

과제 1

Assignment 1

391040277 | Nathan Cho | 조나단

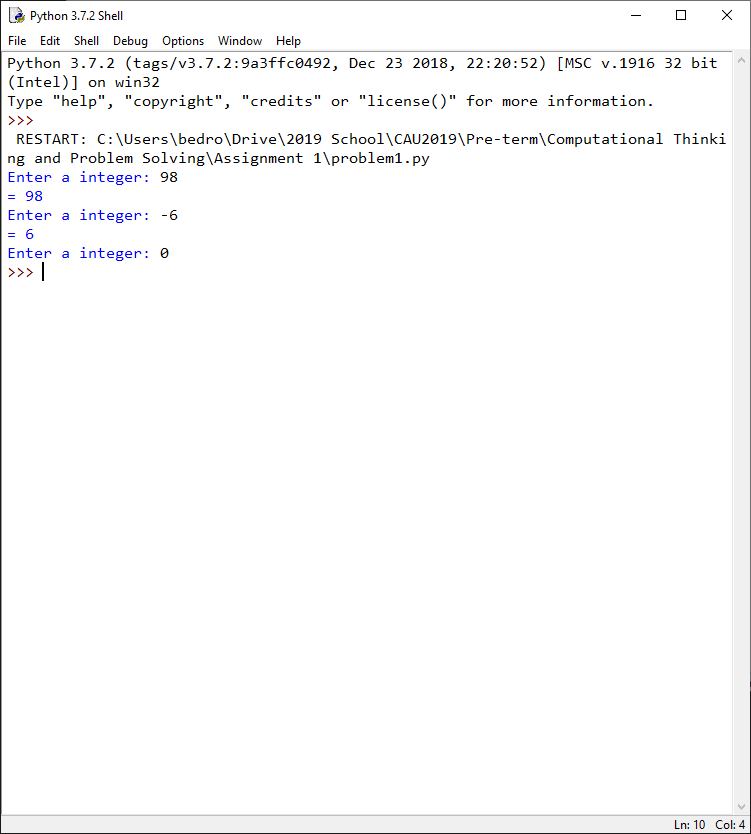
Computational Thinking and Problem Solving | 2019-02-09

# Problem 1. 절대값 출력하기

## 소스 코드

1. # problem 1: get a user input of a integer and print its absolute value
3. **while** True:
4. i = int(input("Enter a integer: "))
5. **if** i > 0:
6. **print**("=", i)
7. **elif** i < 0:
8. **print**("=", -i)
9. **else**:
10. **break**

## 결과 화면



## 설명

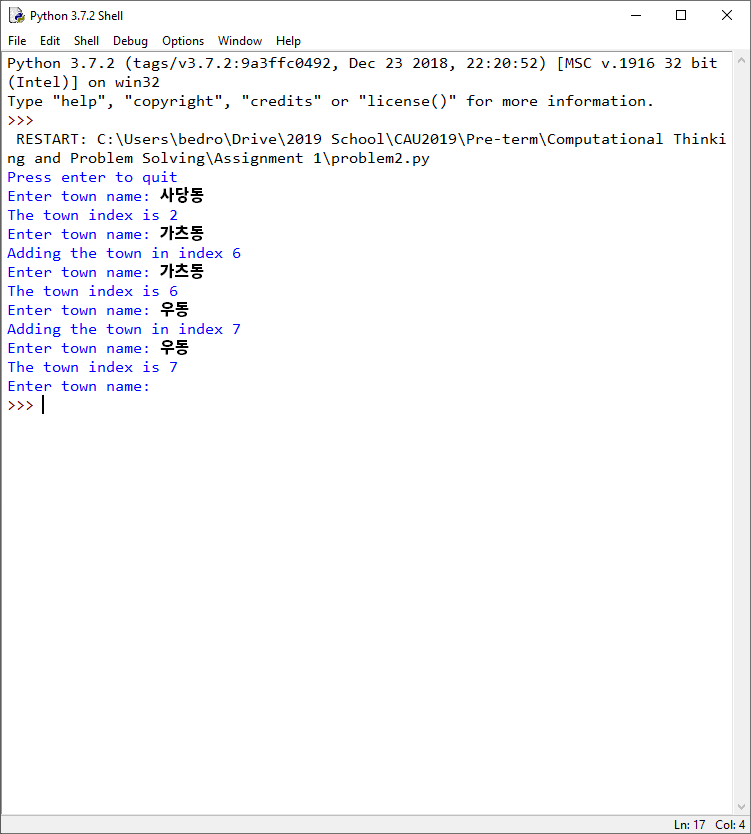
양수일 경우에는 바로 출력하고 음수일 경우에는 부호를 바꾸어서 출력하는 간단한 알고리즘이다.

