

# 201800294\_midterm

April 24, 2023

Copyright (C) 2023 201800294\_DongilKim All rights reserved (<https://KimTein.github.io>)

## 1 Midterm

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

## 2 Problem. 1

```
[ ]: # initializing
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')
```

## [결과창]

 x data - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
-10 -9.99 -9.98 -9.97 -9.96
-9.95 -9.94 -9.93 -9.92 -9.91
-9.9 -9.89 -9.88 -9.87 -9.86
-9.85 -9.84 -9.83 -9.82 -9.81
-9.8 -9.79 -9.78 -9.77 -9.76
-9.75 -9.74 -9.73 -9.72 -9.71
-9.7 -9.69 -9.68 -9.67 -9.66
-9.65 -9.64 -9.63 -9.62 -9.61
-9.6 -9.59 -9.58 -9.57 -9.56
-9.55 -9.54 -9.53 -9.52 -9.51
-9.5 -9.49 -9.48 -9.47 -9.46
-9.45 -9.44 -9.43 -9.42 -9.41
-9.4 -9.39 -9.38 -9.37 -9.36
-9.35 -9.34 -9.33 -9.32 -9.31
-9.3 -9.29 -9.28 -9.27 -9.26
-9.25 -9.24 -9.23 -9.22 -9.21
-9.2 -9.19 -9.18 -9.17 -9.16
-9.15 -9.14 -9.13 -9.12 -9.11
-9.1 -9.09 -9.08 -9.07 -9.06
```

 y=x - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
-10 -9.99 -9.98 -9.97 -9.96
-9.95 -9.94 -9.93 -9.92 -9.91
-9.9 -9.89 -9.88 -9.87 -9.86
-9.85 -9.84 -9.83 -9.82 -9.81
-9.8 -9.79 -9.78 -9.77 -9.76
-9.75 -9.74 -9.73 -9.72 -9.71
-9.7 -9.69 -9.68 -9.67 -9.66
-9.65 -9.64 -9.63 -9.62 -9.61
-9.6 -9.59 -9.58 -9.57 -9.56
-9.55 -9.54 -9.53 -9.52 -9.51
-9.5 -9.49 -9.48 -9.47 -9.46
-9.45 -9.44 -9.43 -9.42 -9.41
-9.4 -9.39 -9.38 -9.37 -9.36
-9.35 -9.34 -9.33 -9.32 -9.31
-9.3 -9.29 -9.28 -9.27 -9.26
-9.25 -9.24 -9.23 -9.22 -9.21
-9.2 -9.19 -9.18 -9.17 -9.16
-9.15 -9.14 -9.13 -9.12 -9.11
-9.1 -9.09 -9.08 -9.07 -9.06
```

y=2x2 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

200 199.6 199.2 198.8 198.4  
198.01 197.61 197.21 196.81 196.42  
196.02 195.62 195.23 194.83 194.44  
194.05 193.65 193.26 192.86 192.47  
192.08 191.69 191.3 190.91 190.52  
190.13 189.74 189.35 188.96 188.57  
188.18 187.79 187.4 187.02 186.63  
186.25 185.86 185.47 185.09 184.7  
184.32 183.94 183.55 183.17 182.79  
182.41 182.02 181.64 181.26 180.88  
180.5 180.12 179.74 179.36 178.98  
178.61 178.23 177.85 177.47 177.1  
176.72 176.34 175.97 175.59 175.22  
174.85 174.47 174.1 173.72 173.35  
172.98 172.61 172.24 171.87 171.5  
171.13 170.76 170.39 170.02 169.65  
169.28 168.91 168.54 168.18 167.81  
167.45 167.08 166.71 166.35 165.98  
165.62 165.26 164.89 164.53 164.17

y=3x3 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

-3000 -2991.01 -2982.04 -2973.08 -2964.14  
-2955.22 -2946.32 -2937.44 -2928.57 -2919.73  
-2910.9 -2902.09 -2893.29 -2884.51 -2875.76  
-2867.01 -2858.29 -2849.59 -2840.9 -2832.23  
-2823.58 -2814.94 -2806.32 -2797.72 -2789.14  
-2780.58 -2772.03 -2763.5 -2754.99 -2746.5  
-2738.02 -2729.56 -2721.12 -2712.69 -2704.29  
-2695.9 -2687.52 -2679.17 -2670.83 -2662.51  
-2654.21 -2645.92 -2637.65 -2629.4 -2621.17  
-2612.95 -2604.75 -2596.57 -2588.4 -2580.26  
-2572.13 -2564.01 -2555.91 -2547.83 -2539.77  
-2531.73 -2523.7 -2515.69 -2507.69 -2499.71  
-2491.75 -2483.81 -2475.88 -2467.97 -2460.08  
-2452.2 -2444.34 -2436.5 -2428.67 -2420.86  
-2413.07 -2405.3 -2397.54 -2389.79 -2382.07  
-2374.36 -2366.67 -2358.99 -2351.33 -2343.69  
-2336.06 -2328.45 -2320.86 -2313.29 -2305.73  
-2298.18 -2290.66 -2283.15 -2275.65 -2268.17  
-2260.71 -2253.27 -2245.84 -2238.43 -2231.03

