

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

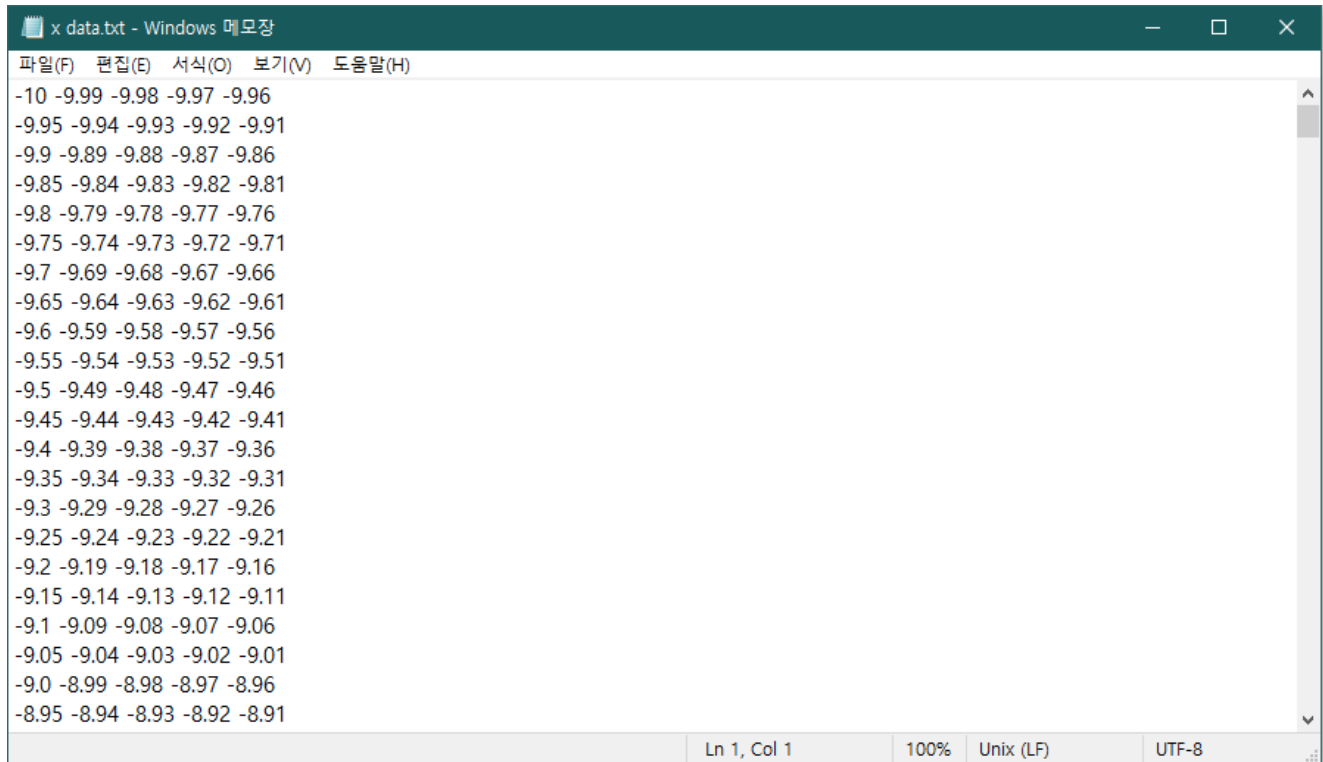
# touch .txt file with file1, file2, file3, file4
with open('y=x.txt', 'w') as file1, \
    open('y=2x**2.txt', 'w') as file2, \
    open('y=3x**3.txt', 'w') as file3, \
    open('x data.txt', 'w') as file4:

    while x <= end: # while iteration
        file1.write(str(round(eval(func1), 2)) + ' ') # writing f1 file with
        ↪ func1
        file2.write(str(round(eval(func2), 4)) + ' ') # wirting f2 file with
        ↪ func2
        file3.write(str(round(eval(func3), 6)) + ' ') # writing f3 file with
        ↪ func3
        file4.write(str(round(x, 2)) + ' ') # f4 : input x data
```

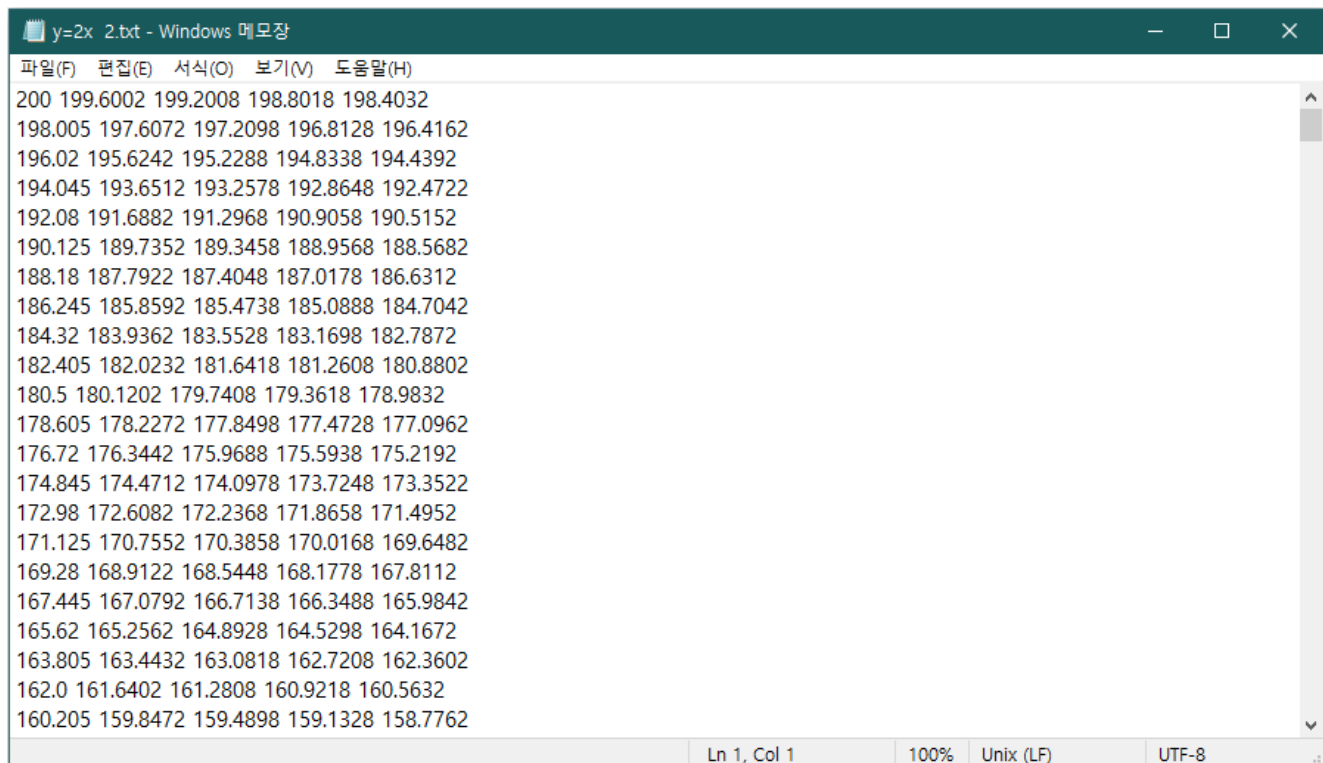
```
count += 1

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.txt - 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
-9.05 -9.04 -9.03 -9.02 -9.01
-9.0 -8.99 -8.98 -8.97 -8.96
-8.95 -8.94 -8.93 -8.92 -8.91
Ln 1, Col 1 100% Unix (LF) UTF-8
```



