Exercise set 4

String functions, exception handling, Boolean-variables

1. Create an application that has the following information in separate variables (don't ask the values from user with input()):

   - Date (text)
   - Client's name (text)
   - Year of birth (int)
   - Balance (int)

   Combine all variables into a single new variable **that follows the following format exactly** (variables marked in red):

   John Doe (1984), balance: 345 €, updated on 11.9.2021.

   Small extra task: Get the date by using the date-module! (See materials part 1)

   Example of the application running

   ```
   Test Person (1982), balance: 1698 €, updated on 12.8.2020.
   ```

   Be VERY precise with the spaces and special characters while printing! f-string is recommended for this exercise!

   Filename of the exercise = ***exercise4_1.py***

   Typical code amount: **5-8 lines** (empty lines/comments not included)

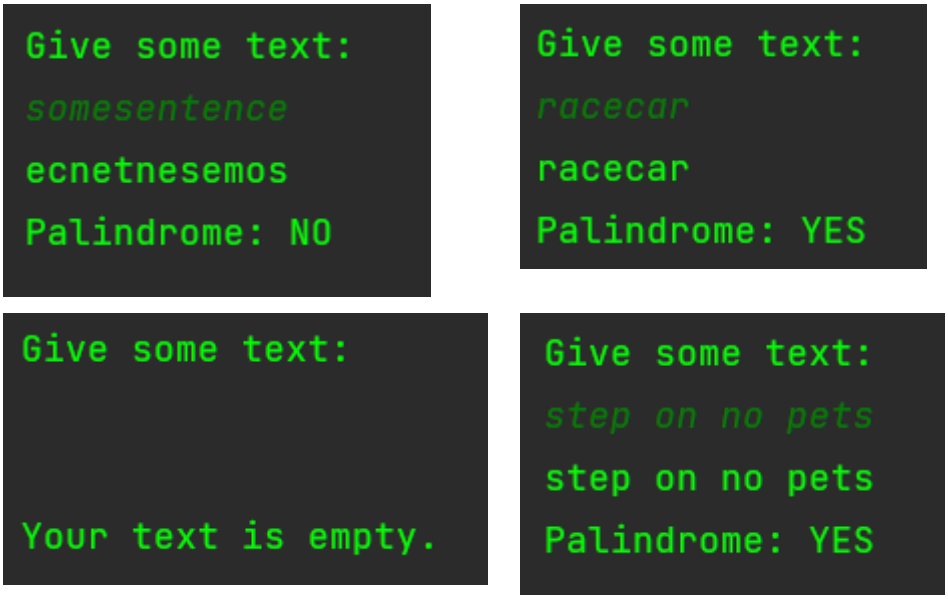
2. **Create an application that asks the user for some text, and save it to a variable.**
   - Reverse the text, and print it on the screen.

   - Check also, whether the given text is a palindrome or not (if the original text is identical to the reversed text, it is considered a palindrome. For example: ”racecar”.).

     If the word is a palindrome, print "Palindrome: YES", if not, print "Palindrome: NO".

   - If the user inputs an empty string, print "Your text is empty."

   Examples of the application running:



Tip: Use the same wording as in the examples. Otherwise, **codePost** might not understand how your application is working!

Filename of the exercise = ***exercise4_2.py***

Typical code amount: **7-10 lines** (empty lines/comments not included)

3. Save the following text into a variable (string) called **text**, and print it on the screen **(you can copy this code from Moodle too!):**

```
text = "The quick brown fox jumps over the lazy dog. That \
sentence contains every letter in the English alphabet. Neat!"

print(text)
```

NOTE! Everytime you do one of the tasks, DO NOT create a new variable for text, ALWAYS modify the **text**-variable, unless instructed otherwise!

a) Modify the **text**-variable so that you replace the word "fox" with the word "duck". **Print the modified** **text**-variable on the screen again.

b) Ask the user for a word that will be removed from the **text**-variable.

- **If** the word cannot be found in the **text**-variable, print "Word not found.".
- **If** the word was found in the **text**-variable, remove the word from the **text**-variable, and print the new version of the **text**-variable on the screen.

Tip: You can remove a word from a string by replacing the needed word with an empty string: ""

c) Print the number of characters in the **text**-variable.

Small extra task: Print also the number of words! in the **text**-variable

d) Replace the dots/periods in the **text**-variable into newlines. Print the modified **text**-variable on the screen.

e) Ask the user for a new sentence (with input()), and save it into a variable called **usertext**

- Create an if-statement: if **usertext** is 20 characters long or less, print **usertext**-variable as it is

- If **usertext**-variable is more than 20 characters long, shorten **usertext** to be exactly 20 characters long, and add three dots/periods "..." in the end of the **usertext**-variable. Print the modified **usertext**-variable on the screen.

