과제 1

ASSIGNMENT 1

391040277 | Nathan Cho | 조나단

Computational Thinking and Problem Solving | 2019-02-02

Problem 1. 절대값 출력하기

소스 코드

```
# problem 1: get a user input of a integer and print its absolute value

while True:
    i = int(input("Enter a integer: "))
    if i > 0:
        print("=", i)
    elif i < 0:
        print("=", -i)
    else:
        break
```

결과 화면

```
Enter a integer: 98
= 98
Enter a integer: -6
= 6
Enter a integer: 0
>>>
```

설명

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

Problem 2. 리스트에서 데이터 찾기

소스 코드

```
# problem 2: finding and inserting a data to list

town = ['흑석동', '사당동', '상도동', '노량진동', '규동']

print("Press enter to quit")

while True:
    i = input("Enter town name: ")
    if i is "":
        break
    else:
        if i in town:
            print("The town index is", town.index(i) + 1)
        else:
            town.append(i)
            print("Adding the town in index", len(town))
```

결과 화면

```
Press enter to quit
Enter town name: 사당동
The town index is 2
Enter town name: 가츠동
Adding the town in index 6
Enter town name: 가츠동
The town index is 6
Enter town name: 우동
Adding the town in index 7
Enter town name: 우동
The town index is 7
Enter town name:
>>>
```

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

```
# problem 3: menu - price robot

menu = {'noodle':500, 'ham':200, 'egg':100, 'spaghetti':900}

print("Press enter to quit")

while True:
    print("\nHello, please choose a menu from below.")
    selection = input("[noodle, ham, egg, spaghetti] : ")

    if selection == "":
        break

    try:
        print("It is", menu[selection], "won.")
    except:
        print("There is no such menu.")
```

결과 화면

```
Press enter to quit

Hello, please choose a menu from below.
[noodle, ham, egg, spaghetti] : ham
It is 200 won.

Hello, please choose a menu from below.
[noodle, ham, egg, spaghetti] : spaghetti
It is 900 won.

Hello, please choose a menu from below.
[noodle, ham, egg, spaghetti] : bread
There is no such menu.

Hello, please choose a menu from below.
[noodle, ham, egg, spaghetti] :
>>>
```

설명

dictionary 를 사용해서 음식 이름 (key) – 가격 (value)의 모델을 만들어 사용했다. 입력한 값을 dictionary 에서 key 로 사용해 가격을 가져오고 except 구문을 통해 key 가 존재하지 않을 때 생기는 예외를 처리했다.

Problem 4. 오름차순 출력

소스 코드

```
# problem 4: ascending sort of user input

data = []

print("Input 0 to end input\n")

while True:
    i = int(input("Input data: "))

    if i == 0:
        # end of input
        break
    else:
        # append data to the list
        data.append(i)

# sort the list to ascending order
data.sort()

print("Result:", data, "(" + str(len(data)) + " items)")
```

결과 화면

```
Input 0 to end input

Input data: 90
Input data: 55
Input data: 86
Input data: 79
Input data: 91
Input data: 0
Result: [55, 79, 86, 90, 91] (5 items)
>>>
```

설명

사용자의 입력으로 만들어진 리스트를 list 자료형의 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

```
# problem 5: print the multiplting table according to user input

# corrects the user input value
while True:
    try:
        start = int(input("Please input starting column number: "))
        end = int(input("Please input ending column number: "))

        # check for invalid input
        if start > 9 or start < 1 or end > 9 or end < 1:
            print("Invalid input - Both column numbers should be between 1 and 9")
            continue

        if end - start > 4:
            print("The difference should be same or smaller than 4\n")
            continue
        elif end - start <= 0:
            print("Ending column should be bigger than starting column\n")
            continue
        else:
            break
    except:
        continue

def print_blank(repeat):
    s = ""

    # creates a string that contain repeat amount of spaces
    for i in range(int(repeat)):
        s += " "

    print(s, end='')

window_width = 80 # seems it's default for IDLE on startup
column_width = 10
column_count = end - start + 1
column_margin = 4
# use below to use all same margin
# column_margin = (window_width - column_width*column_count) / (column_count + 1)

print()
print_blank(window_width/2 - 4)
print("From", start, "to", end)
print()

for i in range(1, 10):
    # print the blanks to start the first column
    # remove below to use all same margin
    print_blank((window_width - column_width*column_count - column_margin*column_count)/2)

    for j in range(start, end + 1):
        expression = str(j) + " X " + str(i) + " = "
        answer = i*j

        if int(answer / 10) == 0:
            # the number is 1 digit, needs an extra space
            expression += " "

        # print the expression and answer
        print_blank(column_margin)
        print(expression + str(answer), end='')

    # line break
    print()
```