```
y=2x 2.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
200 199.6002 199.2008 198.8018 198.4032
198.005 197.6072 197.2098 196.8128 196.4162
196.02 195.6242 195.2288 194.8338 194.4392
194.045 193.6512 193.2578 192.8648 192.4722
192.08 191.6882 191.2968 190.9058 190.5152
190.125 189.7352 189.3458 188.9568 188.5682
188.18 187.7922 187.4048 187.0178 186.6312
186.245 185.8592 185.4738 185.0888 184.7042
184.32 183.9362 183.5528 183.1698 182.7872
182.405 182.0232 181.6418 181.2608 180.8802
180.5 180.1202 179.7408 179.3618 178.9832
178.605 178.2272 177.8498 177.4728 177.0962
176.72 176.3442 175.9688 175.5938 175.2192
174.845 174.4712 174.0978 173.7248 173.3522
172.98 172.6082 172.2368 171.8658 171.4952
171.125 170.7552 170.3858 170.0168 169.6482
169.28 168.9122 168.5448 168.1778 167.8112
167.445 167.0792 166.7138 166.3488 165.9842
165.62 165.2562 164.8928 164.5298 164.1672
163.805 163.4432 163.0818 162.7208 162.3602
162.0 161.6402 161.2808 160.9218 160.5632
160.205 159.8472 159.4898 159.1328 158.7762
Ln 1, Col 1 100% Unix (LF) UTF-8
```

y=3x 3.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
-3000 -2991.008997 -2982.035976 -2973.080919 -2964.143808
-2955.224625 -2946.323352 -2937.439971 -2928.574464 -2919.726813
-2910.897 -2902.085007 -2893.290816 -2884.514409 -2875.755768
-2867.014875 -2858.291712 -2849.586261 -2840.898504 -2832.228423
-2823.576 -2814.941217 -2806.324056 -2797.724499 -2789.142528
-2780.578125 -2772.031272 -2763.501951 -2754.990144 -2746.495833
-2738.019 -2729.559627 -2721.117696 -2712.693189 -2704.286088
-2695.896375 -2687.524032 -2679.169041 -2670.831384 -2662.511043
-2654.208 -2645.922237 -2637.653736 -2629.402479 -2621.168448
-2612.951625 -2604.751992 -2596.569531 -2588.404224 -2580.256053
-2572.125 -2564.011047 -2555.914176 -2547.834369 -2539.771608
-2531.725875 -2523.697152 -2515.685421 -2507.690664 -2499.712863
-2491.752 -2483.808057 -2475.881016 -2467.970859 -2460.077568
-2452.201125 -2444.341512 -2436.498711 -2428.672704 -2420.863473
-2413.071 -2405.295267 -2397.536256 -2389.793949 -2382.068328
-2374.359375 -2366.667072 -2358.991401 -2351.332344 -2343.689883
-2336.064 -2328.454677 -2320.861896 -2313.285639 -2305.725888
-2298.182625 -2290.655832 -2283.145491 -2275.651584 -2268.174093
-2260.713 -2253.268287 -2245.839936 -2238.427929 -2231.032248
-2223.652875 -2216.289792 -2208.942981 -2201.612424 -2194.298103
-2187.0 -2179.718097 -2172.452376 -2165.202819 -2157.969408
-2150.752125 -2143.550952 -2136.365871 -2129.196864 -2122.043913
```

