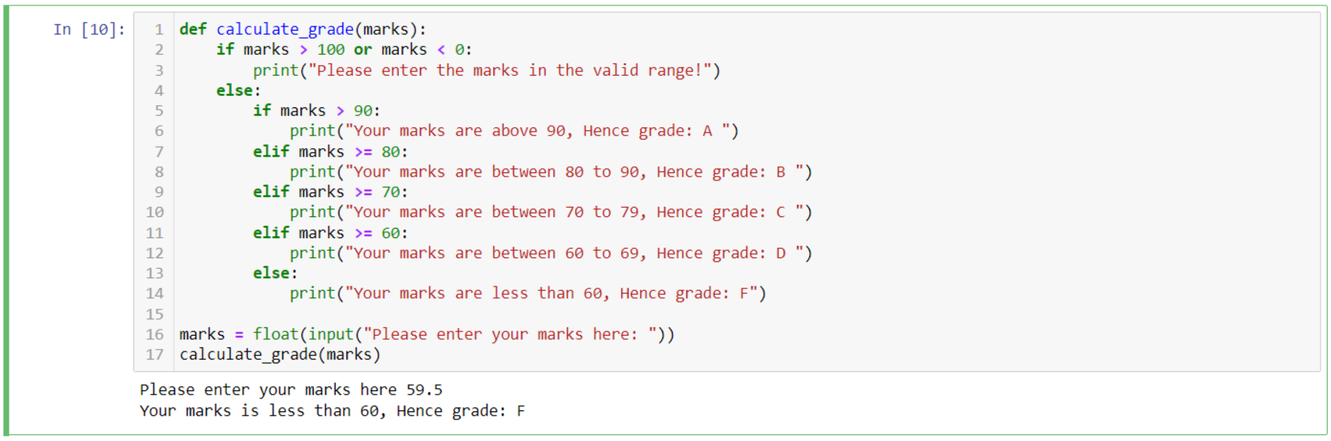
NAME: Neev Shah **CLASS:** AIML-C

USN: 22BTRCL107 **DATE:** 24th April 2024

ASSIGNMENT ON PYTHON

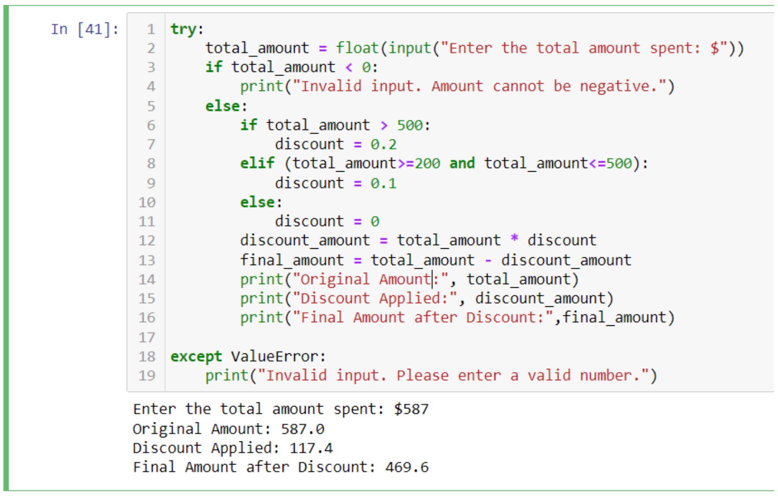
1. Create a Python script that takes a student's score (0-100) as input and prints their grade based on the following criteria:
Above 90: "Grade: A"
80 to 90: "Grade: B"
70 to 79: "Grade: C"
60 to 69: "Grade: D"
Below 60: "Grade: F"



```python
def calculate_grade(marks):
    if marks > 100 or marks < 0:
        print("Please enter the marks in the valid range!")
    else:
        if marks > 90:
            print("Your marks are above 90, Hence grade: A ")
        elif marks >= 80:
            print("Your marks are between 80 to 90, Hence grade: B ")
        elif marks >= 70:
            print("Your marks are between 70 to 79, Hence grade: C ")
        elif marks >= 60:
            print("Your marks are between 60 to 69, Hence grade: D ")
        else:
            print("Your marks are less than 60, Hence grade: F")

marks = float(input("Please enter your marks here: "))
calculate_grade(marks)
```

```
Please enter your marks here 59.5
Your marks is less than 60, Hence grade: F
```

2. Create a Python program that applies a discount to a purchase based on the amount spent. The program asks for the total amount and applies the following discount rates:
Spend over $500: 20% discount
Spend $200 - $500: 10% discount
Spend below $200: No discount
The program should print the original amount, the discount applied, and the final amount after the discount.



```python
try:
    total_amount = float(input("Enter the total amount spent: $"))
    if total_amount < 0:
        print("Invalid input. Amount cannot be negative.")
    else:
        if total_amount > 500:
            discount = 0.2
        elif (total_amount>=200 and total_amount<=500):
            discount = 0.1
        else:
            discount = 0
        discount_amount = total_amount * discount
        final_amount = total_amount - discount_amount
        print("Original Amount:", total_amount)
        print("Discount Applied:", discount_amount)
        print("Final Amount after Discount:",final_amount)

except ValueError:
    print("Invalid input. Please enter a valid number.")
```

```
Enter the total amount spent: $587
Original Amount: 587.0
Discount Applied: 117.4
Final Amount after Discount: 469.6
```

3. Create a program that asks for the user's birth month and day and then tells them their zodiac sign. For simplicity, you can use the following date ranges:

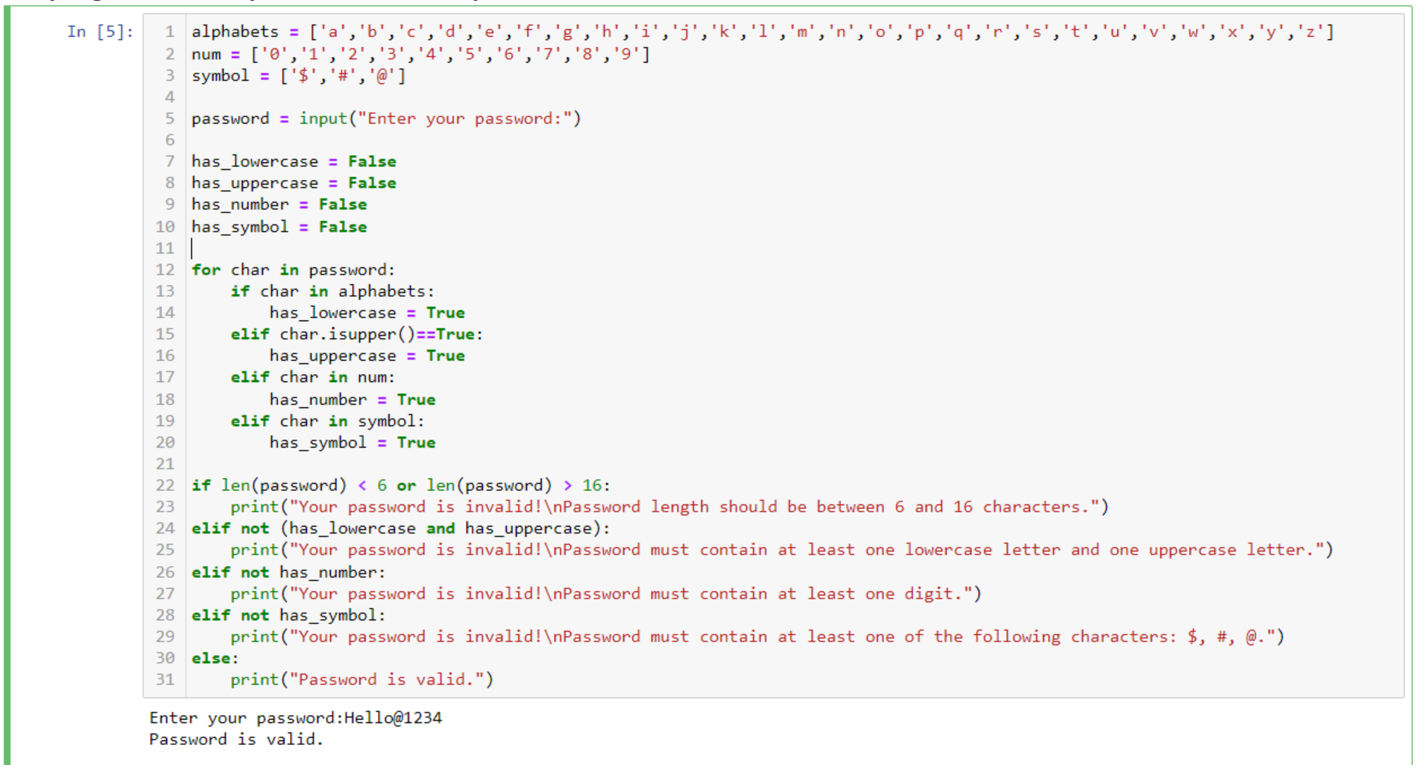
Aries: March 21 - April 19
Taurus: April 20 - May 20
Gemini: May 21 - June 20
Cancer: June 21 - July 22
Leo: July 23 - August 22
Virgo: August 23 - September 22
Libra: September 23 - October 22
Scorpio: October 23 - November 21
Sagittarius: November 22 - December 21
Capricorn: December 22 - January 19
Aquarius: January 20 - February 18
Pisces: February 19 - March 20
Make sure to handle invalid inputs gracefully.

In [45]:

```python
def calculate_zodiac_sign():
    try:
        month = input("Enter your birth month, e.g., January: ").lower()
        day = int(input("Enter your birth day: "))

        if not (1 <= day <= 31):
            print("Invalid day. Please enter valid values.")
        else:
            if month == "january":
                zodiac_sign = 'Capricorn' if (day < 20) else 'Aquarius'
            elif month == "february":
                zodiac_sign = 'Aquarius' if (day < 19) else 'Pisces'
            elif month == "march":
                zodiac_sign = 'Pisces' if (day < 21) else 'Aries'
            elif month == "april":
                zodiac_sign = 'Aries' if (day < 20) else 'Taurus'
            elif month == "may":
                zodiac_sign = 'Taurus' if (day < 21) else 'Gemini'
            elif month == "june":
                zodiac_sign = 'Gemini' if (day < 21) else 'Cancer'
            elif month == "july":
                zodiac_sign = 'Cancer' if (day < 23) else 'Leo'
            elif month == "august":
                zodiac_sign = 'Leo' if (day < 23) else 'Virgo'
            elif month == "september":
                zodiac_sign = 'Virgo' if (day < 23) else 'Libra'
            elif month == "october":
                zodiac_sign = 'Libra' if (day < 23) else 'Scorpio'
            elif month == "november":
                zodiac_sign = 'Scorpio' if (day < 22) else 'Sagittarius'
            elif month == "december":
                zodiac_sign = 'Sagittarius' if (day < 22) else 'Capricorn'
            else:
                zodiac_sign = None
            print("Your zodiac sign is:", zodiac_sign)
    except ValueError:
        print("Invalid input. Please enter numeric values for the day.")

calculate_zodiac_sign()
```

```
Enter your birth month, e.g., January: september
Enter your birth day: 4
Your zodiac sign is: Virgo
```

4. Write a Python program to check the validity of a password entered by the user. The password is considered valid if it meets the following criteria:
At least 1 letter between [a-z] and 1 letter between [A-Z].
At least 1 number between [0-9].
At least 1 character from [$#@].
Minimum length of 6 characters.
Maximum length of 16 characters.
The program should print whether the password is valid or not based on these criteria.