결과 화면

```
Please input starting column number: 2
Please input ending column number: 3

                                From 2 to 3

                            2 X 1 =  2    3 X 1 =  3
                            2 X 2 =  4    3 X 2 =  6
                            2 X 3 =  6    3 X 3 =  9
                            2 X 4 =  8    3 X 4 = 12
                            2 X 5 = 10    3 X 5 = 15
                            2 X 6 = 12    3 X 6 = 18
                            2 X 7 = 14    3 X 7 = 21
                            2 X 8 = 16    3 X 8 = 24
                            2 X 9 = 18    3 X 9 = 27
>>>

Please input starting column number: 4
Please input ending column number: 6

                                From 4 to 6

                        4 X 1 =  4    5 X 1 =  5    6 X 1 =  6
                        4 X 2 =  8    5 X 2 = 10    6 X 2 = 12
                        4 X 3 = 12    5 X 3 = 15    6 X 3 = 18
                        4 X 4 = 16    5 X 4 = 20    6 X 4 = 24
                        4 X 5 = 20    5 X 5 = 25    6 X 5 = 30
                        4 X 6 = 24    5 X 6 = 30    6 X 6 = 36
                        4 X 7 = 28    5 X 7 = 35    6 X 7 = 42
                        4 X 8 = 32    5 X 8 = 40    6 X 8 = 48
                        4 X 9 = 36    5 X 9 = 45    6 X 9 = 54
>>>

Please input starting column number: 6
Please input ending column number: 9

                                From 6 to 9

                    6 X 1 =  6    7 X 1 =  7    8 X 1 =  8    9 X 1 =  9
                    6 X 2 = 12    7 X 2 = 14    8 X 2 = 16    9 X 2 = 18
                    6 X 3 = 18    7 X 3 = 21    8 X 3 = 24    9 X 3 = 27
                    6 X 4 = 24    7 X 4 = 28    8 X 4 = 32    9 X 4 = 36
                    6 X 5 = 30    7 X 5 = 35    8 X 5 = 40    9 X 5 = 45
                    6 X 6 = 36    7 X 6 = 42    8 X 6 = 48    9 X 6 = 54
                    6 X 7 = 42    7 X 7 = 49    8 X 7 = 56    9 X 7 = 63
                    6 X 8 = 48    7 X 8 = 56    8 X 8 = 64    9 X 8 = 72
                    6 X 9 = 54    7 X 9 = 63    8 X 9 = 72    9 X 9 = 81
>>>
```

```
Please input starting column number: 4
Please input ending column number: 9
The difference should be same or smaller than 4

Please input starting column number:
```

4 단 이상을 출력할 수 없다는 오류 메시지

```
Please input starting column number: 5
Please input ending column number: 2
Ending column should be bigger than starting column
```

시작이 끝보다 클 수 없다는 오류 메시지

```
Please input starting column number: -1
Please input ending column number: 10
Invalid input - Both column numbers should be between 1 and 9
```

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.

```
def print_blank(repeat):
    s = ""

    # creates a string that contain repeat amount of spaces
    for i in range(int(repeat)):
        s += " "

    print(s, end='')
```

print_blank 라는 함수를 제작해 많은 양의 공백을 간단하게 처리했다.

```
        if int(answer / 10) == 0:
            # the number is 1 digit, needs an extra space
            expression += " "
```

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.

```
# use below to use all same margin
# column_margin = (window_width - column_width*column_count) / (column_count + 1)

    # print the blanks to start the first column
    # remove below to use all same margin
```

