Week 8

Write and run your programs with IDLE editor. Submit finished programs to CodeGrade. Note that some tasks have several steps (A, B, C, …) in CodeGrade.

IMPORTANT: End each input-command string with a newline symbol \n. For example:
`variable = input("Some text:\n")`

Task 1: Write a program that does trigonometric calculations according to the example run below. The program needs to be menu-based. Use precision of 3 decimals.

Example run 1:

```
Trigonometric Calculations:
1. Sine Calculation
2. Cosine Calculation
3. Inverse Sine Calculation
4. Inverse Cosine Calculation
5. Exit
Enter your choice (1/2/3/4/5):
6
Invalid choice. Please select a valid option.

Trigonometric Calculations:
1. Sine Calculation
2. Cosine Calculation
3. Inverse Sine Calculation
4. Inverse Cosine Calculation
5. Exit
Enter your choice (1/2/3/4/5):
1
Enter an angle in degrees:
46
The sine of 46.0 degrees is 0.719

Trigonometric Calculations:
1. Sine Calculation
2. Cosine Calculation
3. Inverse Sine Calculation
4. Inverse Cosine Calculation
5. Exit
Enter your choice (1/2/3/4/5):
3
Enter the sine value:
0.719
The inverse sine (in degrees) of 0.719 is 45.972
```

```
Trigonometric Calculations:
1. Sine Calculation
2. Cosine Calculation
3. Inverse Sine Calculation
4. Inverse Cosine Calculation
5. Exit
Enter your choice (1/2/3/4/5):
5
Bye!
```

Task 2: Your goal in this task is to create a random password generator. For the purposes of checking code with CodeGrade, we will be using Python's random package. HOWEVER, in real world, you shoud use "secrets" package, which gives truly random values (https://docs.python.org/3/library/secrets.html)

To have agreements on what the character set is, you need to import string package and agree with the following combinations:

```
LETTERS = string.ascii_letters
DIGITS = string.digits
SPECIAL = string.punctuation
```

Then the set of used characters is

```
COMB = LETTERS + DIGITS + SPECIAL
```

Then ask the user for the length of the password. For the purposes of submitting this task to CodeGrade, set python to use `random.seed(8292)`. That way you get the same result every time. Use `random.choice(COMB)`, where the string `COMB` is defined as above.

Example run 1:

```
Enter the length of the password:
0
Password length must be a positive integer.
Enter the length of the password:
16
Generated password: bQ:<AXWQ7`SZ@*i*
```

Task 3: Write a python program which Imports `datetime` class from `datetime` module. It asks for a datetime string in format: `"%Y/%m/%d %H:%M:%S"`. Convert that to a datetime object by using `strptime` function.

Then print the month, weekday, week number and day of the year according to the example below.

Example run 1:

```
Give a datetime string in format "%Y/%m/%d %H:%M:%S":
2023/12/03 13:31:26
Month: December
Weekday: Sunday
Week nr: 48
Day nr: 337
```

Task 4: Write a program in which the user is prompted to enter two dates in the format "`DD.MM.YYYY`". The program then converts these input strings into `datetime` objects. Depending which day is later, it calculates the time difference between the two dates so that the difference is always positive. It extracts the number of days from the time difference. Finally, it prints the number of days between the two dates according to the example below.

Example run 1:

```
Enter the first date (DD.MM.YYYY):
10.10.1960
Enter the second date (DD.MM.YYYY):
07.07.2023
The number of days between 10.10.1960 and 07.07.2023 is 22915
days.
```

Example run 2:

```
Enter the first date (DD.MM.YYYY):
27.10.2023
Enter the second date (DD.MM.YYYY):
7.5.1965
The number of days between 27.10.2023 and 7.5.1965 is 21357 days.
```

Task 5: You can use the `datetime` module to check whether a given date is valid. You can attempt to create a `datetime` object with the provided date, and if an exception is raised, it means the date is not valid.

What is an **exception**? We will consider exceptions properly in Lecture 10, but here is a simple Python example of using a try-except block to catch a ZeroDivisionError exception:

```
try:
    result = 10 / 0  # This is not allowed
except:
    print("Error: Division by zero")
```

In this example, the "try" block attempts to perform a division operation that would result in division by zero. Since this is an exceptional event, a `ZeroDivisionError` is raised, and the program gracefully handles the error in the "except" block by printing an error message. The advantage of the try-except structure is that an error does not crash the program

Write a program that asks the user to enter a date in `YYYY-MM-DD` format. Whether the date is correctly formed, it prints the answer according to the examples below.

Example run 1:

```
Enter a date in YYYY-MM-DD format:
2023-02-29
2023-02-29 is not a valid date.
```

Example run 2:

```
Enter a date in YYYY-MM-DD format:
2023-09-31
2023-09-31 is not a valid date.
```

Example run 3:

```
Enter a date in YYYY-MM-DD format:
2023-12-31
2023-12-31 is a valid date.
```