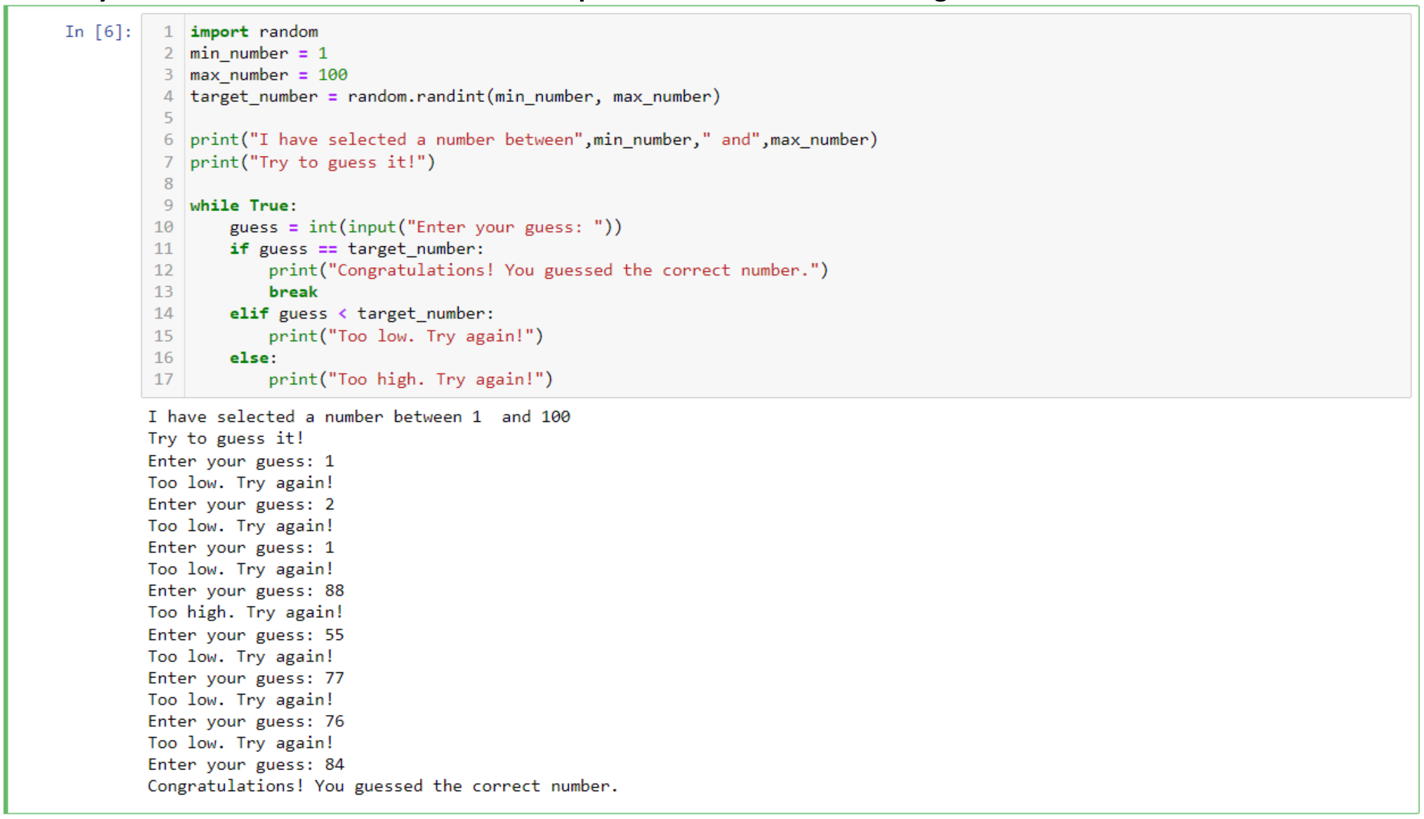


In [5]:

```python
alphabets = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
num = ['0','1','2','3','4','5','6','7','8','9']
symbol = ['$','#','@']

password = input("Enter your password:")

has_lowercase = False
has_uppercase = False
has_number = False
has_symbol = False

for char in password:
    if char in alphabets:
        has_lowercase = True
    elif char.isupper()==True:
        has_uppercase = True
    elif char in num:
        has_number = True
    elif char in symbol:
        has_symbol = True

if len(password) < 6 or len(password) > 16:
    print("Your password is invalid!\nPassword length should be between 6 and 16 characters.")
elif not (has_lowercase and has_uppercase):
    print("Your password is invalid!\nPassword must contain at least one lowercase letter and one uppercase letter.")
elif not has_number:
    print("Your password is invalid!\nPassword must contain at least one digit.")
elif not has_symbol:
    print("Your password is invalid!\nPassword must contain at least one of the following characters: $, #, @.")
else:
    print("Password is valid.")
```

```
Enter your password:Hello@1234
Password is valid.
```

5. Implement a simple number guessing game. First, set a target number within a certain range (e.g., 1 to 100). Then, using a while loop, ask the user to guess the number. Provide feedback for each guess ("too high" or "too low"). The game ends when the user guesses the number correctly. Use a break statement to exit the loop once the correct number is guessed.