위 두 주석은 **column_margin** 변수의 단 사이에 기본 4 칸 간격을 사용하지 않고 창 크기에 맞추어서 일정한 간격을 가질 수 있도록 한다.

```
Please input starting column number: 3
Please input ending column number: 5

                                From 3 to 5

            3 X 1 =  3          4 X 1 =  4          5 X 1 =  5
            3 X 2 =  6          4 X 2 =  8          5 X 2 = 10
            3 X 3 =  9          4 X 3 = 12          5 X 3 = 15
            3 X 4 = 12          4 X 4 = 16          5 X 4 = 20
            3 X 5 = 15          4 X 5 = 20          5 X 5 = 25
            3 X 6 = 18          4 X 6 = 24          5 X 6 = 30
            3 X 7 = 21          4 X 7 = 28          5 X 7 = 35
            3 X 8 = 24          4 X 8 = 32          5 X 8 = 40
            3 X 9 = 27          4 X 9 = 36          5 X 9 = 45
>>>
```

일정한 간격을 가지게 출력한 경우

Problem 6. 경주 게임

소스 코드

```
# problem 6: dice based racing game

from random import randint

p1 = input("Input player 1's name: ")
p2 = input("Input player 2's name: ")

p1_pos = 0
p2_pos = 0

round_count = 0

while True:
    round_count += 1
    print("\n [ Round", round_count, "]")

    p1_throw = randint(1, 6)
    p1_pos += p1_throw
    print(p1, "got", p1_throw, ", current position is", p1_pos)

    p2_throw = randint(1, 6)
    p2_pos += p2_throw
    print(p2, "got", p2_throw, ", current position is", p2_pos)

    if p1_pos >= 30 and p2_pos >= 30:
        print('\n', p1, "and", p2, "duced in", round_count, "rounds.")
        break
    elif p1_pos >= 30 and p2_pos < 30:
        print('\n', p1, "won in", round_count, "rounds!")
        break
    elif p2_pos >= 30 and p1_pos < 30:
        print('\n', p2, "won in", round_count, "rounds!")
        break
```

결과 화면

```
Input player 1's name: Lorem
Input player 2's name: Ipsum

 [ Round 1 ]
Lorem got 2 , current position is 2
Ipsum got 6 , current position is 6

 [ Round 2 ]
Lorem got 2 , current position is 4
Ipsum got 4 , current position is 10

 [ Round 3 ]
Lorem got 3 , current position is 7
Ipsum got 2 , current position is 12

 [ Round 4 ]
Lorem got 6 , current position is 13
Ipsum got 5 , current position is 17

 [ Round 5 ]
Lorem got 3 , current position is 16
Ipsum got 1 , current position is 18

 [ Round 6 ]
Lorem got 5 , current position is 21
Ipsum got 1 , current position is 19

 [ Round 7 ]
Lorem got 5 , current position is 26
Ipsum got 1 , current position is 20

 [ Round 8 ]
Lorem got 5 , current position is 31
Ipsum got 5 , current position is 25

 Lorem won in 8 rounds!
>>>
```