# Problem 2. 리스트에서 데이터 찾기

## 소스 코드

1. # problem 2: finding and inserting a data to list
3. town = ['흑석동', '사당동', '상도동', '노량진동', '규동']
5. **print**("Press enter to quit")
7. **while** True:
8. i = input("Enter town name: ")
9. **if** i **is** "":
10. **break**
11. **else**:
12. **if** i **in** town:
13. **print**("The town index is", town.index(i) + 1)
14. **else**:
15. town.append(i)
16. **print**("Adding the town in index", len(town))

## 결과 화면



## 설명

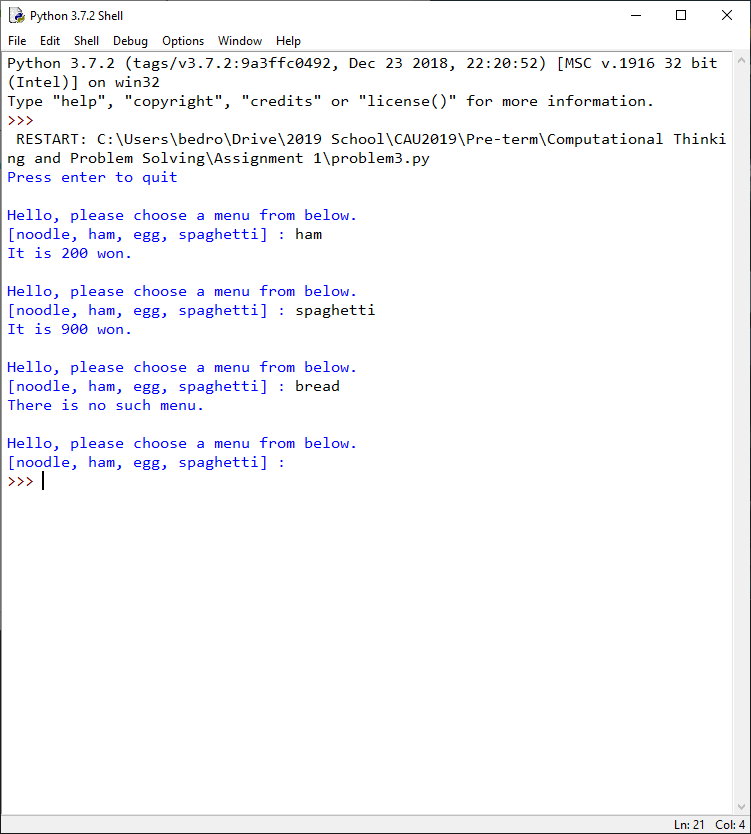
if 와 in 구문을 사용해서 입력한 동이 리스트에 존재하는지 확인한 후에 있을 경우에는 index 함수를 사용해서 몇 번째 동인지를 출력하고, 없을 경우에는 append 함수를 사용해서 리스트의 가장 마지막에 추가하는 알고리즘이다.

# Problem 3. 식당 메뉴 표

## 소스 코드

1. # problem 3: menu - price robot
3. menu\_name = ['noodle','ham', 'egg', 'spaghetti']
4. menu\_price = [500, 200, 100, 900]
6. **print**("Press enter to quit")
8. **while** True:
9. **print**("\nHello, please choose a menu from below.")
10. selection = input("[noodle, ham, egg, spaghetti] : ")
12. **if** selection == "":
13. **break**
15. **if** selection **in** menu\_name:
16. **print**("It is", menu\_price[menu\_name.index(selection)], "won.")
17. **else**:
18. **print**("There is no such menu.")