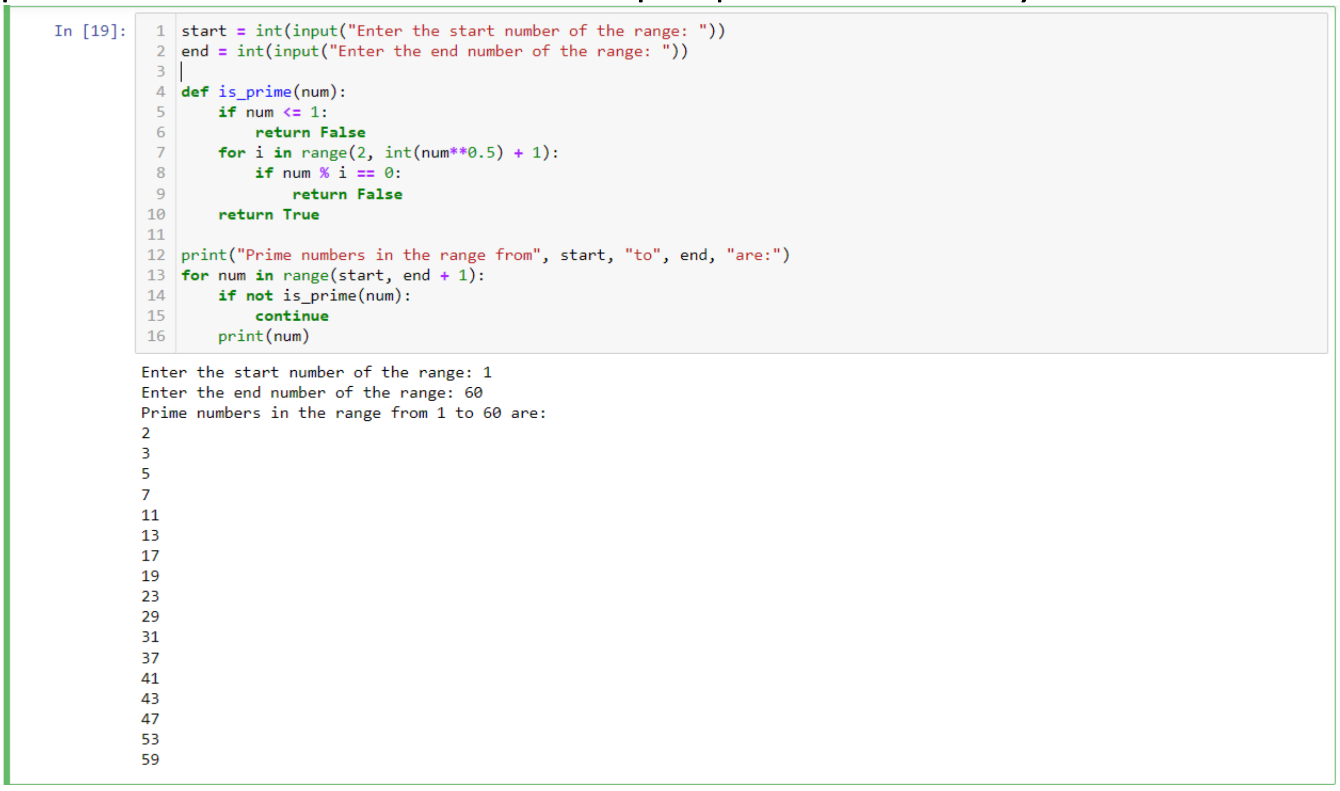


In [6]:

```python
import random
min_number = 1
max_number = 100
target_number = random.randint(min_number, max_number)

print("I have selected a number between",min_number," and",max_number)
print("Try to guess it!")

while True:
    guess = int(input("Enter your guess: "))
    if guess == target_number:
        print("Congratulations! You guessed the correct number.")
        break
    elif guess < target_number:
        print("Too low. Try again!")
    else:
        print("Too high. Try again!")
```

```
I have selected a number between 1  and 100
Try to guess it!
Enter your guess: 1
Too low. Try again!
Enter your guess: 2
Too low. Try again!
Enter your guess: 1
Too low. Try again!
Enter your guess: 88
Too high. Try again!
Enter your guess: 55
Too low. Try again!
Enter your guess: 77
Too low. Try again!
Enter your guess: 76
Too low. Try again!
Enter your guess: 84
Congratulations! You guessed the correct number.
```

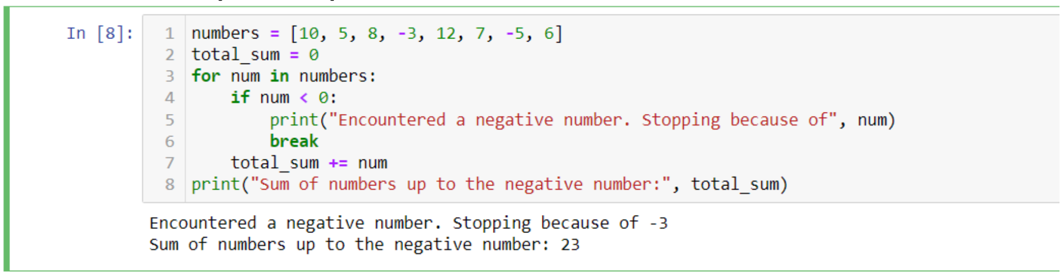
6. Write a Python program that asks the user to enter a range (start and end numbers). Use a for loop to iterate through this range, and for each number, check if it is a prime number. If it is, print the number. Use the continue statement to skip non-prime numbers efficiently.



```
In [19]:
start = int(input("Enter the start number of the range: "))
end = int(input("Enter the end number of the range: "))

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

print("Prime numbers in the range from", start, "to", end, "are:")
for num in range(start, end + 1):
    if not is_prime(num):
        continue
    print(num)
```

```
Enter the start number of the range: 1
Enter the end number of the range: 60
Prime numbers in the range from 1 to 60 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
```

