**Q1: Write a Python program to calculate number of days between two dates.**

**Sample dates: (2014, 7, 2), (2014, 7, 11) Expected output : 9 days**

```
from datetime import date


d1 = date(2014, 7, 2)

d2 = date(2014, 7, 11)


days = (d2 - d1).days


print(days, "days")
```

```
9 days

=== Code Execution Successful ===
```

**Q2: Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn**

```
n = int(input("Enter an integer: "))

n1 = n

n2 = int(str(n) * 2)

n3 = int(str(n) * 3)

result = n1 + n2 + n3

print("Result:", result)
```

```
Enter an integer: 6
Result: 738

=== Code Execution Successful ===
```

**Q3: Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2?**

```
number = int(input("Please enter any integer number: "))


if number % 2 == 0:
    print("You entered:", number)
    print("Result: The number is EVEN.")
    print("Explanation: When divided by 2, the remainder is 0.")
else:
    print("You entered:", number)
    print("Result: The number is ODD.")
    print("Explanation: When divided by 2, the remainder is not 0.")
```

```
Please enter any integer number: 5
You entered: 5
Result: The number is ODD.
Explanation: When divided by 2, the remainder is not 0.

=== Code Execution Successful ===
```

**Q4: Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.**

```
values = input("Enter comma-separated numbers: ")

number_list = values.split(",")

number_list = [int(num.strip()) for num in number_list]

number_tuple = tuple(number_list)

print("\nGenerated List:", number_list)
print("Generated Tuple:", number_tuple)
```

```
Enter comma-separated numbers: 10,80,20,30

Generated List: [10, 80, 20, 30]
Generated Tuple: (10, 80, 20, 30)

=== Code Execution Successful ===
```

**Q5: Write a Python program to calculate the sum of three given numbers, if the values are equal then return thrice of their sum.**

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))

total = a + b + c

if a == b == c:
    result = total * 3
    print("All numbers are equal.")
    print("Thrice of their sum is:", result)
else:
    print("The sum of the three numbers is:", total)
```

```
Enter first number: 4
Enter second number: 6
Enter third number: 8
The sum of the three numbers is: 18

=== Code Execution Successful ===
```

**Q6: Write a Python program to test whether a passed letter is a vowel or not.**

```
ch = input("Enter a single letter: ")


ch = ch.lower()
if ch in ('a', 'e', 'i', 'o', 'u'):
    print("The entered letter is a VOWEL.")
else:
    print("The entered letter is a CONSONANT.")
```

```
Enter a single letter: A
The entered letter is a VOWEL.

=== Code Execution Successful ===
```

```
Enter a single letter: G
The entered letter is a CONSONANT.

=== Code Execution Successful ===
```

**Q7: Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] and write a program that prints out all the elements of the list that are less than 5.**
**Extras: a) Instead of printing the elements one by one, make a new list that has all the elements less than 5 from this list in it and print out this new list.**

**b) Write this in one line of Python.**

**c) Ask the user for a number and return a list that contains only elements from the original list a that are smaller than that number given by the user.**

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

print("Original List:", a)

print("\nElements less than 5:")
for x in a:
    if x < 5:
        print(x)

new_list = []

for x in a:
    if x < 5:
        new_list.append(x)

print("\nNew List (elements less than 5):", new_list)
```

```
one_line_list = [x for x in a if x < 5]

print("\nOne-line List (elements less than 5):", one_line_list)


num = int(input("\nEnter a number: "))


user_filtered_list = [x for x in a if x < num]


print("Elements smaller than", num, "are:", user_filtered_list)
```

```
Original List: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

Elements less than 5:
1
1
2
3

New List (elements less than 5): [1, 1, 2, 3]

One-line List (elements less than 5): [1, 1, 2, 3]

Enter a number: 6
Elements smaller than 6 are: [1, 1, 2, 3, 5]

=== Code Execution Successful ===
```

**Q8: Create a program that asks the user for a number and then prints out a list of all the divisors of that number. (If you don't know what a divisor is, it is a number that divides evenly into another number. For example, 13 is a divisor of 26 because 26 / 13 has no remainder.)**

```
num = int(input("Enter a number: "))

divisors = []

for i in range(1, num + 1):
    if num % i == 0:
        divisors.append(i)

print("Divisors are:", divisors)
```

```
Enter a number: 6
Divisors are: [1, 2, 3, 6]

=== Code Execution Successful ===
```

**Q9: Take two lists, say for example these two: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]**

**and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.**

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

common = list(set(a) & set(b))

print("Common elements without duplicates:", common)

```
Common elements without duplicates: [1, 2, 3, 5, 8, 13]

=== Code Execution Successful ===
```

**Q10: Ask the user for a string and print out whether this string is a palindrome or not. (A palindrome is a string that reads the same forwards and backwards.)**

```
text = input("Enter a string: ")


text = text.lower().replace(" ", "")


if text == text[::-1]:
    print("The string is a Palindrome")
else:
    print("The string is Not a Palindrome")
```

```
Enter a string: HELLO
The string is Not a Palindrome

=== Code Execution Successful ===
```