설명

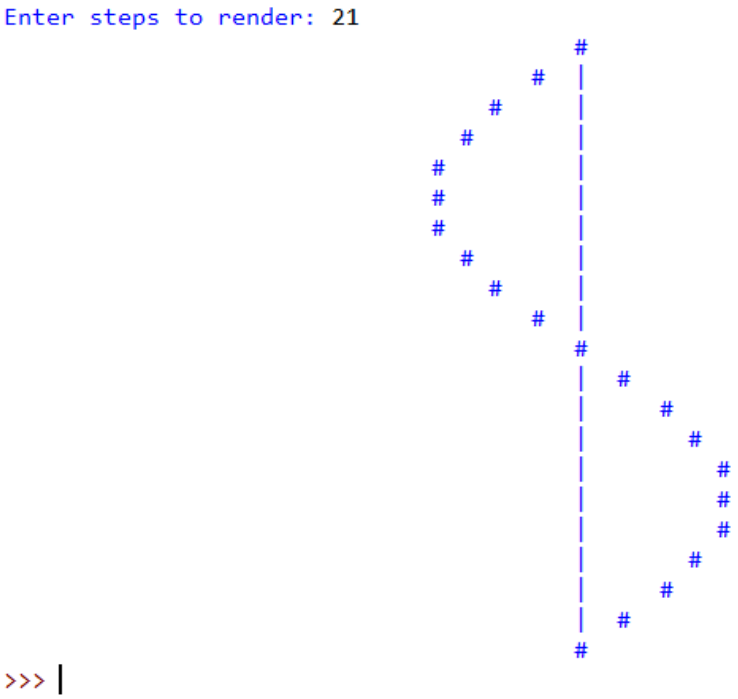
플레이어의 위치를 기록하는 **p1_pos, p2_pos** 변수에 randint 에서 나온 수를 더하면서 라운드를 진행하고, 라운드마다 30 에 도달한 플레이어가 있는지 확인하는 알고리즘이다.

Problem 7. 사인 그래프 그리기

소스 코드

```
# problem 7: printing a sin graph vertically

from math import sin, pi

limit = int(input("Enter steps to render: "))

graph_character = '#'
window_width = 80
max_height = 10 # defines the height of 1
step = 10 # defines steps in single radian pi

def print_blank(repeat):
    s = ""

    # creates a string that contain repeat amount of spaces
    for i in range(round(repeat)):
        s += " "

    print(s, end='')

for i in range(limit):
    value = sin(i / step * pi)

    if int(value*max_height) > 0:
        print_blank(window_width/2 - value*max_height)
        print(graph_character, end='')

        print_blank(value*max_height - 1)
        print("|")

    elif int(value*max_height) < 0:
        print_blank(window_width/2)
        print("|", end='')

        print_blank(-value*max_height - 1)
        print(graph_character)

    else:
        print_blank(window_width/2)
        print(graph_character)
```