  For example: If user inputs "This is a very long sentence indeed", this would be the result:

```
This is a very long ...
```

Example of the application running (image has been cropped horizontally):

```
The quick brown fox jumps over the lazy dog. That sentence cont

The quick brown duck jumps over the lazy dog. That sentence cor

Which word should be removed?:
quick
The  brown duck jumps over the lazy dog. That sentence contains

107 characters.

The  brown duck jumps over the lazy dog
 That sentence contains every letter in the English alphabet
 Neat!

Write a new sentence:
Rovaniemi is located close to the Arctic Circle.
Rovaniemi is located...
```

Filename of the exercise = ***exercise4_3.py***

Typical code amount: **18-28 lines** (empty lines/comments not included)

4. Ask the user for two different numbers. Create a calculation, where you divide the first number by the second number. If the user inputs text, print "Incorrect format.". If the user tries to divide by zero, print "You can't divide by zero.". **Use exception handling (try/except)**.

   Examples of the application running:

```
First number:
5
Second number:
3
1.6666666666666667
```

```
First number:
hello
Incorrect format.
```

```
First number:
11
Second number:
0
You can't divide by zero.
```

   Filename of the exercise = ***exercise4_4.py***

   Typical code amount: **9-16 lines** (empty lines/comments not included)

5. Create an application that asks the user whether they are a student or not (y / n) as well as the number of the month, when the user is planning to travel. Create a Boolean-variable in the application, which indicates, whether the user is entitled to a special price or not (for example, a variable called **special_price**)

   By using conditional statements, determine whether the user is entitled to a special price or not. If the user is entitled to it, change the **special_price** –variable to ***True***. If the user is not entitled to it, change it to ***False***. The user is entitled to a special price, **if they are a student** (y) and the reservation **is not** on months 6 - 8.

   Finally, based on the value in **special_price** –variable , print it on the screen if the user is entitled to a special price or not, for example:

```
if special_price:
   # print "Special price!”
else:
   # print "Normal price."
```

   Note: Do not ask the actual price in the exercise, the purpose is to only figure out if the user is entitled to special price or not.

   Tip: When checking the current month, you can use the Python's comparison for two values, for example: **if 0 < number < 10** etc. (basically comparing whether a variable if it's between 0 and 10)

   Examples of the application running:

```
Are you a student? (y/n)
y
Travel month? (1-12)
4
Special price!
```

```
Are you a student? (y/n)
y
Travel month? (1-12)
7
Normal price.
```

```
Are you a student? (y/n)
n
Travel month? (1-12)
11
Normal price.
```

   Filename of the exercise = ***exercise4_5.py***

   Typical code amount: **12-20 lines** (empty lines/comments not included)

ADVANCED EXERCISES:

6. Ask the user the current outside temperature and the humidity percentage. Create also the following Boolean-variables:
   - **freezing** = False
     heatwave = False
     raining = False
     hailstorm = False

   If the given temperature is less than 0, set the freezing-variable to True. If the humidity is over 80, set the raining-variable to True, however, if the temperature is lower than -20 degrees, set the hailstorm-variable to True ***instead*** of the raining-variable. If the temperature is over 20 degrees, set the heatwave-variable to True.

   Finally, print the given temperature on the screen. If the freezing-variable is True, print **"It's freezing outside."**. If the raining-variable is true, print **"It's raining."**. If the hailstorm-variable is True, print **"There's a hailstorm, be careful!"**. If the heatwave-variable is true, print **"Heatwave! Remember to hydrate!"**. If it's raining and there's a heatwave at the same time, print also **"It's damp and hot."**.

   ***Note**: Booleans are useful, when you want to make it easier to code conditions that are related to each other in a complex manner. Without Booleans, the conditions in conditional statements can get quite tricky at times!*

   Filename of the exercise = ***exercise4_6.py***

   Typical code amount: **20-36 lines** (empty lines/comments not included)

7. Create an application that converts numbers into words. For example, if the user inputs "1465", the application prints "one four six five". Allow only values of maximum of 5 different numbers.
    - **Extra task**: Allow the user also to convert words into numbers too. For example, "one two three" would result into 123

Filename of the exercise = ***exercise4_7.py***

Typical code amount: **24-32 lines** (empty lines/comments not included). If doing the extra task, the code lines can amount to even 48-60 lines.

It is possible to do this exercise with less code lines, but you'll need collections for that approach. We're going to cover collections later in the course!