## 결과 화면



## 설명

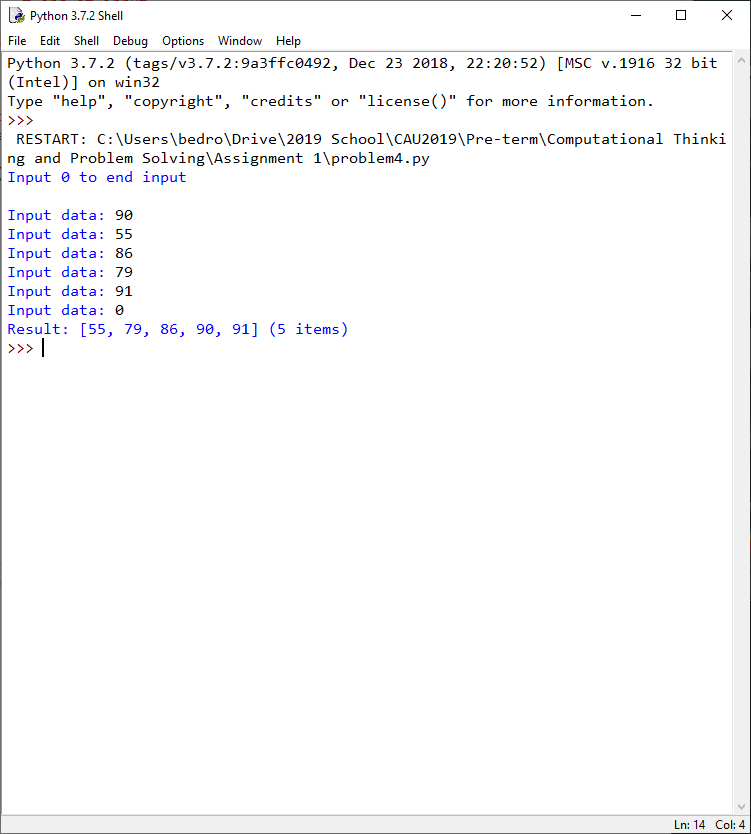
list 를 사용해서 음식 이름과 가격의 리스트를 만들어 사용했다. 가격을 불러오기 전에 미리 메뉴에 입력한 값이 존재하는지를 확인해 ValueError가 일어나지 않도록 제작했다.

# Problem 4. 오름차순 출력

## 소스 코드

1. # problem 4: ascending sort of user input
3. data = []
5. **print**("Input 0 to end input\n")
7. **while** True:
8. i = int(input("Input data: "))
10. **if** i == 0:
11. # end of input
12. **break**
13. **else**:
14. # append data to the list
15. data.append(i)
17. # sort the list to ascending order
18. data.sort()
20. **print**("Result:", data, "(" + str(len(data)) + " items)")

## 결과 화면



## 설명

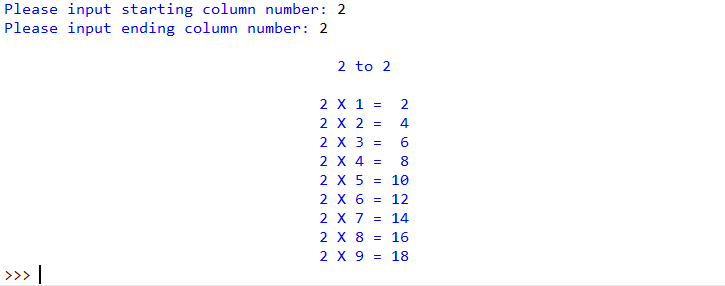
사용자의 입력으로 만들어진 리스트를 list자료형의 sort 함수를 통해 오름차순으로 정렬하고 출력하는 알고리즘이다. 마지막에 개수를 출력하는 부분에는 정수를 반환하는 len 함수를 바로 문자열에 더할 수 없기 때문에 str (문자열) 자료형으로 변환해 문자열끼리 더한다.