[알고리즘]:

1. 각각의 함수를 정의하고 시작 구간과 끝 구간을 정의한다. (이때 소수점 2 자리로 끊었다.)
2. 반복횟수에 따른 줄바꿈을 넣기 위해 count 변수 초기화한다.
3. x 값은 step 을 더하며 갱신되며 각 x 에 해당되는 함수의 결과를 eval() 사용한다.
4. 반복문을 시행하며 매 시행 시 count 값이 갱신되고 5 의 배수가 될 때 마다 줄바꿈을 정의한다.

[예상 결과 및 실제 결과]:

결과창과 동일한 형태를 기대하였으며 주의할 점으로는 기계적인 오차로 인한 차이가 발생할 수 있다. 따라서 정확한 수치적인 값을 얻기 위해서는 원하는 조건이 필요로 하며 이를 위한 보완을 요구한다.

# 레포트 작성을 위해 파일을 옮기는 과정으로 실행 결과의 파일명을 간단히 작성함. 소스코드로 생성된 파일은 지장 없음.

### 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')
```

[결과창]

y=x1+x2+x3 - Windows 메모장	
파일(F)	편집(E) 서식(O) 보기(V) 도움말(H)
9.83	3052.68
9.84	3061.78
9.85	3070.9
9.86	3080.06
9.87	3089.21
9.88	3098.4
9.89	3107.6
9.9	3116.82
9.91	3126.06
9.92	3135.3
9.93	3144.58
9.94	3153.87
9.95	3163.17
9.96	3172.5
9.97	3181.85
9.98	3191.22
9.99	3200.6
10.0	3210.0

[알고리즘]:

1. 앞서 각각의 x 값과 해당되는 함수값을 출력한 파일을 읽어온다.
2. 최종 결과값을 위한 y\_data 파일을 출력하기로 한다.
3. 불러온 데이터들의 값만을 가져오기 위해 반복문을 통하여 한줄씩 읽어와 전처리를 한다.  
(`readline()`, `strip()`, `split()` 을 사용)
4. 한번의 반복문 시행 시 불러온 데이터들을 list 형태로 변환하며 append 해준다. 이때 우리가 원하는 y\_data 의 리스트 크기는 입력값의 크기와 동일하게 맞춰준다.
5. 각 함수의 결과값들을 리스트 형태로 불러왔으므로 이를 y\_data 리스트에 summation 하며 하나씩 append 하여준다.
6. 이후, y\_list 의 각 인덱스 별로 입력값과 출력값을 txt 파일로 export 한다.

[예상 결과 및 실제 결과]:

결과창과 동일한 형태를 기대하였으며 다만, 소수점 자리를 맞추는 과정에 수치적인 오차가 쌓일 수 있다.



## 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]])

# decomposition 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")

```

## [결과창]

```
interval(-1,1) - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
x : -0.01, max value = -0.0
x : -0.0, max value = -0.0
x : 0.01, max value = -0.0
x : -0.99, min value = -38.24
x : 0.99, min value = -38.24
```

```
interval(-5,5) - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
x: -0.01, max value = -0.0
x: -0.0, max value = -0.0
x: 0.01, max value = -0.0
x: -4.48, min value = -400.0
x: -4.47, min value = -400.0
```