**Q11: Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list a and makes a new list that has only the even elements of this list in it.**

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

even_list = [x for x in a if x % 2 == 0]

print(even_list)
```

```
[4, 16, 36, 64, 100]

=== Code Execution Successful ===
```

**Q12: Generate a random number between 1 and 9 (including 1 and 9). Ask the user to guess the number, then tell them whether they guessed too low, too high, or exactly right. (Hint: remember to use the user input lessons from the very first exercise)**

```
import random

number = random.randint(1, 9)

guess = int(input("Guess a number between 1 and 9: "))

if guess < number:
    print("Too low!")
elif guess > number:
    print("Too high!")
else:
    print("Exactly right!")
```

```
Guess a number between 1 and 9: 2
Exactly right!

=== Code Execution Successful ===
```

**Q13: Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.).**

```
num = int(input("Enter a number: "))


if num > 1:
   for i in range(2, int(num**0.5) + 1):
      if num % i == 0:
         print(num, "is Not a Prime number")
         break
   else:
      print(num, "is a Prime number")
else:
   print(num, "is Not a Prime number")
```

```
Enter a number: 5
5 is a Prime number

=== Code Execution Successful ===
```

**Q14: Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.**

```
def remove_duplicates(lst):
    new_list = []
    for item in lst:
        if item not in new_list:
            new_list.append(item)
    return new_list

# Example usage
original_list = [1, 2, 2, 3, 4, 4, 5, 1, 6]
result = remove_duplicates(original_list)
print("Original List:", original_list)
print("List without duplicates:", result)
```

```
Original List: [1, 2, 2, 3, 4, 4, 5, 1, 6]
List without duplicates: [1, 2, 3, 4, 5, 6]

=== Code Execution Successful ===
```

**Q15: Write a function that takes an ordered list of numbers (a list where the elements are in order from smallest to largest) and another number. The function decides whether or not the given number is inside the list and returns (then prints) an appropriate boolean.**

```
def is_number_in_list(ordered_list, number):
    return number in ordered_list

ordered_list = [1, 3, 5, 7, 9, 11]
num = int(input("Enter a number to check: "))

result = is_number_in_list(ordered_list, num)
print("Is the number in the list?", result)
```

```
Enter a number to check: 9
Is the number in the list? True

=== Code Execution Successful ===
```

**Q16: Implement a function that takes as input three variables, and returns the largest of the three. Do this without using the Python max() function!**

```
def largest_of_three(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Example usage
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
z = int(input("Enter third number: "))

largest = largest_of_three(x, y, z)
print("The largest number is:", largest)
```

```
Enter first number: 3
Enter second number: 6
Enter third number: 9
The largest number is: 9

=== Code Execution Successful ===
```

## Q17: Python program to perform read and write operations on a file.

```
import io

file = io.StringIO()

file.write("Hello, this is a sample file.\n")
file.write("We can write multiple lines.\n")

file.seek(0)
content = file.read()
print("File content:\n", content)

file.write("Adding a new line at the end.\n")

file.seek(0)
updated_content = file.read()
print("Updated file content:\n", updated_content)
```

```
File content:
 Hello, this is a sample file.
We can write multiple lines.

Updated file content:
 Hello, this is a sample file.
We can write multiple lines.
Adding a new line at the end.


=== Code Execution Successful ===
```

**Q18: Python program to copy the contents of a file to another file**.

```
import io


source_file = io.StringIO()
source_file.write("This is the content of the source file.\n")
source_file.write("It will be copied to the destination file.\n")


source_file.seek(0)


destination_file = io.StringIO()


content = source_file.read()
destination_file.write(content)


destination_file.seek(0)
copied_content = destination_file.read()


print("Content copied successfully!")
print("Destination file content:\n", copied_content)
```

```
Content copied successfully!
Destination file content:
 This is the content of the source file.
It will be copied to the destination file.


=== Code Execution Successful ===
```

## Q19: Python program to count frequency of characters in a given file.

```
import io

from collections import Counter


file_content = io.StringIO()

file_content.write("Hello world! \n")

file_content.write("We will count the frequency of each character.")


file_content.seek(0)


text = file_content.read()

freq = Counter(text)

print("Character frequencies:")

for char, count in freq.items():

    print(f"'{char}': {count}")
```

```
Character frequencies:
'H': 1
'e': 7
'l': 5
'o': 4
' ': 9
'w': 2
'r': 4
'd': 1
'!': 1
'
': 1
'W': 1
'i': 1
'c': 5
'u': 2
'n': 2
't': 3
'h': 3
'f': 2
'q': 1
'y': 1
'a': 3
```

## Q20: Python program to print each line of a file in reverse order.