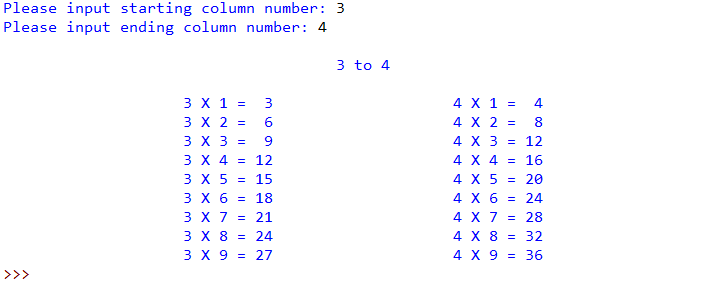
# Problem 5. 구구단

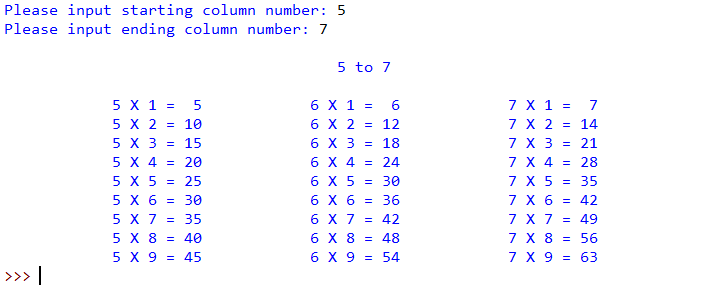
## 소스 코드

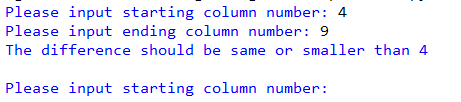
1. # problem 5: print the multiplting table according to user input
3. # corrects the user input value
4. **while** True:
5. start = int(input("Please input starting column number: "))
6. end = int(input("Please input ending column number: "))
8. # check for invalid input
9. **if** start > 9 **or** start < 1 **or** end > 9 **or** end < 1:
10. **print**("Invalid input - Both column numbers should be between 1 and 9")
11. **continue**
13. **if** end - start > 4:
14. **print**("The difference should be same or smaller than 4\n")
15. **continue**
16. **elif** end - start < 0:
17. **print**("Ending column should be bigger than starting column\n")
18. **continue**
19. **else**:
20. **break**
22. window\_width = 80 # seems it's default for IDLE on startup
23. column\_width = 10
24. column\_count = end - start + 1
25. column\_margin = int((window\_width - column\_width\*column\_count) / (column\_count + 1))
27. **print**('\n', " "\*int(window\_width/2 - 3), start, " to ", end, '\n', sep='')
29. **for** i **in** range(1, 10):
30. # print the blanks to start the first column
32. **for** j **in** range(start, end + 1):
33. expression = str(j) + " X " + str(i) + " = "
34. answer = i\*j
36. **if** int(answer / 10) == 0:
37. # the number is 1 digit, needs an extra space
38. expression += " "
40. # print the expression and answer
41. **print**(" "\*column\_margin, expression, answer, end='', sep='')
43. # line break
44. **print**()

## 결과 화면









4단 이상을 출력할 수 없다는 오류 메시지



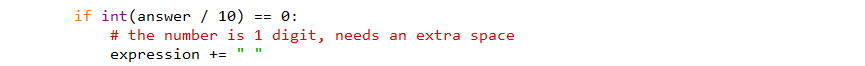
시작이 끝보다 클 수 없다는 오류 메시지



범위를 벗어나는 수에 대한 오류 메시지

## 설명

print 함수가 한 줄마다 출력하는 것을 고려해 구구단의 단을 한번에 출력하는 것이 아니라 각 단의 식들을 하나씩 출력하는 방식의 알고리즘이다. **window\_width** 라는 변수를 통해서 사용자의 창 크기에 맞추어 가운데 정렬이 되도록 제작하였다.