interval(-10,10) - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

x: -9.99, max value = 5968.06

x: 9.99, max value = 5968.06

x: -4.47, min value = -400.0

x: -4.46, min value = -399.99

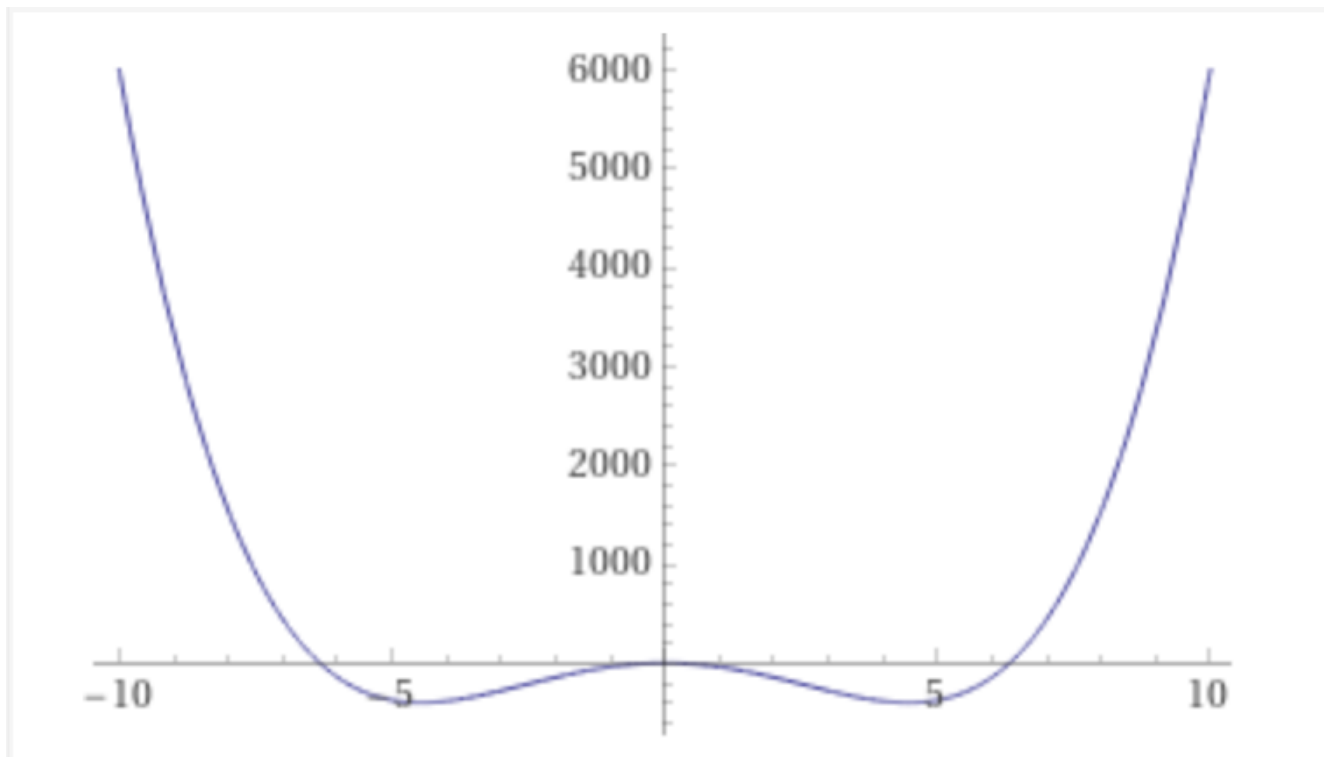
x: 4.48, min value = -400.0

x: 4.49, min value = -399.97

[알고리즘]:

1. Built-in 함수를 대신하기 위해 max, min 함수를 새롭게 정의한다.
2. 구간을 최대 범위인 -10 부터 10 까지 정의하며 각 변수를 초기화 한다.
3. 위 구간에 해당하는 함수 값들을 val\_box 안에 저장한다.
4. 문제에 제시된 구간별로 값들을 slicing 하여 새롭게 정의한다.
5. 이후, 위에서 정의한 max, min 함수를 이용하여 각 구간의 최대값과 최소값을 구한다.
6. 각 구간별로 위에서 구한 최대값과 최소값에 해당하는 인덱스를 추출하여 Result 리스트 안에 정의한다.
7. 이후, 제시된 형태로 txt 파일로 출력한다.

[예상 결과 및 실제 결과]



문제에 제시된 함수를 plot 하면 위와 같은 곡선이 그려진다.

각 최대값과 최소값에 해당하는  $x$  값은 대략적으로 우리는 알 수 있다.

허나, 실제 결과에서는 -4.48, -4.47 과 같이  $x$  값이 하나로 정의 되지 않고 여러 개가 나오는 것을 알 수 있다. 이는, 파이썬에서 내장된 기본 값들을 사용할 때 기계적인 오차가 발생하였기 때문이다. 결과창에서 값을 표기하기 위해 변수를 지정할 때 round()함수를 사용하였지만 수치적인 계산을 하는 과정 중에선 그렇지 않다. 따라서, 이를 보완하기 위해선 함수를 각 항별로 쪼갠 후 작은 값의 계산을 피하는 과정을 요한다.

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

---

Copyright (C) 2023 201800294\_DongilKim All rights reserved (<https://KimTein.github.io>)