```
import io

file_content = io.StringIO()
file_content.write("Hello World\n")
file_content.write("This is a sample file\n")
file_content.write("Python programming is fun\n")

file_content.seek(0)

print("Lines in reverse order:")
for line in file_content:
    print(line.strip()[::-1])
```

```
Lines in reverse order:
dlroW olleH
elif elpmas a si sihT
nuf si gnimmargorp nohtyP

=== Code Execution Successful ===
```

**Q21: Python program to compute the number of characters, words and lines in a file.**

```
import io

file_content = io.StringIO()
file_content.write("Hello World\n")
file_content.write("This is a sample file\n")
file_content.write("Python programming is fun\n")

file_content.seek(0)

num_lines = 0
num_words = 0
num_chars = 0

for line in file_content:
    num_lines += 1
    num_chars += len(line)
    num_words += len(line.split())

print("Number of lines:", num_lines)
print("Number of words:", num_words)
print("Number of characters:", num_chars)
```

```
Number of lines: 3
Number of words: 11
Number of characters: 60

=== Code Execution Successful ===
```

**Q22: Write a program that prompts the user to enter his name. The program then greets the person with his name. But if the person's name is 'Rahul' and exception is thrown and he is asked to quit the program.**

```
class QuitException(Exception):
    pass

try:
    name = input("Enter your name: ")

    if name == "Rahul":
        raise QuitException("Access denied. Please quit the program.")

    print(f"Hello, {name}! Welcome!")

except QuitException as e:
    print(e)
```

```
Enter your name: GARIMA
Hello, GARIMA! Welcome!

=== Code Execution Successful ===
```

**Q23: Write a program that accepts date of birth along with the other personal details of a person. Throw an exception if an invalid date is entered.**

```
from datetime import datetime

class InvalidDateException(Exception):
    pass

try:

    name = input("Enter your name: ")
    age = input("Enter your age: ")
    dob_input = input("Enter your date of birth (YYYY-MM-DD): ")

    try:
        dob = datetime.strptime(dob_input, "%Y-%m-%d")
    except ValueError:
        raise InvalidDateException("Invalid date format or value entered.")

    print("\nPersonal Details Entered Successfully:")
    print("Name:", name)
    print("Age:", age)
    print("Date of Birth:", dob.strftime("%Y-%m-%d"))

except InvalidDateException as e:
    print("Error:", e)
```

```
Enter your name: GARIMA
Enter your age: 21
Enter your date of birth (YYYY-MM-DD): 2004-10-21

Personal Details Entered Successfully:
Name: GARIMA
Age: 21
Date of Birth: 2004-10-21

=== Code Execution Successful ===
```

**Q24: Write a Regular Expression to represent all 10 digit mobile numbers. Rules: 1. Every number should contains exactly 10 digits. 2. The first digit should be 7 or 8 or 9 Write a Python Program to check whether the given number is valid mobile number or not?**

```
import re

pattern = r'^[789]\d{9}$'

mobile = input("Enter a 10-digit mobile number: ")

if re.match(pattern, mobile):
    print("Valid mobile number")
else:
    print("Invalid mobile number")
```

```
Enter a 10-digit mobile number: 9311607318
Valid mobile number

=== Code Execution Successful ===
```

**Q25: A spell checker can be a helpful tool for people who struggle to spell words correctly. In this exercise, you will write a program that reads a file and displays all of the words in it that are misspelled. Misspelled words will be identified by checking each word in the file against a list of known words. Any words in the user's file that do not appear in the list of known words will be reported as spelling mistakes. The user will provide the name of the file to check for spelling mistakes as a command line parameter. Your program should display an appropriate error message if the command line parameter is missing. An error message should also be displayed if your program is unable to open the user's file. Words followed by a comma, period or other punctuation mark are not reported as spelling mistakes. Ignore the capitalization of the words when checking their spelling.**

```
import sys
import string

known_words = {"this", "is", "a", "sample", "file", "with", "some", "text", "python", "program", "to", "check", "spelling", "mistakes"}

def clean_word(word):
    """Remove punctuation and convert to lowercase"""
    return word.strip(string.punctuation).lower()

def spell_checker(filename):
    try:
        with open(filename, "r") as file:
```

```
            content = file.read()
    except FileNotFoundError:
        print(f"Error: Unable to open file '{filename}'")
        return

    words = content.split()
    misspelled = set()

    for word in words:
        cleaned = clean_word(word)
        if cleaned and cleaned not in known_words:
            misspelled.add(cleaned)

    if misspelled:
        print("Misspelled words found:")
        for word in sorted(misspelled):
            print(word)
    else:
        print("No spelling mistakes found!")

if len(sys.argv) < 2:
    print("Error: Please provide a filename as a command line argument.")
else:
    filename = sys.argv[1]
```

spell_checker(filename)

```
A module you have imported isn't available at the moment. It will be
    available soon.

=== Code Execution Successful ===
```