CSCI2040: Introduction to Python **2024-2025 Term 1**

Lab Assignment 5

Instructor: Yim Pan, CHUI Due: 23:59 on Wed. Dec. 06th, 2024

Notes

1. Name your script files as `p1.py`, `p2.py`, etc. (sensitive to cases). Please install the `numpy` and `matplotlib` packages via `pip` or your IDE's package manager before proceeding.
2. Your implemented Python programs on Numpy (Exercise 1, 2) for this lab assignment will be graded by our grading scripts, which will run your submitted scripts and compare their produced output with the expected output for different test cases.
3. So, please follow closely the format of the sample output for each question. Your program should produce exactly the same output as the sample (same text, symbols, letter case, spacing, number format, etc.). The output for each question should end with a single newline character, which has been provided by the `print()` function by default.
4. **We do not have any test scripts on Visualization (Exercise 3, 4) to provide for this lab assignment.** The generated images will be reviewed individually and graded manually.
5. Apart from `numpy` and `matplotlib`, you are **NOT** allowed to use any other third-party libraries in all your solutions for this assignment.
6. Please follow exactly the specified name, parameter list, and return value(s) for the methods required in each question (to facilitate our grading). You are, however, allowed to define additional methods or inner functions for further task decomposition of a required method if you see fit.
7. Your main client code (for calling and testing the required methods) should be put under an `if __name__ == '__main__':` check for hiding your testing output when we import your scripts into our grading scripts.
8. You may assume that all user inputs are **always valid** and your program for each question need not include code to validate them unless specified otherwise. You may assume that all the corner cases **we have not mentioned** in this document will **not appear** in the hidden grading cases, so you do not have to worry too much about wrong inputs (e.g., entering "abc" for an input expecting an integer value) unless you are required to do so.
9. Only **Python 3.x** is acceptable.
10. Your code should contain only specified functions. Please delete all the debug statements (e.g. print) before submission.

Exercise 1 (25 marks)

You are given a matrix A of shape(m,n) and a matrix B of shape(n,m). Your task is to compute the matrix $C = A^T + B$ using NumPy, which involves taking the transpose of matrix A and adding it to matrix B, where C is a matrix of shape(n,m). Matrices A and B have the same data type.

Write a function `matrix_calculation(A, B)` in a script named `p1.py`, which takes matrices A and B as input and returns their product, namely matrix C. The prototype of the function is given as follows:

```
def matrix_calculation(A, B):

    # Check the shape of matrix A and B
    # ...
    # Create a new matrix A_transpose (hint: np.zeros())
    # ...

    return C
```

Note that it could happen that the dimensions of the input matrices are incompatible with carrying out matrix calculation. In this case, the function prints the message "Please check the shape of matrix A and B" and returns `none` to the caller.

Notes: You are NOT allowed to use NumPy's built-in functions like *np.transpose*() to carry out the matrix calculation. Otherwise, you will get zero marks for this question. Instead, you have to implement your own matrix calculating function using only basic NumPy operations such as array indexing, slicing, and element-wise operations.

Sample Run:

Given matrix $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix}$, and matrix $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix}$,

we calculate matrix $C = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \\ a_{14} & a_{24} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \\ a_{31}+b_{31} & a_{32}+b_{32} \\ a_{41}+b_{41} & a_{42}+b_{42} \end{bmatrix}$

Exercise 2 (25 marks)

In this task, you are required to write a function `reshape_and_translate(array, new_shape, shift_r, shift_c)` that accomplishes the following:

- **Reshape the Input Array:** Reshape the one-dimensional input NumPy array `array` into a matrix with the specified shape `new_shape`, which is a tuple of two positive integers.
- **Perform the Shift Operation:** Shift the elements of the reshaped matrix according to the specified shift values:

  - `shift_r`: Corresponds to the movement along **rows**. A positive value shifts the matrix **downward**. A negative value shifts the matrix **upward**.

  - `shift_c`: Corresponds to the movement along **columns**. A positive value shifts the matrix **to the right**. A negative value shifts the matrix **to the left**.