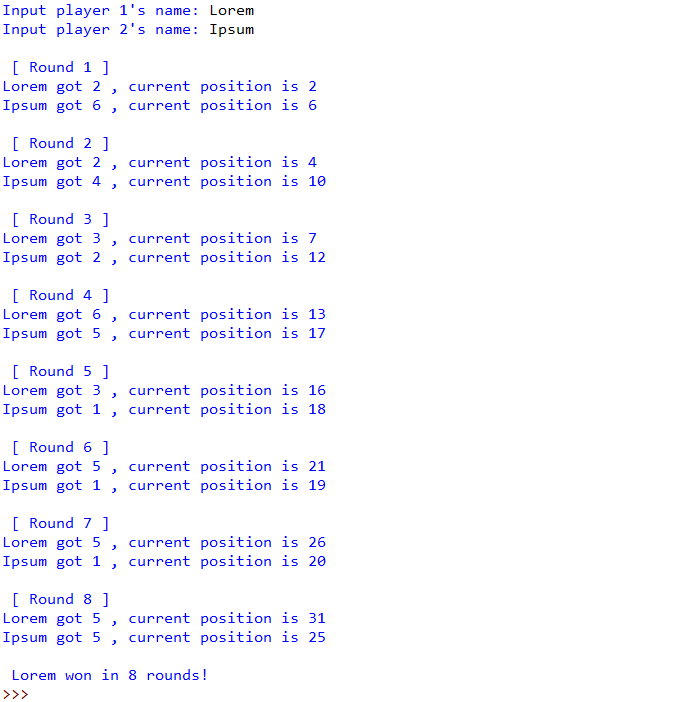
한 자리 수와 두 자리 수가 같이 표시됨에 따라 뒤 단이 정렬되지 않는 문제점을 해결하기 위해 한 자리 수일 경우에는 숫자 앞에 공백을 하나 넣어 정렬되도록 제작하였다.

# Problem 6. 경주 게임

## 소스 코드

1. # problem 6: dice based racing game
3. **from** random **import** randint
5. p1 = input("Input player 1's name: ")
6. p2 = input("Input player 2's name: ")
8. p1\_pos = 0
9. p2\_pos = 0
11. round\_count = 0
13. **while** True:
14. round\_count += 1
15. **print**("\n [ Round", round\_count, "]")
17. p1\_throw = randint(1, 6)
18. p1\_pos += p1\_throw
19. **print**(p1, "got", p1\_throw, ", current position is", p1\_pos)
21. p2\_throw = randint(1, 6)
22. p2\_pos += p2\_throw
23. **print**(p2, "got", p2\_throw, ", current position is", p2\_pos)
25. **if** p1\_pos >= 30 **and** p2\_pos >= 30:
26. **print**('\n', p1, "and", p2, "tied in", round\_count, "rounds.")
27. **break**
28. **elif** p1\_pos >= 30 **and** p2\_pos < 30:
29. **print**('\n', p1, "won in", round\_count, "rounds!")
30. **break**
31. **elif** p2\_pos >= 30 **and** p1\_pos < 30:
32. **print**('\n', p2, "won in", round\_count, "rounds!")
33. **break**

## 결과 화면



## 설명

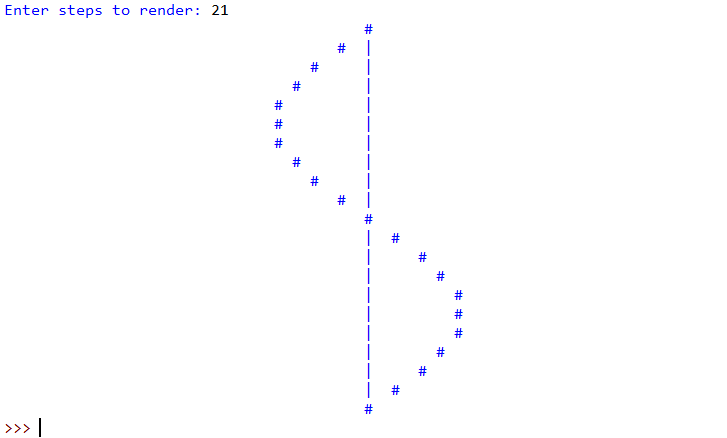
플레이어의 위치를 기록하는 **p1\_pos, p2\_pos** 변수에 **randint**에서 나온 수를 더하면서 라운드를 진행하고, 라운드마다 30에 도달한 플레이어가 있는지 확인하는 알고리즘이다.

