# 201800294_midterm

April 24, 2023

## 1 Midterm

```
[ ]: from IPython.core.interactiveshell import InteractiveShell
     InteractiveShell.ast_node_interactivity = 'all'
```

## 2 Problem. 1

```
[ ]: # initalizing
     func1 = "x" # define function 1
     func2 = "2*x**2" # define function 2
     func3 = "3*x**3" # define function 3

     start = round(-10, 2) # starting point
     end = round(10, 2) # ending point

     x = start # x intializing
     step = round(0.01, 2) # x step
     count = 0 # iteration counting num

     # touch .txt file with file1, file2, file3, file4
     with open('y=x.txt', 'w') as file1, \
          open('y=2x**2.txt', 'w') as file2, \
          open('y=3x**3.txt', 'w') as file3, \
          open('x data.txt', 'w') as file4:


         while x <= end: # while iteration
             file1.write(str(round(eval(func1), 2)) + ' ') # writing f1 file with␣
      ↪func1
             file2.write(str(round(eval(func2), 2)) + ' ') # wirting f2 file with␣
      ↪func2
             file3.write(str(round(eval(func3), 2)) + ' ') # writing f3 file with␣
      ↪func3
             file4.write(str(round(x, 2)) + ' ') # f4 : input x data
```

```
        count += 1

        x += step # x + 0.01
        # new line
        if count % 5 == 0:
            file1.write('\n')
            file2.write('\n')
            file3.write('\n')
            file4.write('\n')
```

## 3 Problem.2

```
[ ]: # import data from txt file
with open('x data.txt', 'r') as x_data, \
     open('y=x.txt', 'r') as y_data1, \
     open('y=2x**2.txt', 'r') as y_data2, \
     open('y=3x**3.txt', 'r') as y_data3, \
     open('y=x + 2x**2 + 3x**3.txt', 'w') as y_data:

    # while iteration
    while True:
        # imported data to list
        x_data_box = x_data.readline().strip().split() # data_x to list type

        y_list = [1] * len(x_data_box) # y_list intializiing size with x_data
        y_data1_list = list(map(float, y_data1.readline().strip().split())) #␣
 ↪y_data1 to list
        y_data2_list = list(map(float, y_data2.readline().strip().split())) #␣
 ↪y_data2 to list
        y_data3_list = list(map(float, y_data3.readline().strip().split())) #␣
 ↪y_data3 to list

        # while break
        if not x_data_box:
            break


        # change y_list to y_data sum
        for j in range(len(y_list)):
            y_list[j] = round((y_data1_list[j] + y_data2_list[j] +␣
 ↪y_data3_list[j]), 2) # summation y_data1, y_data2, y_data3
            y_data.write(x_data_box[j] + ' ' + str(y_list[j]) + '\n')
```

## 4 Problem.3

[ ]:
```
# define Max, Min
def getMax(numbers):
    result = -100000000
    for number in numbers:
        if result < number:
            result = number
    return result

def getMin(numbers):
    result = 100000000
    for number in numbers:
        if result > number:
            result = number
    return result

# make data
f = "-40*x**2+x**4" # define func

start = round(-10, 2) # starting point
end = round(10, 2) # ending point

x = start # initailizing x
step = 0.01 # x step

# make data box
y_list = []
x_list = []
val_box = []

# fill x_list, y_list, val_box
while x <= (end):
    x_list.append(round(x, 2))
    y_list.append(round(eval(f), 2))
    x += step

for i in range(len(x_list)):
    val_box.append([x_list[i], y_list[i]])

# decomposion value by interval

    # interval (-1,1)
x_list_1 = x_list[901:1100]
x_list_2 = x_list[501:1500]
x_list_3 = x_list[1:-1]
    # interval (-5,5)
```

```
y_list_1 = y_list[901:1100]
y_list_2 = y_list[501:1500]
y_list_3 = y_list[1:-1]
    # interval (-10,10)
val_box_1 = val_box[901:1100]
val_box_2 = val_box[501:1500]
val_box_3 = val_box[1:-1]


# get Max, Min value
    # interval (-1,1)
Max_val_1 = getMax(y_list_1)
Min_val_1 = getMin(y_list_1)
    # interval (-5,5)
Max_val_2 = getMax(y_list_2)
Min_val_2 = getMin(y_list_2)
    # interval (-10,10)
Max_val_3 = getMax(y_list_3)
Min_val_3 = getMin(y_list_3)

# Make result list
    # interval (-1,1)
Max_result_1 = []
Min_result_1 = []
    # interval (-5,5)
Max_result_2 = []
Min_result_2 = []
    # interval (-10,10)
Max_result_3 = []
Min_result_3 = []

    # interval (-1,1)
for i in range(len(y_list_1)):
    if y_list_1[i] == Max_val_1:
        Max_result_1.append(val_box_1[i])
    elif y_list_1[i] == Min_val_1:
        Min_result_1.append(val_box_1[i])

    # interval (-5,5)
for i in range(len(y_list_2)):
    if y_list_2[i] == Max_val_2:
        Max_result_2.append(val_box_2[i])
    elif y_list_2[i] == Min_val_2:
        Min_result_2.append(val_box_2[i])

     # interval (-10,10)
for i in range(len(y_list_3)):
    if y_list_3[i] == Max_val_3:
        Max_result_3.append(val_box_3[i])
    elif y_list[i] == Min_val_3:
        Min_result_3.append(val_box_3[i])

# export min and max data by intervals
with open('interval(-1,1).txt', 'w') as file_1, \
     open('interval(-5,5).txt', 'w') as file_2, \
     open('interval(-10,10).txt', 'w') as file_3 :

        # interval (-1,1)
    for i in range(len(Max_result_1)):
        file_1.write("x : " + str( Max_result_1[i][0]) + ", " + "max value = "␣
 ↪+ str(Max_result_1[i][1]) + "\n")
    for j in range(len(Min_result_1)):
        file_1.write("x : " + str( + Min_result_1[j][0]) + ", " + "min value =␣
 ↪" + str(Min_result_1[j][1]) + "\n")

        # interval (-5,5)
    for i in range(len(Max_result_2)):
        file_2.write("x: " + str(Max_result_2[i][0]) + ", " + "max value = " +␣
 ↪str(Max_result_2[i][1]) + "\n")
    for j in range(len(Min_result_1)):
        file_2.write("x: " + str( Min_result_2[j][0]) + ", " + "min value = " +␣
 ↪str(Min_result_2[j][1]) + "\n")

        # interval (-10,10)
    for i in range(len(Max_result_3)):
        file_3.write("x: " + str(Max_result_3[i][0]) + ", " + "max value = " +␣
 ↪str(Max_result_3[i][1]) + "\n")
    for j in range(len(Min_result_3)):
        file_3.write("x: " + str(Min_result_3[j][0]) + ", " + "min value = " +␣
 ↪str(Min_result_3[j][1]) + "\n")
```

---

Reference * Title: Physics Programming Lecture Note (INU) * Author: Jeongwoo Kim, Ph.D. * Availability: https://sites.google.com/view/jeongwookim

---