Ln 1, Col 1 100% Unix (LF) UTF-8

y=x.txt - 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
-9.05 -9.04 -9.03 -9.02 -9.01
-9.0 -8.99 -8.98 -8.97 -8.96
-8.95 -8.94 -8.93 -8.92 -8.91
```

Ln 1, Col 1 100% Unix (LF) UTF-8

[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 각 계산 값의 유효숫자를 고려하였다.

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:

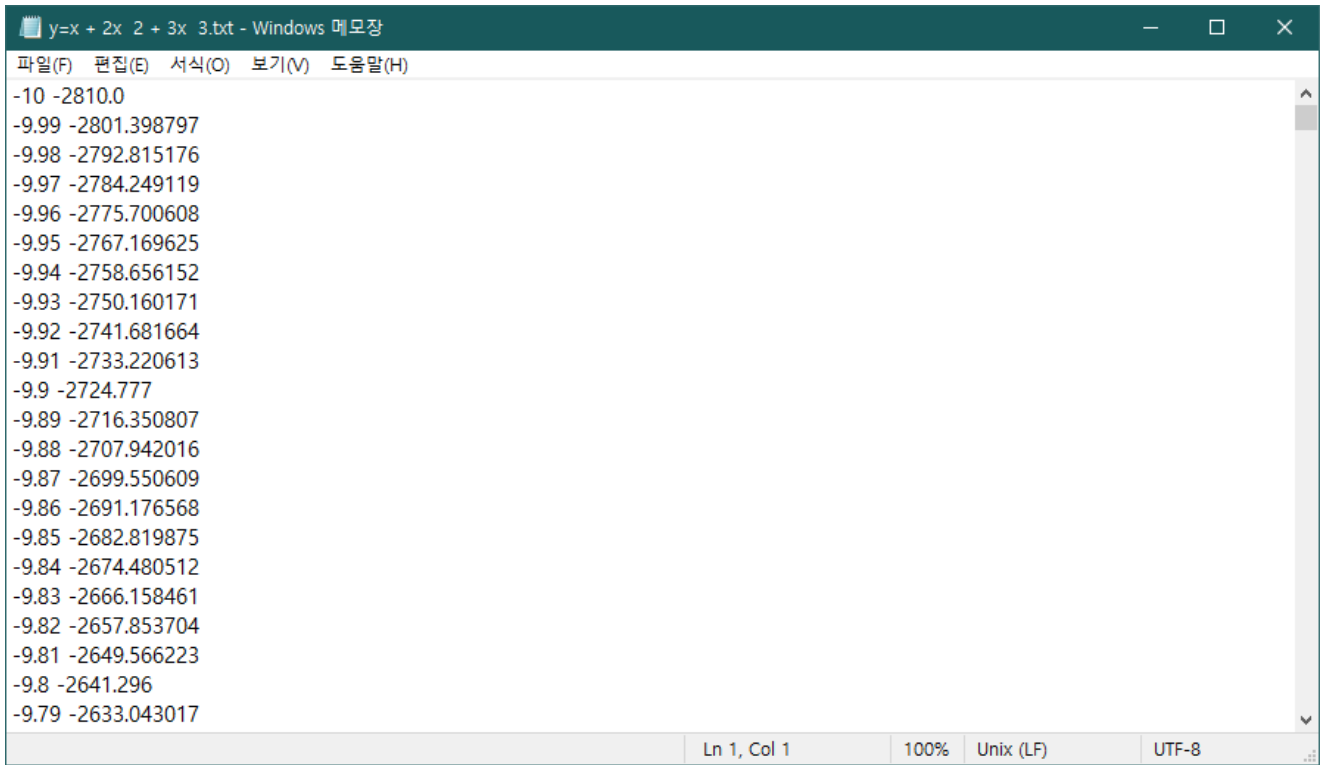
    # while iteration
    while True:
        # imported data to list
        x_data_box = x_data.readline().strip().split() # data_x to list type

        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((round(y_data1_list[j], 2) +
        ↪ round(y_data2_list[j], 4) + round(y_data3_list[j], 6)), 6) # summation
        ↪ y_data1, y_data2, y_data3
        y_data.write(x_data_box[j] + ' ' + str(y_list[j]) + '\n')
```

[결과창]



```
y=x + 2x^2 + 3x 3.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
-10 -2810.0
-9.99 -2801.398797
-9.98 -2792.815176
-9.97 -2784.249119
-9.96 -2775.700608
-9.95 -2767.169625
-9.94 -2758.656152
-9.93 -2750.160171
-9.92 -2741.681664
-9.91 -2733.220613
-9.9 -2724.777
-9.89 -2716.350807
-9.88 -2707.942016
-9.87 -2699.550609
-9.86 -2691.176568
-9.85 -2682.819875
-9.84 -2674.480512
-9.83 -2666.158461
-9.82 -2657.853704
-9.81 -2649.566223
-9.8 -2641.296
-9.79 -2633.043017
Ln 1, Col 1 100% Unix (LF) UTF-8
```

[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]:

예상한 결과와 동일한 결과가 나왔다.

4 Problem.3

```
[ ]: # define Max, Min
def getMax(numbers):
    result = -1000000000
    for number in numbers:
        if result < number:
            result = number
    return result

def getMin(numbers):
    result = 1000000000
    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 = round(0.01, 2) # 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), 8))
    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 val_box_1[i][1] == Max_val_1:
        Max_result_1.append(val_box_1[i])
    elif val_box_1[i][1] == Min_val_1:
        Min_result_1.append(val_box_1[i])

    # interval (-5,5)
for i in range(len(y_list_2)):
    if val_box_2[i][1] == Max_val_2:
        Max_result_2.append(val_box_2[i])
    elif val_box_2[i][1] == Min_val_2:
        Min_result_2.append(val_box_2[i])

    # interval (-10,10)
for i in range(len(y_list_3)):