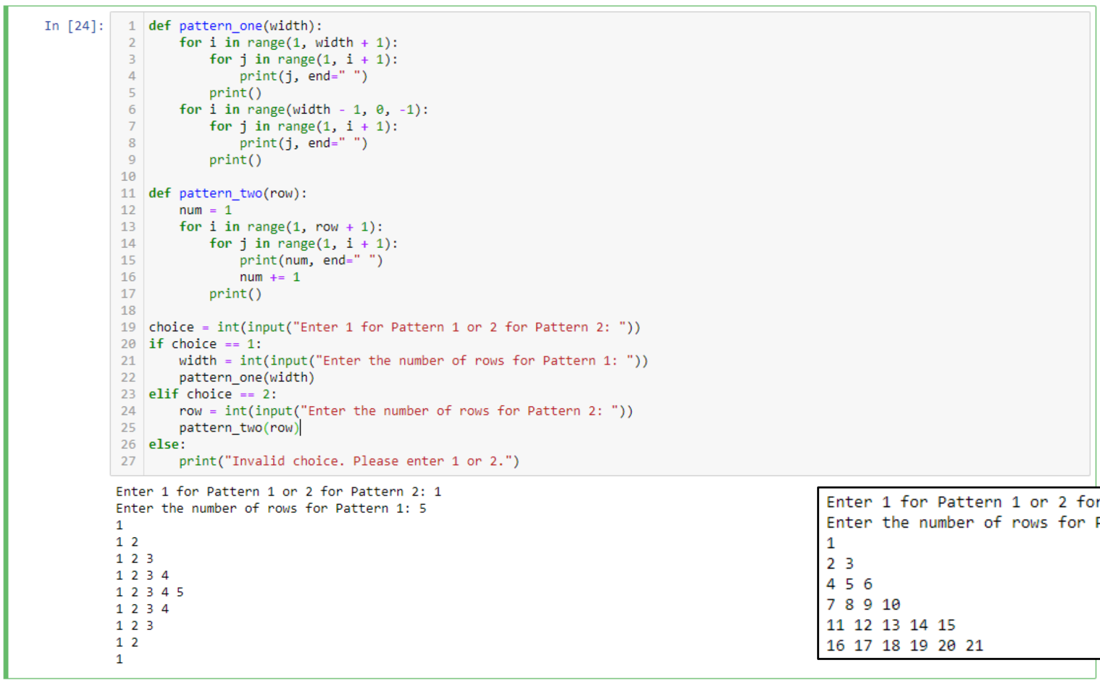
7. Create a Python program that iterates through a list of numbers (you can define the list in the code) and calculates the sum of the numbers. However, if the program encounters a number that is negative, it should stop adding any further numbers (i.e., break out of the loop) and print the current sum up to that point.



```
In [8]:
numbers = [10, 5, 8, -3, 12, 7, -5, 6]
total_sum = 0
for num in numbers:
    if num < 0:
        print("Encountered a negative number. Stopping because of", num)
        break
    total_sum += num
print("Sum of numbers up to the negative number:", total_sum)
```

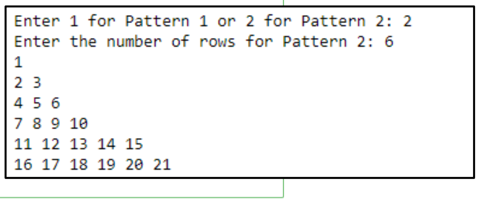
```
Encountered a negative number. Stopping because of -3
Sum of numbers up to the negative number: 23
```

8. Write a Python program to print the following patterns



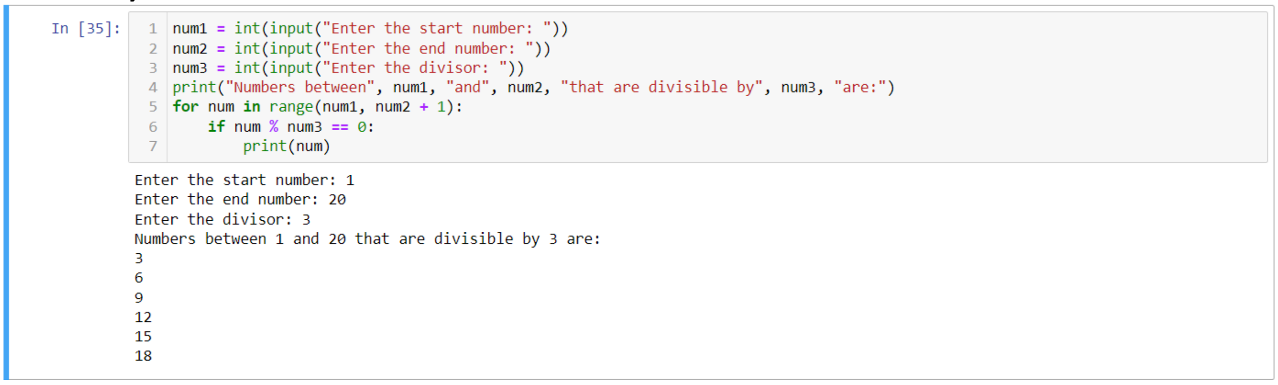
```
In [24]:
def pattern_one(width):
    for i in range(1, width + 1):
        for j in range(1, i + 1):
            print(j, end=" ")
        print()
    for i in range(width - 1, 0, -1):
        for j in range(1, i + 1):
            print(j, end=" ")
        print()

def pattern_two(row):
    num = 1
    for i in range(1, row + 1):
        for j in range(1, i + 1):
            print(num, end=" ")
            num += 1
        print()

choice = int(input("Enter 1 for Pattern 1 or 2 for Pattern 2: "))
if choice == 1:
    width = int(input("Enter the number of rows for Pattern 1: "))
    pattern_one(width)
elif choice == 2:
    row = int(input("Enter the number of rows for Pattern 2: "))
    pattern_two(row)
else:
    print("Invalid choice. Please enter 1 or 2.")
```

```
Enter 1 for Pattern 1 or 2 for Pattern 2: 1
Enter the number of rows for Pattern 1: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```



```
Enter 1 for Pattern 1 or 2 for Pattern 2: 2
Enter the number of rows for Pattern 2: 6
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

