Lab: Basic Syntax, Conditional Statements, and Loops

Please, submit your source code solutions for the described problems to the Judge System.

1. Number Definer

Write a program that reads a floating-point number and:

- prints **"zero"** if the number is zero
- prints **"positive"** or **"negative"**
- adds **"small"** if the absolute value of the number is less than 1 and it is not 0, or **"large"** if it exceeds 1 000 000

Example

Input	Output
25	positive
0.7	small positive
435247392.921	large positive
-0.005	small negative
-103.21	negative
-358583355123.001	large negative

Hints

First, we read the number from the console as a float because we are going to receive floating-point numbers:

```
1    number = float(input())
```

Then, we write a condition to check if the number is zero:

```
2    if number == 0:
3        print("zero")
```

After that, we write a condition to check if the number is positive and add the additional conditions:

```
4    elif number > 0:
5        if number < 1:
6            print("small positive")
7        elif number > 1000000:
8            print("large positive")
9        else:
10           print("positive")
```

Then, we check if the number is negative. To check if the number is small or large, we use the absolute value of the negative number:

```
else:
    if abs(number) < 1:
        print("small negative")
    elif abs(number) > 1000000:
        print("large negative")
    else:
        print("negative")
```

2. Largest of Three Numbers

Write a program that receives **three whole numbers** and prints the **largest one.**

Example

Input	Output
3 -1 5	5
0 -1 -2	0

Hints

We start by reading the three numbers from the console:

biggest-of-three-numbers.py

```
first_num = int(input())
second_num = int(input())
third_num = int(input())

```

Then we compare them and print the largest one:

```
if first_num > second_num and first_num > third_num:
    print(first_num)
elif second_num > first_num and second_num > third_num:
    print(second_num)
else:
    print(third_num)

```

3. Word Reverse

Write a program that receives a **single word**, **reverses it,** and **prints it**.

Example

Input	Output

Python	nohtyP
banana	ananab

Hints

We read the word from the console:

```
word = input()
```

Next, we create a new variable to store the reversed word:

```
reversed_word = ""
```

After that, we create a for loop which would iterate backward through the word and add each character to the new word:

```
for i in range(len(word) - 1, -1, -1):
    reversed_word += word[i]
print(reversed_word)
```

- The starting point value of the loop should be the length of the word minus one position.
- We should loop until we reach the **0** index, so the stop value of the loop should be **-1**.
- Finally, the step is backward, so the step size is **-1**.

4. Even Numbers

Write a program that receives a number **n** on the first line. On the **following n lines**, it receives **different integer numbers**. If the program receives an odd number, print **"{num} is odd!"** and **end the program**. Otherwise, if all numbers given are even, print **"All numbers are even."**.

Example

Input	Output
2 12 286	All numbers are even.
5 2 9	9 is odd!

Hints

We start by reading an integer number:

```
n = int(input())
```

Then, we create a for loop that receives a number and checks if the given number is odd. If the condition is met, the program should print the number and then should break the loop and end:

```
for i in range(n):
    number = int(input())
    if not number % 2 == 0:
        print(f"{number} is odd!")
        break
```

If the condition is not met at all, we enter the else-condition and print the result:

```
else:
    print("All numbers are even.")
```

5. Number Between 1 and 100

Write a program that reads different floating-point numbers from the console. When it receives a number between 1 and 100 inclusive, the program should stop reading and should print **"The number {number} is between 1 and 100"**.

Example

Input	Output
-3 0.9 44	The number 44.0 is between 1 and 100
0.5 90 -4 101	The number 90.0 is between 1 and 100

Hints

We start by reading a floating-point number:

```
number = float(input())
```

Then, we create a while loop that checks if the given number is less than 1 or greater than 100. If the condition is met, the program should enter the body of the loop:

```
while number < 1 or number > 100:
    number = float(input())
```

If the condition is not met, we exit the loop and print the result:

```
print(f'The number {number} is between 1 and 100')
```

6. Shopping

Write a program that reads an **integer number** representing a **budget**. On the following lines, it reads **integer numbers** representing the **prices** of each product you should buy until it receives the command **"End"**.

During the iterations, if there is **not enough budget** left to buy the next product, it prints **"You went in overdraft!"** and end the program.

Otherwise, if you accomplished to **buy all products** before receiving **"End"**, it prints **"You bought everything needed."**

Example

Input	Output
100 5 End	You bought everything needed.
50 25 20 10	You went in overdraft!

Hints

We start by reading an integer number and the first command:

```
budget = int(input())
command = input()
```

Then, we create a while loop that checks if the given command is not **"End"**. If the condition is met, the program should enter the body of the loop. It defines a variable for the product price and checks if the budget is not enough. If the condition is met, the program should print **"You went in overdraft!"** and then should break the loop and end. Otherwise, the loop should receive the next command and iterate over the code again:

```
while command != "End":
    product_price = int(command)
    budget -= product_price
    if budget < 0:
        print("You went in overdraft!")
        break
    command = input()
```

If the condition is not met before we receive the **"End"** command, we enter the else-condition and print the result:

```
else:
    print("You bought everything needed.")
```

7. Patterns

Write a program that receives a **number** and **creates the following pattern**. The number represents the largest count of stars on one row.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Example

Input	Output
3	* ** *** ** *
5	* ** *** **** ***** **** *** ** *

Hints

First, we read the number from the console:

```
number = int(input())
```

We create the first loop, which will print half of the pattern until **`i == number`**:

```
for i in range(1, number + 1):
    for j in range(0, i):
        print('*', end='')
    print()
```

We create the inner loop through the numbers from **`0`** to **`i`** (**`i`** is not inclusive). We use **`end=''`** to stay on the same line. We print the new line after we draw all the stars for the particular line.

Next, we create the second loop backward as we need to decrease the number of stars with each line. It should draw one line less than the previous one:

```
for i in range(number - 1, 0, -1):
    for j in range(0, i):
        print('*', end='')
    print()
```