```

```

    if val_box_3[i][1] == Max_val_3:
        Max_result_3.append(val_box_3[i]
    ) elif val_box_3[i][1] ==
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")

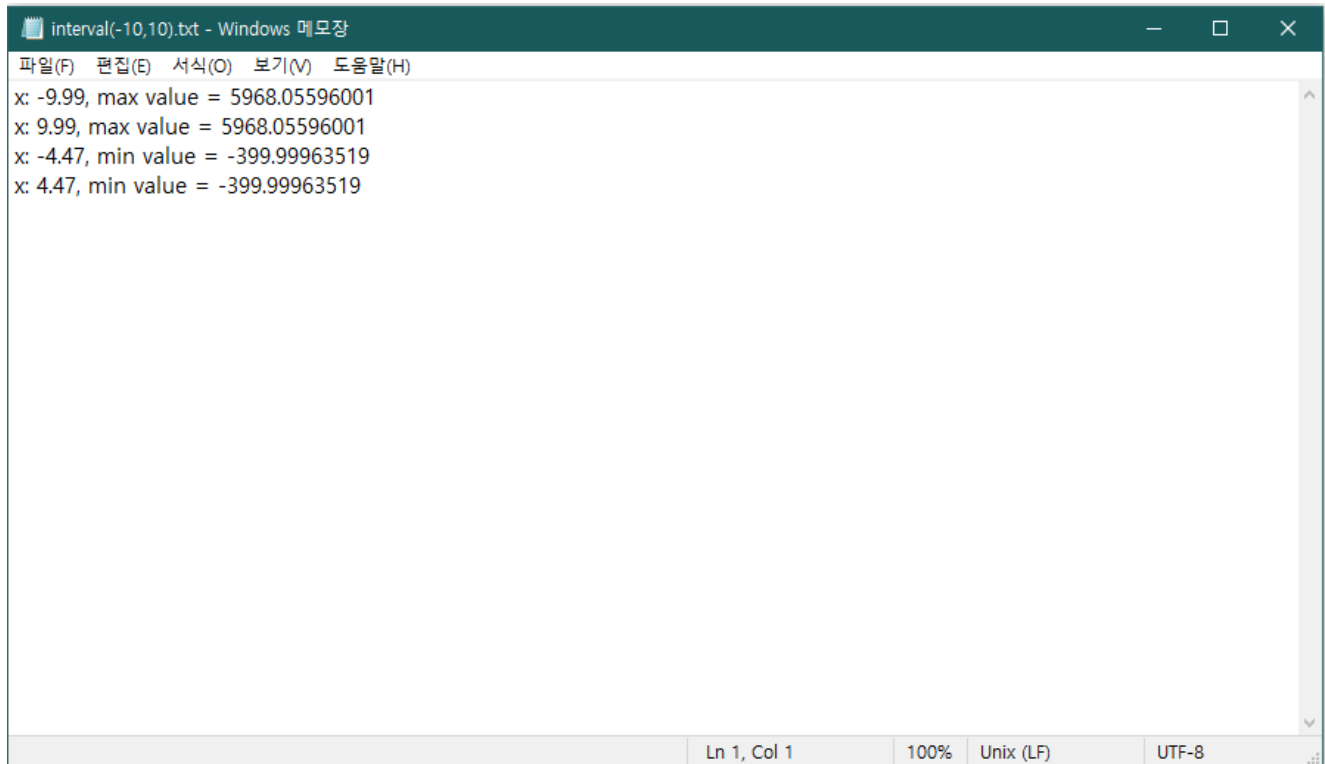
```

[결과창]

The image displays two screenshots of a Windows Notepad application. The top window, titled 'Interval(-1,1).txt - Windows 메모장', contains the following text: '파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)', 'x : -0.0, max value = -0.0', 'x : -0.99, min value = -38.24340399', and 'x : 0.99, min value = -38.24340399'. The bottom window, titled 'interval(-5,5).txt - Windows 메모장', contains the following text: '파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)', 'x: -0.0, max value = -0.0', 'x: -4.47, min value = -399.99963519', and 'x: 4.47, min value = -399.99963519'. Both windows show a status bar at the bottom with 'Ln 1, Col 1', '100%', 'Unix (LF)', and 'UTF-8'.

```
Interval(-1,1).txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
x : -0.0, max value = -0.0
x : -0.99, min value = -38.24340399
x : 0.99, min value = -38.24340399
Ln 1, Col 1 100% Unix (LF) UTF-8

interval(-5,5).txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
x: -0.0, max value = -0.0
x: -4.47, min value = -399.99963519
x: 4.47, min value = -399.99963519
Ln 1, Col 1 100% Unix (LF) UTF-8
```