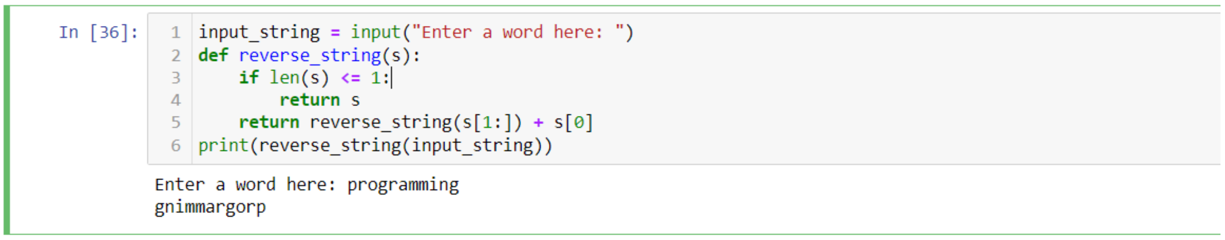
9. Create a program that asks for two numbers and prints all the numbers between them that are divisible by a third number asked from the user.



```
In [35]:
num1 = int(input("Enter the start number: "))
num2 = int(input("Enter the end number: "))
num3 = int(input("Enter the divisor: "))
print("Numbers between", num1, "and", num2, "that are divisible by", num3, "are:")
for num in range(num1, num2 + 1):
    if num % num3 == 0:
        print(num)
```

```
Enter the start number: 1
Enter the end number: 20
Enter the divisor: 3
Numbers between 1 and 20 that are divisible by 3 are:
3
6
9
12
15
18
```

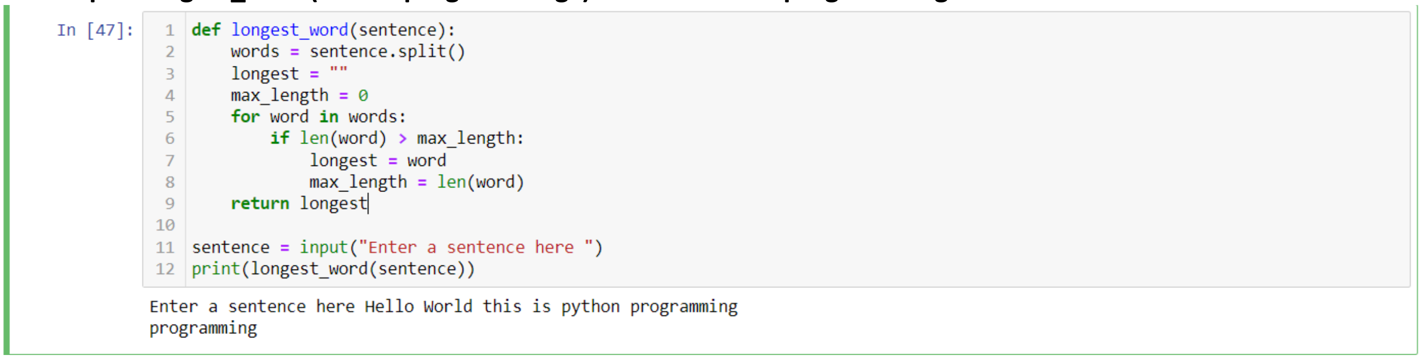
10. Write a recursive function named reverse_string that takes a string as input and returns its reverse. The function must use recursion to accomplish this task and should not use any loops or slicing ([::-1]).
Example Usage:
print(reverse_string("hello"))
Expected Output:
"olleh"



```
In [36]:
input_string = input("Enter a word here: ")
def reverse_string(s):
    if len(s) <= 1:
        return s
    return reverse_string(s[1:]) + s[0]
print(reverse_string(input_string))
```

```
Enter a word here: programming
gnimmargorp
```

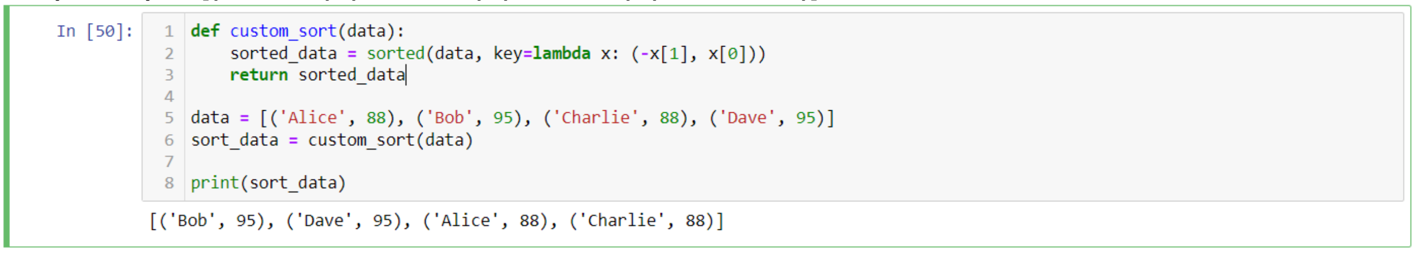
11. Create a function longest_word(sentence) that finds and returns the longest word in the given string sentence. If there are multiple words of the same length, return the first one encountered.
# Example: longest_word("I love programming") should return "programming"



```
In [47]:
def longest_word(sentence):
    words = sentence.split()
    longest = ""
    max_length = 0
    for word in words:
        if len(word) > max_length:
            longest = word
            max_length = len(word)
    return longest

sentence = input("Enter a sentence here ")
print(longest_word(sentence))
```

```
Enter a sentence here Hello World this is python programming
programming
```