  - Both `shift_c` and `shift_r` are of type "int."
- **Handle Out-of-Bounds Positions:** For positions that are shifted out of the matrix boundaries, fill them with zeros. There is no wrapping around of elements.
- **Return the Shifted Matrix:** After performing the shift operation, return the resulting matrix.

Function Prototype:

```
def reshape_and_translate(array, new_shape, shift_r, shift_c):

    # Reshape the array
    # ...
    # Create a new matrix with the same shape
    # (hint: translated_array = np.zeros_like(reshaped_array))
    # The dtype of input ndarray is "int"

    # ...
    # Perform the shift operation
    # ...

    return translated_array
```

Sample Run 1:
An array consisting of numbers from 1 to 16 is reshaped to shape `(2, 8)` and then shifted 2 columns to the right and 1 row down.

- **Input:**

  - `array` = np.arange(1, 17)
  - `new_shape` = (2, 8)
  - `shift_r` = 1
  - `shift_c` = 2
- **Original Array:** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
- **Matrix After Reshaping:**

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{bmatrix}$$

- **Output Matrix After Shifting:**

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

Sample Run 2 (Negative Shifts):
An array consisting of numbers from 1 to 12 is reshaped to shape `(3, 4)` and then shifted 1 row up and 2 columns to the left.

- **Input:**
  - `array` = np.arange(1, 13)
  - `new_shape` = (3, 4)
  - `shift_r` = -1
  - `shift_c` = -2
- **Original Array:** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
- **Matrix After Reshaping:**

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- **Output Matrix After Shifting:**

$$\begin{bmatrix} 7 & 8 & 0 & 0 \\ 11 & 12 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Notes:
- If the shift moves all elements out of the matrix boundaries, the resulting matrix should be filled with zeros.

- The input array must be compatible with the specified `new_shape` (i.e., the total number of elements matches).
- The data type of the input array should be maintained in the output array. And the dtype of input ndarray is "int".
- Do not use any wrapping or circular shifting; elements shifted out are discarded, and new positions are filled with zeros.

Exercise 3 (25 marks)

Write a script in `p3.py` to plot a histogram for the `random_numbers` array generated by the following code:

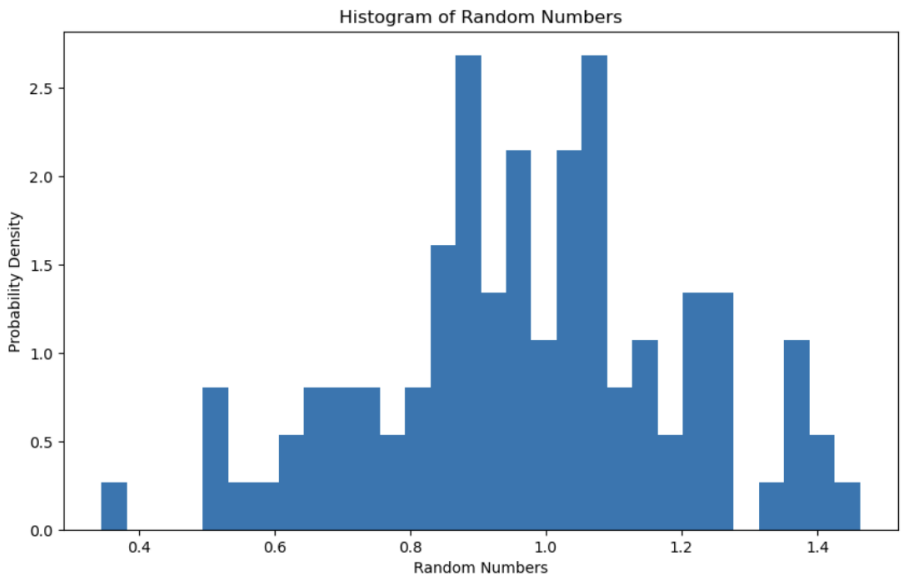
```
# Generate random numbers with fixed seed
# import numpy as np
# np.random.seed(42)
# random_numbers = np.random.normal(1, 0.25, 100)
```

Hints:

- The function `np.random.normal()` is used to generate random numbers from a normal (Gaussian) distribution. In this case, the mean is set to 1, the standard deviation is set to 0.25, and the number of samples generated is 100. Refer to the official documentation for more details: `np.random.normal()`.
- Use the function `plt.hist()` from `matplotlib` to create the histogram. You can refer to the official documentation here: `plt.hist()`.
- Set the `bins` argument for the histogram as 30. The `bins` argument determines how the data is grouped into intervals, affecting the level of detail shown in the histogram.
- Set the x-axis as the value of the random numbers, and the y-axis as the probability density.
- The size of the figure is 10 inches by 6 inches.
- Set the figure title as "Histogram of Random Numbers".

