeError("Unsupported type for comparison.")

Matrix.__lt__ = __lt__
Matrix.__gt__ = __gt__
```

```
matA = Matrix([[1,1], [2,2]])
matB = Matrix([[3,4], [5,6]])
matC = Matrix([[1,5], [7,8]])

print("\n Is matA greater than matB?:", matA>matB)
print("\n Is matB greater than matA?:", matB>matA)
print("\n Is matA lesser than matB?:", matA<matB)
print("\n Is matB lesser than matA?:", matB<matA)
print("\n Is matA greater than matC?:", matA>matC)
print("\n Is matA lesser than matC?:", matA<matC)
```



```
Is matA greater than matB?: False
Is matB greater than matA?: True
Is matA lesser than matB?: True
Is matB lesser than matA?: False
Is matA greater than matC?: False
Is matA lesser than matC?: False
```

## ✓ g. Square Matrix (8 Points)

Write a subclass `SquareMatrix` that inherits from the `Matrix` class. The constructor of `SquareMatrix` should accept two arguments: the size of the square matrix (number of rows or columns) and optionally the matrix data. Ensure that the `SquareMatrix` class inherits all methods from the `Matrix` class and handles the square matrix constraints appropriately.

```
class SquareMatrix(Matrix):
    def __init__(self, data=None, size=None):
        if data:
            # Ensure the provided matrix is square
            if len(data) != len(data[0]):
                raise ValueError("Provided matrix is not square.")
            super().__init__(data=data)
        elif size:
            # Create a zero square matrix of given size
            super().__init__(data=[[0 for _ in range(size)] for _ in range(size)])
        else:
            raise ValueError("Either data or size must be provided for a square matrix.")
```

```
square_matrixA = SquareMatrix([[21,35], [15,34]],2)
print("\nSquare matrix A:\n", square_matrixA)
print("\nTranspose of Square matrix A:\n", square_matrixA.transpose())
square_matrixB = SquareMatrix([[45,43], [56,3]])
print("Square matrix B:\n", square_matrixB)
print("\nsquare matrix addition:\n", square_matrixA + square_matrixB)
print("\nsquare matrix subtraction:\n", square_matrixA - square_matrixB)
print("\nsquare matrix multiplication:\n", square_matrixA * square_matrixB)
```

```
k = 3
print("\nsquare matrix scalar multiplication:\n", k * square_matrixA)
```



```
Square matrix A:
[21, 35]
[15, 34]

Transpose of Square matrix A:
[21, 15]
[35, 34]
Square matrix B:
[45, 43]
[56, 3]

square matrix addition:
[66, 78]
[71, 37]

square matrix subtraction:
[-24, -8]
[-41, 31]

square matrix multiplication:
[2905, 1008]
[2579, 747]

square matrix scalar multiplication:
[63, 105]
[45, 102]
```

#### h. Trace (2 Points)

Write the `trace` method in `SquareMatrix` class that gives the trace of a matrix (the sum of its diagonal elements).

```
class SquareMatrix(Matrix):
    def __init__(self, data=None, size=None):
        if data:
            # Ensure the provided matrix is square
            if len(data) != len(data[0]):
                raise ValueError("Provided matrix is not square.")
            super().__init__(data=data)
        elif size:
            # Create a zero square matrix of given size
            super().__init__(data=[[0 for _ in range(size)] for _ in range(size)])
        else:
            raise ValueError("Either data or size must be provided for a square matrix.")

    def trace(self):
        # Calculate the sum of the diagonal elements
        return sum(self.data[i][i] for i in range(self.rows))

# Example usage
square_matrixA = SquareMatrix([[21, 35], [15, 34]], 2)
print("\nSquare matrix A:\n", square_matrixA)
```



```
Square matrix A:
[21, 35]
[15, 34]
```

```
print("\nTrace of Square matrix A:\n", square_matrixA.trace())
```



```
Trace of Square matrix A:
55
```

```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```



```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
texlive-latex-extra is already the newest version (2021.20220204-1).
texlive-xetex is already the newest version (2021.20220204-1).
The following additional packages will be installed:
  libcbmark-gfm-extensions0.29.0.gfm.3 libcbmark-gfm0.29.0.gfm.3 pandoc-data
Suggested packages:
  texlive-luatex pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc nodejs php python
```



```

libjs-mathjax libjs-katex citation-style-language-styles
The following NEW packages will be installed:
  libcmack-gfm-extensions0.29.0.gfm.3 libcmack-gfm0.29.0.gfm.3 pandoc pandoc-data texlive
0 upgraded, 5 newly installed, 0 to remove and 49 not upgraded.
Need to get 20.6 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmack-gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmack-gfm-extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all 2.9.2.1-3ubuntu2 [81.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64 2.9.2.1-3ubuntu2 [20.3 MB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all 2021.20220204-1 [14.3 kB]
Fetched 20.6 MB in 1s (15.7 MB/s)
Selecting previously unselected package libcmack-gfm0.29.0.gfm.3:amd64.
(Reading database ... 161617 files and directories currently installed.)
Preparing to unpack .../libcmack-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb ...
Unpacking libcmack-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Selecting previously unselected package libcmack-gfm-extensions0.29.0.gfm.3:amd64.
Preparing to unpack .../libcmack-gfm-extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb ...
Unpacking libcmack-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Selecting previously unselected package pandoc-data.
Preparing to unpack .../pandoc-data_2.9.2.1-3ubuntu2_all.deb ...
Unpacking pandoc-data (2.9.2.1-3ubuntu2) ...
Selecting previously unselected package pandoc.
Preparing to unpack .../pandoc_2.9.2.1-3ubuntu2_amd64.deb ...
Unpacking pandoc (2.9.2.1-3ubuntu2) ...
Selecting previously unselected package texlive.
Preparing to unpack .../texlive_2021.20220204-1_all.deb ...
Unpacking texlive (2021.20220204-1) ...
Setting up texlive (2021.20220204-1) ...
Setting up libcmack-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Setting up libcmack-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Setting up pandoc-data (2.9.2.1-3ubuntu2) ...
Setting up pandoc (2.9.2.1-3ubuntu2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

```

```

from google.colab import drive
drive.mount('/content/drive')

```

 Mounted at /content/drive