# Problem 7. 사인 그래프 그리기

## 소스 코드

1. # problem 7: printing a sin graph vertically
3. **from** math **import** sin, pi
5. limit = int(input("Enter steps to render: "))
7. graph\_character = '#'
8. window\_width = 80
9. max\_height = 10 # defines the height of 1
10. step = 10 # defines steps in single radian pi
12. **def** print\_blank(repeat):
13. s = ""
15. # creates a string that contain repeat amount of spaces
16. **for** i **in** range(round(repeat)):
17. s += " "
19. **print**(s, end='')
21. **for** i **in** range(limit):
22. value = sin(i / step \* pi)
24. **if** int(value\*max\_height) > 0:
25. print\_blank(window\_width/2 - value\*max\_height)
26. **print**(graph\_character, end='')
28. print\_blank(value\*max\_height - 1)
29. **print**("|")
31. **elif** int(value\*max\_height) < 0:
32. print\_blank(window\_width/2)
33. **print**("|", end='')
35. print\_blank(-value\*max\_height - 1)
36. **print**(graph\_character)
38. **else**:
39. print\_blank(window\_width/2)
40. **print**(graph\_character)

## 결과 화면



## 설명

사용자가 그래프를 어디까지 그릴 것인지 입력한 값과 변수들에 따라서 사인 그래프를 그리는 알고리즘이다.

**graph\_character** 는 그래프를 그리는 문자를 담고 있어 다른 기호를 사용할 때 변경할 수 있다.

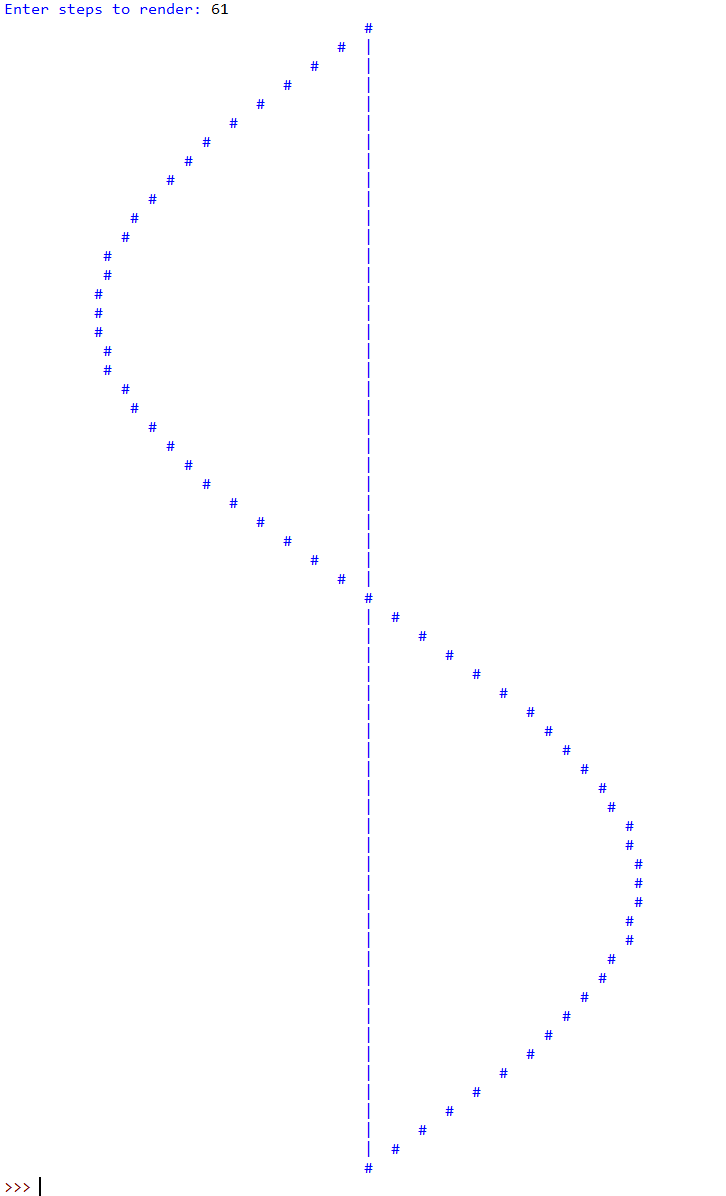
**window\_width**는 사용자의 창 너비를 나타내 그래프가 화면의 중앙에 그려지도록 하는데 사용된다. (80은 IDLE Shell의 기본 너비임으로 초기값으로 80을 사용함)