결과 화면



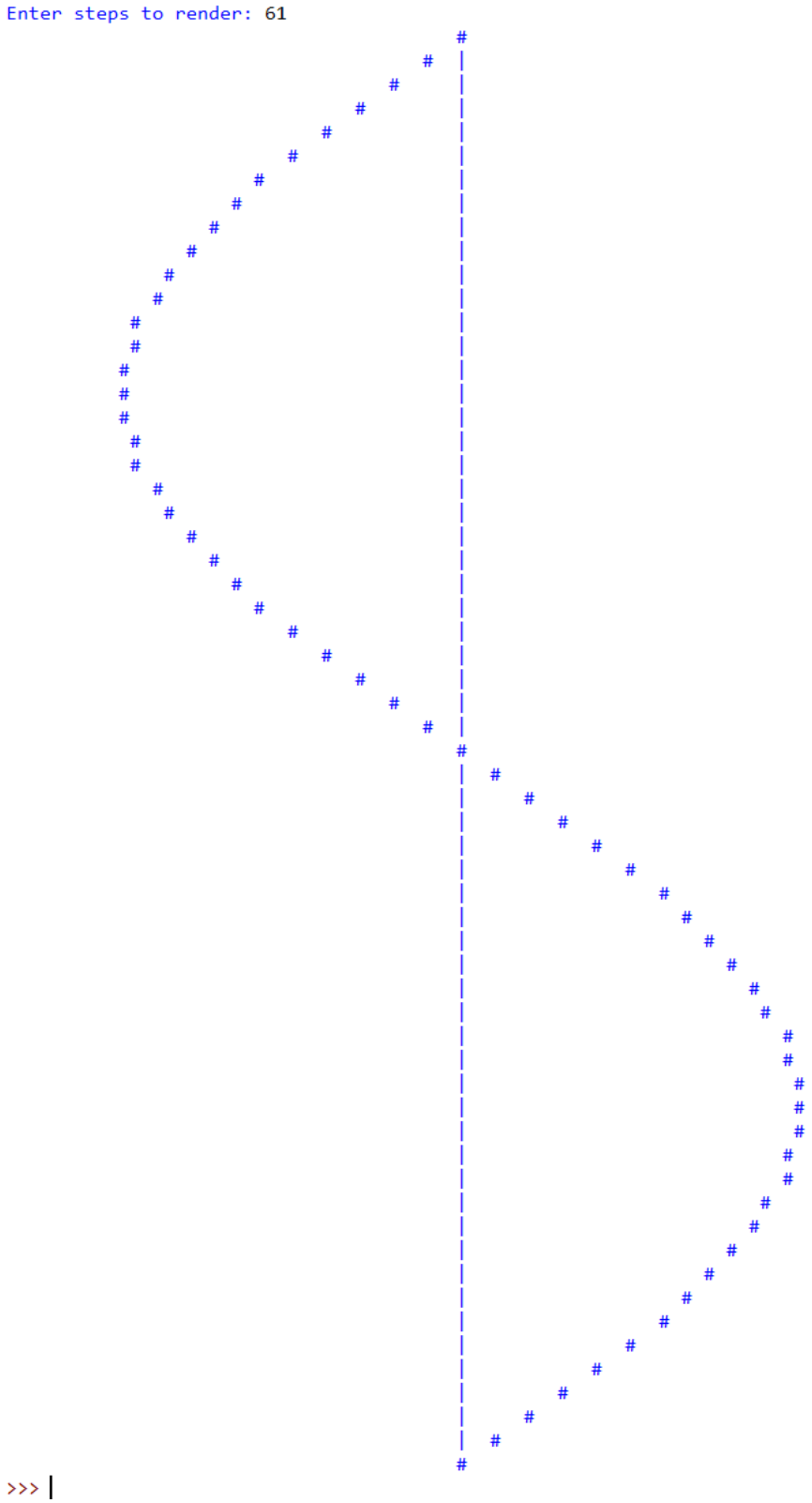
설명

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

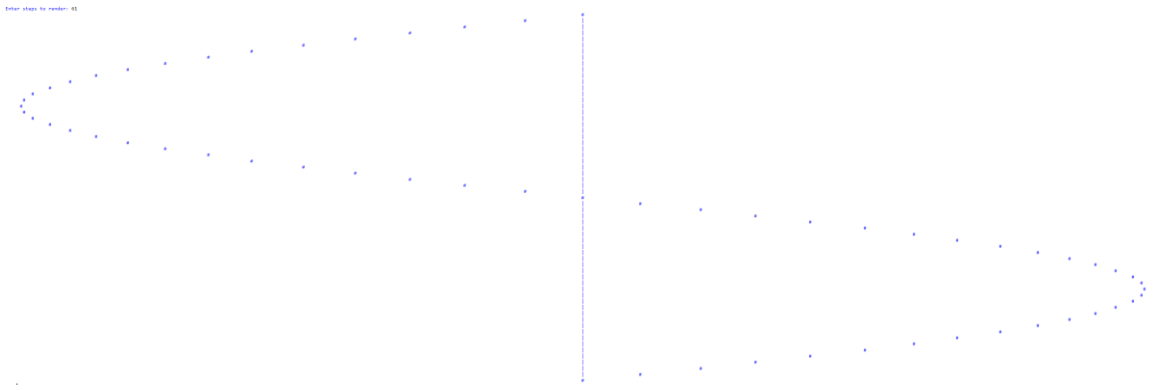
graph_character 는 그래프를 그리는 문자를 담고 있어 다른 기호를 사용할 때 변경할 수 있다.
window_width 는 사용자의 창 너비를 나타내 그래프가 화면의 중앙에 그려지도록 하는데 사용된다. (80 은 IDLE Shell 의 기본 너비임으로 초기값으로 80 을 사용함)
max_height 는 사인 그래프에서 최대 높이인 1 이 몇 글자로 그려질지를 정한다.
step 은 1 pi 가 몇 단계로 그려질지를 정한다.

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

그래프의 X 축을 그리기 위해서 그래프가 Y 방향으로 양수, 음수일 때를 구분해서 제작했다. 양수일 때는 그래프가 X 축 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)