ECON6080/8080 Assignment 1

Total marks: 10 marks. Marks will depend on accuracy and clarity. Speed is not an issue.

Use Python to answer all questions.

Q1 [2 marks]

In the class we wrote a program that evaluate an $(n-1)$-th order polynomial

$$\sum_{i=0}^{n-1} a_i x^i,$$

by adding up $a_i x^i$ for $i = 0, 1, ..., n-1$. This is however known to be inefficient.

Horner's method is a more efficient way to evaluate a polynomial. It is an iterative method illustrated by the following pseudo-code:

1. Initialize $x \in \mathbb{R}$ and $a \in \mathbb{R}^n$.
2. Initialize $p = 0.0$.
3. Starting from $i = n-1$,

$$\begin{aligned} p &\leftarrow a[i] + p \times x \\ i &\leftarrow i - 1 \end{aligned}$$

   and keep repeating until $i = 0$.

Write a function that implements Horner's method (steps 2 and 3 above). It must take (x, a) as input and return p as output. (It may be useful to realize that, if `a` is a list or a `numpy.ndarray`, then `a[::-1]` is a list/array with a reversed order. But you are not required to use it.)

Q2 [2 marks]

Write a program which

1. Asks a user to enter a number;
2. Displays "It is a positive odd number." if the entered number is a positive odd number;
3. Displays "It is a positive even number." if the entered number is a positive even number; and
4. Displays "It is either negative or non-integer." if neither condition is satisfied.

Hint: To judge whether a number is an odd number or an even number, an operator `%` may be useful. See e.g. `https://automatetheboringstuff.com/chapter1/`. Report what happens when the user enters a non-number object such as letters.

Q3 [2 marks]

Write another program that extends the program in Q2 so that it displays "It is not a number." when the user enters a non-number object such as letters. (Don't lose your answer to Q2! Answer in a different cell if you use Jupyter notebook and in a different file if you write `.py` files.)

Hint: Read the Exception Handling section in `https://automatetheboringstuff.com/chapter3/`.

Q4 [2 marks]

Consider a difference equation:

$$x_{n+1} = 1 + \frac{1}{x_n}, \quad n = 0, 1, 2, ... \tag{1}$$

with a strictly positive initial value:

$$x_0 > 0.$$

The sequence $\{x_n\}_{n=0}^{\infty}$ converges to a certain value $x^* > 0$. The goal here is to obtain x^* approximately.

An obvious algorithm is to start from a strictly positive initial condition x_0 and to iterate equation (1) until $|x_{n+1} - x_n|$ becomes sufficiently small.

Write a program that implements such an algorithm. Make sure you display the value of x_{n+1} at each step n. (Hint: pre-specify a stopping criterion or a convergence criterion, $xtol > 0$ (a sufficiently small number), and then stop the iteration when $|x_{n+1} - x_n| < xtol$.)

Q5 [2 marks]

Some useful classes of polynomials can be obtained recursively. Consider Chebyshev polynomials. Their recursion is given by:

- Chebyshev polynomial (of the first kind)

$$T_0(x) = 1, \quad T_1(x) = x, \text{ and } T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \text{ for n=1, 2,...}$$

For any positive integers N and $n \leq N$, $T_n(x)$ can be written as

$$T_n(x) = \sum_{i=0}^{N} \theta(n,i)x^i.$$

The goal here is to obtain a set of polynomial coefficients $\{\theta(n,i)\}_{i=0}^{N}$ for $n = 0, 1, ..., 5$ ($N = 5$).

Write a program that calculates a $(N+1)$-by-$(N+1)$ `numpy.ndarray` (or a matrix), `Tmat`, such that

- for $i, j \in \{0, ..., N\}$, `Tmat[i,j]` stores $\theta(i,j)$ i.e. a coefficient on x^j in $T_i(x)$.

(Hint: Quite obviously, the first two rows of `Tmat` are `[1, 0, 0, ...., 0]` and `[0, 1, 0, 0, ..., 0]`. Write a for loop statement to fill the remaining rows in `Tmat`.)

You can check whether your calculation is correct, by comparing your `Tmat` with the coefficients in `https://en.wikipedia.org/wiki/Chebyshev_polynomials`.