**max\_height**는 사인 그래프에서 최대 높이인 1이 몇 글자로 그려질지를 정한다.

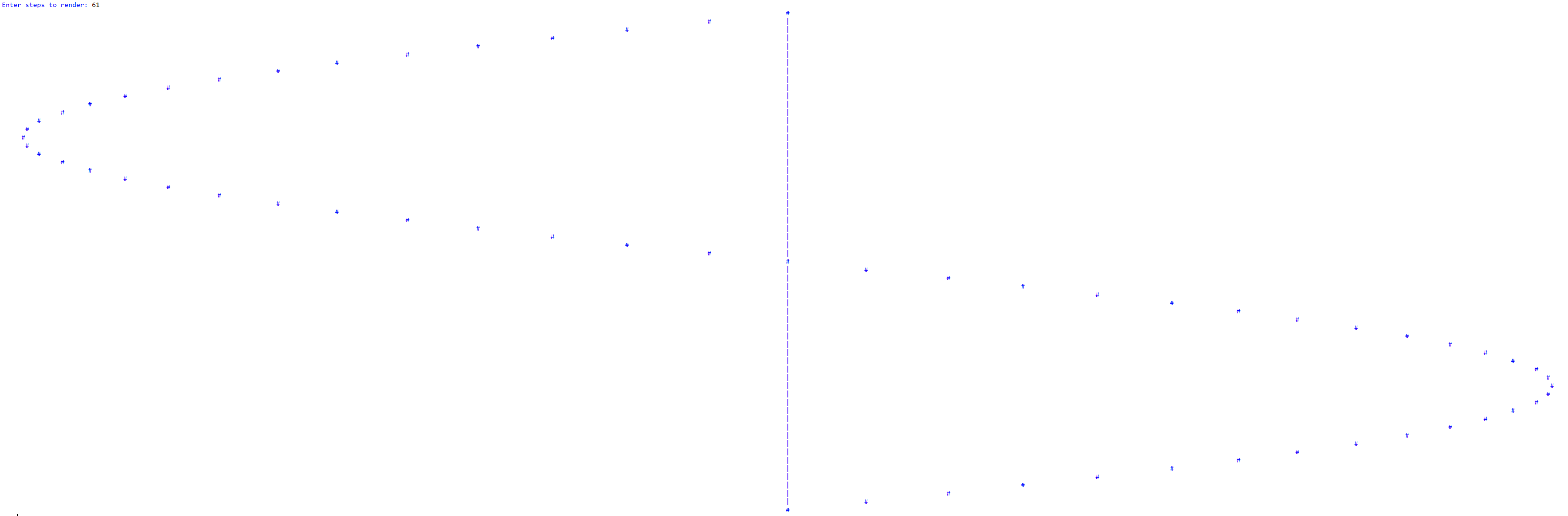
**step**은 1 pi가 몇 단계로 그려질지를 정한다.

이 변수들을 조정해 그래프를 더 자세히 그리거나 더 많은 값까지의 그래프를 그릴 수 있다.

그래프의 X축을 그리기 위해서 그래프가 Y방향으로 양수, 음수일 때를 구분해서 제작했다. 양수일 때는 그래프가 X축 밑에 있기 때문에 그래프를 그리고 축을 그리는 방법을, 음수일 때는 반대로 축을 그리고 그래프를 그리는 방법을 사용했다.



더 자세하게 그래프를 그린 예시 (max\_height = 30, step = 30)



다른 예시 (max\_height = 195, step = 30)