Save your figure as `histogram.png`. (Hint: You could use `savefig()` in `matplotlib`).

Sample Run:
An example result of the histogram figure is as follows:



Exercise 4 (25 marks)

Write a script in `p4.py` to plot a pie chart for the daily sales record of the CU-Store, which is provided in the following table:

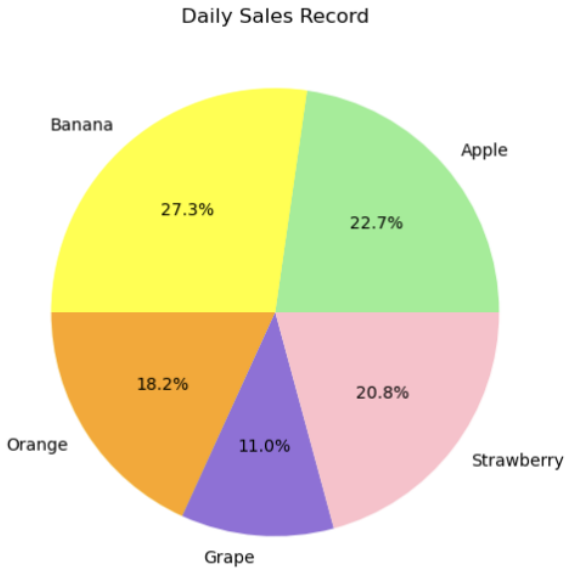
Fruit	Apple	Banana	Orange	Grape	Strawberry
Quantity	35	42	28	17	32

Hints:

- You can customize the appearance of the chart using `matplotlib`'s `rcParams`. The `rcParams` is a configuration dictionary that allows you to control the default styles and settings for your plots.
- Refer to the official documentation for more details: Customizing with rcParams.
- Color requirements: apple should be in `lightgreen`, banana in `yellow`, orange in `orange`, grape in `mediumpurple`, and strawberry in `pink`.
- Use `autopct='%1.1f%%'` to display the percentage value of each category.
- The size of the figure is 10 inches by 6 inches.
- Set the figure title as "Daily Sales Record", and add a separation of 20 points between the title and the pie chart. You can achieve this by using either of the following methods:
  - Set globally: `plt.rcParams['axes.titlepad'] = 20`
  - Set for this plot: `plt.title("Daily Sales Record", pad=20)`

Save your figure as `pie.png`.

Sample Run:
An example result of the pie chart is as follows:



Submission rules

1. Please name your script files with the **exact** names (`p1.py`, ..., `p3.py`) specified in this assignment and test them thoroughly. Any script with any syntax error will not be marked.
2. Please pack all your script files as well as the generated figures from Exercise 2&3 as a single archive named as

   `<student-id>_lab5.zip`

   For example, `1155012345_lab5.zip`, i.e., replace `<student-id>` with your own student ID.
3. For each group, please pack all your script files as a single archive named as

   `<student-id1>_<student-id2>_lab5.zip`

   For example, `1155012345_1155054321_lab5.zip`, i.e., just replace `<student-id1>` and `<student-id2>` with your own student IDs.
4. Upload the zip file to your blackboard (`https://blackboard.cuhk.edu.hk`),
   - Only one member of each group needs to upload the archive file.
   - No need to include the testing script files we provided.
   - Submit no later than 23:59 on Wed. Dec. 06th, 2024.
   - You may submit *multiple times* before the due date but only the last submission will be graded. Make sure that your last submission contains all the required script files of the latest version that you want to submit.
5. Students in the same group would get the same marks. Marks will be deducted if you do not follow the Notes and Submission Rules. Anyone/Anygroup who is caught plagiarizing would get 0 score!