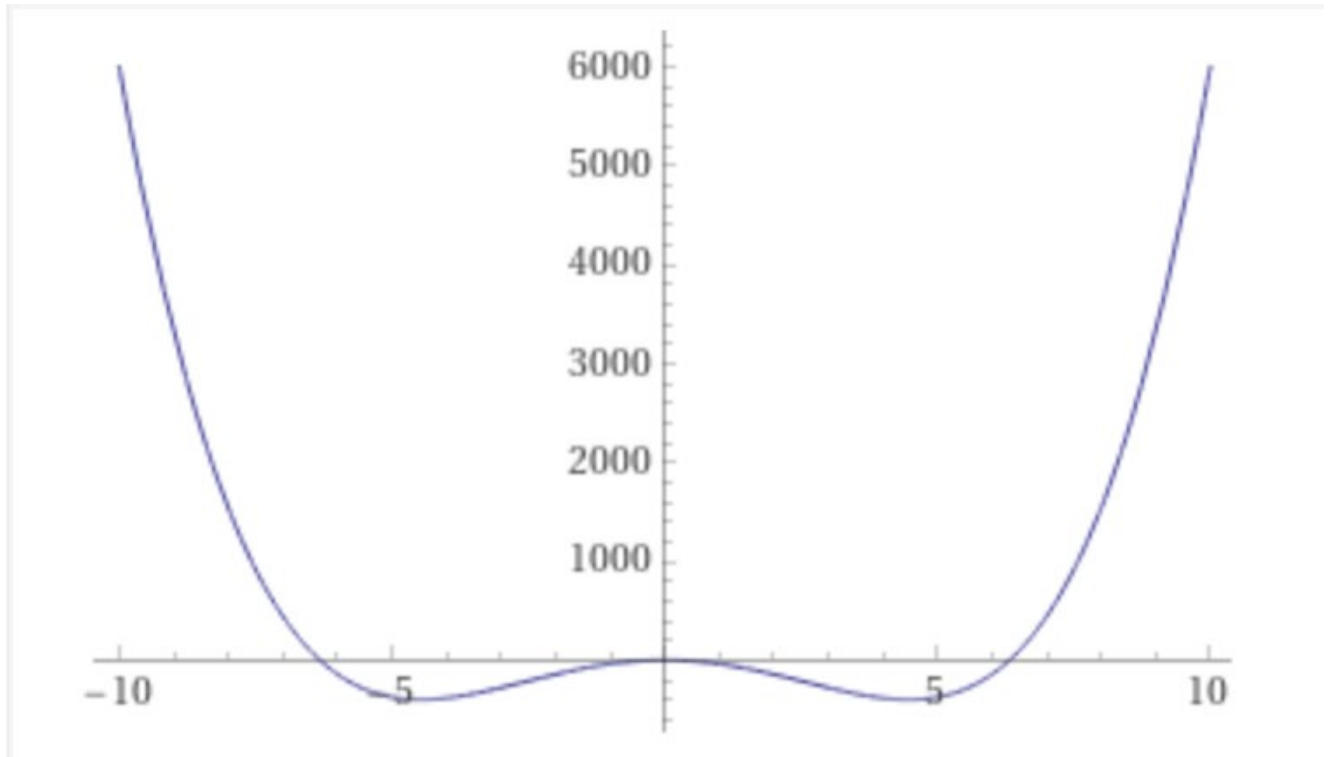


```
interval(-10,10).txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
x: -9.99, max value = 5968.05596001
x: 9.99, max value = 5968.05596001
x: -4.47, min value = -399.99963519
x: 4.47, min value = -399.99963519
Ln 1, Col 1 100% Unix (LF) UTF-8
```

[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.

이를 통해 각 구간별로의 값을 유추하였을 때 실제 결과와 동일한 값이 나온 점을 확인할 수 있다. 다만, x 의 step을 0.01로 잡았기에 함수의 원래 최소값이 나오지 않는다.

(위 함수의 최소값이 나오는 x 는 $+\text{root}(20)$, $-\text{root}(20)$ 이다.)

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>)