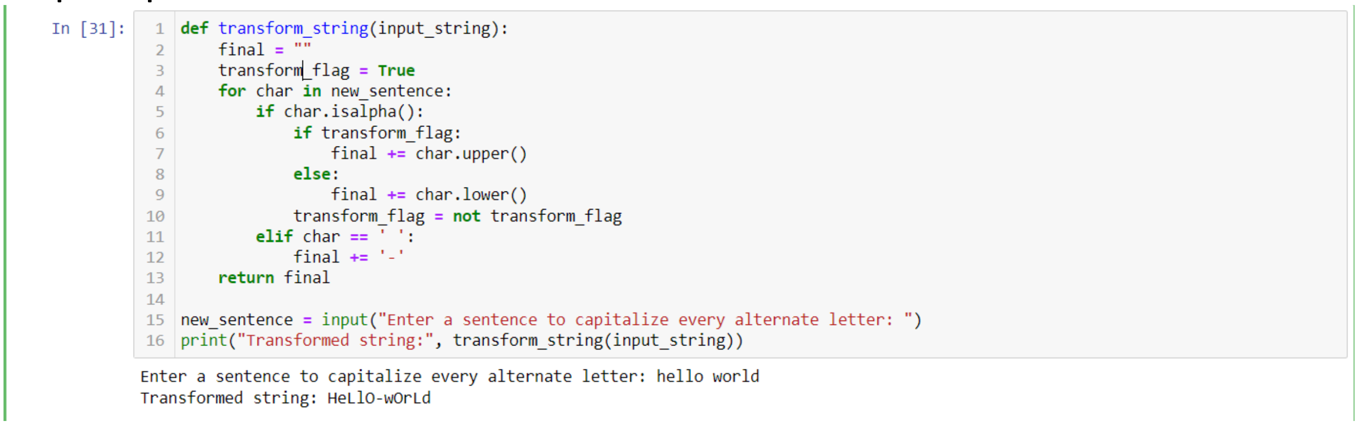
12. Create a Python function named custom_sort that takes a list of tuples where each tuple contains a name and a score. The function should return a new list sorted by scores in descending order. If two tuples have the same score, they should be sorted alphabetically by name in ascending order. Test your function with a predefined list of tuples and print the sorted list.
Sample Input: [('Alice', 88), ('Bob', 95), ('Charlie', 88), ('Dave', 95)]
Sample Output: [('Bob', 95), ('Dave', 95), ('Alice', 88), ('Charlie', 88)]



```
In [50]:
def custom_sort(data):
    sorted_data = sorted(data, key=lambda x: (-x[1], x[0]))
    return sorted_data

data = [('Alice', 88), ('Bob', 95), ('Charlie', 88), ('Dave', 95)]
sort_data = custom_sort(data)

print(sort_data)
```

```
[('Bob', 95), ('Dave', 95), ('Alice', 88), ('Charlie', 88)]
```

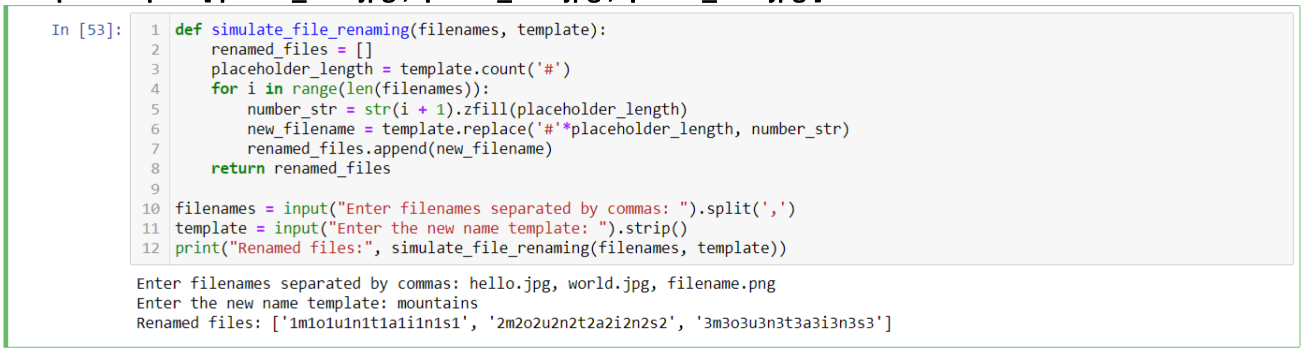
13. Develop a Python function named transform_string that takes a string and performs the following transformations: it capitalizes every other letter starting with the first character (ignoring non-letter characters for the alternation pattern), and it replaces spaces with hyphens (-). For example, hello world becomes HeLlO-WoRlD. After defining the function, ask the user for a string and print its transformation.
Sample Input: hello world
Sample Output: HeLlO-WoRlD



```
In [31]:
def transform_string(input_string):
    final = ""
    transform_flag = True
    for char in new_sentence:
        if char.isalpha():
            if transform_flag:
                final += char.upper()
            else:
                final += char.lower()
            transform_flag = not transform_flag
        elif char == ' ':
            final += '-'
    return final

new_sentence = input("Enter a sentence to capitalize every alternate letter: ")
print("Transformed string:", transform_string(input_string))
```

```
Enter a sentence to capitalize every alternate letter: hello world
Transformed string: HeLlO-wOrLd
```

