Required:

1) You are tasked with creating a bond valuation tool for investors that will help them compute the price of a bond based on several key financial variables. The investors indicate that they are able to provide the values of the following variables: the annual coupon rate, the bond's face value, the annual discount rate or required return, the investor's tax rate, the bond's maturity, and the bond's coupon payment frequency. The coupon payment frequency should be one of the following options: 'monthly', 'quarterly', 'semi-annually', or 'annually'. Write a Python function to calculate the bond's present value (price). Test the function using the necessary input values provided by the investor, as follows:

- Coupon rate: 6.25%
- Principal: 1,000
- Tax rate: 20%
- Maturity: 15 years
- Coupon payment frequency: quarterly
- Required rate of return: n% (where n corresponds to the last digit of your P number. If the last digit is 0, use 10%)

After calculating the bond's price, display the result with a precision of 3 decimal places using the command *print('Bond price = xxxx')*.

[50 points]

2) In addition to calculating the bond price, investors also want to estimate the bond's duration, which measures how sensitive the bond price is to changes in interest rates. Write a Python function to compute the duration based on the necessary bond parameters provided by the investor in the previous task. Once the bond's duration has been calculated, print the result with 3 decimal places using *print('Duration = xxxx')*.

Hint: For each cash flow received, you need to times t_i, which is the time in years until the ith payment is received.

[50 points]

(Coding tips are provided on the next page, some of which might be helpful.)

Deadline: Wednesday, 30 October 2024, 14:00 GMT

Submit your xxxx.*py* file only, which is named using your 9-digit student number, e.g., 222222222.py. You only have one chance to submit.

Note: P number is your 9-digit student number, e.g., 222222222.

Tips: (some might be helpful)

import *numpy* as *np*

- **np.full()** creates an array of a specified size (or shape) with a constant value. For example, *arr = np.full(5, 7)* will create [7, 7, 7, 7, 7], and *np.full((3, 4), 2)* will create a matrix with 3 rows and 4 columns, where all elements are 2.
- **np.arange()** creates arrays with evenly spaced values within a given range. For example, *arr = np.arange(5)* generates an array with values starting from 0 up to (but not including) 5, i.e., [0, 1, 2, 3, 4]. Similarly, *arr = np.arange(1, 10, 2)* creates [1, 3, 5, 7, 9], starting at 1 and going up to (but not including) 10, with a step of 2.
- **np.sum()** computes the sum of all elements in an array, or along a specified axis, and **np.cumsum()** computes the cumulative sum of elements. For example, *np.sum([1, 2, 3, 4])* will return 10, and *np.cumsum([1, 2, 3, 4])* will return [1, 3, 6, 10].
- **else statement**:

```
if condition1:
    % Code to execute if condition1 is True
elif condition2:
    % Code to execute if condition2 is True
elif condition3:
    % Code to execute if condition2 is True
... ...
else:
    % Code to execute if none of the above conditions are True
```

Example:

```
x = 2
if x < 0 or x > 10:
    raise ValueError("x is out of the allowed range (0-10)")
elif x > 5:
    print("x is greater than 5")
elif x == 3:
    print("x is equal to 3")
else:
    print("x is less than or equal to 5, but not equal to 3")
```