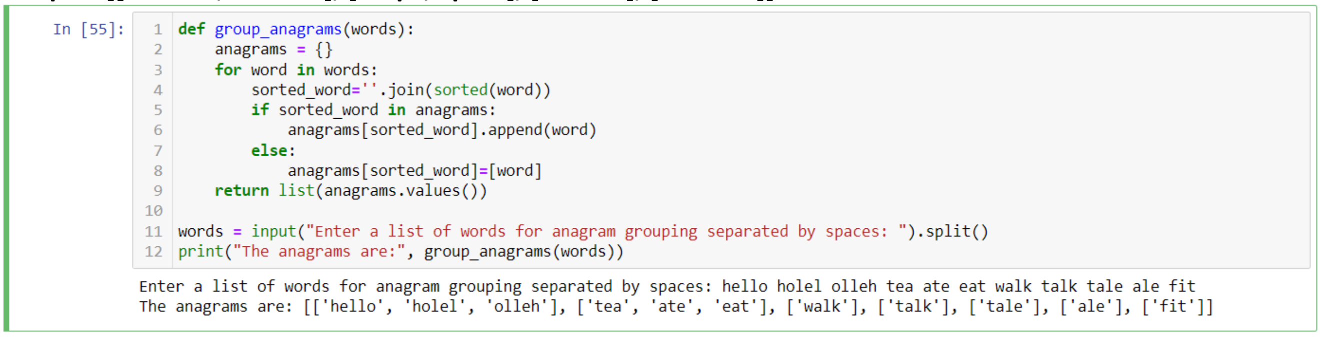
14. Create a function named simulate_file_renaming that takes two parameters: a list of filenames (as strings) and a new name template (a string containing a placeholder for a number, e.g., image_##). The function should return a list of strings representing the new filenames where the placeholder is replaced by an incremental number, starting from 1 and formatted to have leading zeros if necessary, according to the placeholder's length. For instance, renaming ['a.jpg', 'b.jpg', 'c.jpg'] with the template photo_### would result in ['photo_001.jpg', 'photo_002.jpg', 'photo_003.jpg']. This exercise simulates the renaming process, so you should only return the renamed list without actually renaming any files.
Sample Input: ['a.jpg', 'b.jpg', 'c.jpg'], photo_###
Sample Output: ['photo_001.jpg', 'photo_002.jpg', 'photo_003.jpg']



```
In [53]:
def simulate_file_renaming(filenames, template):
    renamed_files = []
    placeholder_length = template.count('#')
    for i in range(len(filenames)):
        number_str = str(i + 1).zfill(placeholder_length)
        new_filename = template.replace('#'*placeholder_length, number_str)
        renamed_files.append(new_filename)
    return renamed_files

filenames = input("Enter filenames separated by commas: ").split(',')
template = input("Enter the new name template: ").strip()
print("Renamed files:", simulate_file_renaming(filenames, template))
```

```
Enter filenames separated by commas: hello.jpg, world.jpg, filename.png
Enter the new name template: mountains
Renamed files: ['1m1o1u1n1t1a1i1n1s1', '2m2o2u2n2t2a2i2n2s2', '3m3o3u3n3t3a3i3n3s3']
```

15. You are given a list of words. Write a Python function called group_anagrams that groups all anagrams together and returns them as a list of lists.
Two words are considered anagrams if they contain the same characters but in a different order.
Examples:
Input: ["eat", "tea", "tan", "ate", "nat", "bat"]
Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]
Input: ["listen", "silent", "top", "pot", "hello", "world"]
Output: [["listen", "silent"], ["top", "pot"], ["hello"], ["world"]]



```
In [55]:
def group_anagrams(words):
    anagrams = {}
    for word in words:
        sorted_word=''.join(sorted(word))
        if sorted_word in anagrams:
            anagrams[sorted_word].append(word)
        else:
            anagrams[sorted_word]=[word]
    return list(anagrams.values())

words = input("Enter a list of words for anagram grouping separated by spaces: ").split()
print("The anagrams are:", group_anagrams(words))
```

```
Enter a list of words for anagram grouping separated by spaces: hello holel olleh tea ate eat walk talk tale ale fit
The anagrams are: [['hello', 'holel', 'olleh'], ['tea', 'ate', 'eat'], ['walk'], ['talk'], ['tale'], ['ale'], ['fit']]
```

16. You are given a list of integers. Write a Python function called max_subarray_sum to find the contiguous subarray within the list that has the largest sum and return that sum.
For example, given the list [−2, 1, −3, 4, −1, 2, 1, −5, 4], the contiguous subarray with the largest sum is [4, −1, 2, 1], and the maximum sum is 6.
Examples:
Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4]
Output: 6 (corresponding to the subarray [4, -1, 2, 1])
Input: [1, 2, 3, 4, 5]
Output: 15 (corresponding to the subarray [1, 2, 3, 4, 5])

In [36]:
```python
def max_subarray_sum(arr):
    if not arr:
        return 0
    max_sum = current_sum = arr[0]
    for num in arr[1:]:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)
    return max_sum

arr = list(map(int, input("Enter a list of integers for the subarray sum separated by spaces: ").split()))
print("The maximum subarray sum:", max_subarray_sum(arr))
```

```
Enter a list of integers for the subarray sum separated by spaces: 5 4 3 6 4 2 6 4 22 11
The maximum subarray sum: 67
```

17. Implement a function that performs a sequential search through a list for a specified target value. The function should return the index of the target if found, and -1 if the target is not in the list.
Sample Input: ([5, 3, 7, 1, 9], 7)
Sample Output: 2

In [63]:
```python
def sequential_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1

arr = [5, 3, 7, 1, 9]
target = 7
print("Index of target:", sequential_search(arr, target))
```

```
Index of target: 2
```

18. Design a method to encode a list of strings to a single string and another method to decode it back to a list of strings.
The encoded string should be concise and easily decodable. Assume there are no character restrictions for individual strings.
Examples:

> **Input: ["hello", "world"]**
> **Encoded Output: "5#hello5#world" (or another unique format of your choice)**
> **Decoded Output: ["hello", "world"]**
> **Input: ["abc", "def", "ghi"]**
> **Encoded Output: "3#abc3#def3#ghi"**
> **Decoded Output: ["abc", "def", "ghi"]**

In [39]:
```python
def encode(strings):
    encoded_string = ""
    for string in strings:
        encoded_string += str(len(string)) + "#" + string
    return encoded_string

def decode(encoded_string):
    decoded_strings = []
    i = 0
    while i < len(encoded_string):
        length = ""
        while encoded_string[i] != '#':
            length += encoded_string[i]
            i += 1
        i += 1
        length = int(length)
        decoded_strings.append(encoded_string[i:i+length])
        i += length
    return decoded_strings

strings = input("Enter a list of words to be encoded here (separated by commas): ").split(',')
encoded_output = encode(strings)
print("Encoded Output:", encoded_output)
decoded_output = decode(encoded_output)
print("Decoded Output:", decoded_output)
```

```
Enter a list of words to be encoded here (separated by commas): hello, world
Encoded Output: 5#hello6# world
Decoded Output: ['hello', ' world']
```