Intro to programming 9

Henri Vandendriessche
henri.vandendriessche@ens.fr

2022-11-29

Exercise on logging

- Use my exercise correction from last week on the cameras and:
  - 1 log every argument/parameter sent to a function
  - 2 log every variable in a return statement

Exercise on pdb 1/3

- Use my exercise correction from last week on the cameras and:
  - 1 execute line by line the program using pdb
  - 2 print argument(s) of all function and check its type
  - 3 set breakpoints right before every return statement of a function

Exercise on pdb 2/3

- Debug the following adding program:

```
print('Enter the first number to add:')
first = input()
print('Enter the second number to add:')
second = input()
print('Enter the third number to add:')
third = input()
print('The sum is ' + first + second + third)
```

Exercise on pdb 3/3

- Debug the following coin toss program:

```
import random
guess = ''
while guess not in ('heads', 'tails'):
    print('Guess the coin toss! Enter heads or tails:')
    guess = input()
toss = random.randint(0, 1) # 0 is tails, 1 is heads
if toss == guess:
    print('You got it!')
else:
    print('Nope! Guess again!')
    guesss = input()
    if toss == guess:
        print('You got it!')
    else:
        print('Nope. You are really bad at this game.')
```

Exercise 1

- Write a program that asks for the first name, last name, age and date of birth of a user. Then create one or several functions that perform input validation
- Check that the age is a valid number,
- Check that the first name and last name are only letters
- Check that the date of birth is in the valid format dd/mm/yyy and is coherent with the their age
- Otherwise, ask again the invalid data

Exercise 2

- Write a Python program to create a Caesar cipher https://en.wikipedia.org/wiki/Caesar_cipher.
- With one function to encrypt and decrypt the text with a shift. Positive shift for encryption and negative shift for decryption.
- The program should consider uppercase and lower case
- Hint: check the function ord(), char() and unicode table.

Exercise 3

- Write a program that try to perform a Brute-force attack:
- using these collection of characters: first “abcdefghijklmnopqrstuvwxyz” then “abcdefghijklmnopqrstuvwxyz1234567890” and finally “abcdefghijklmnopqrstuvwxyz-ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890,;:!?./§%*+=”
- Ask the user a password (1 - 6 character)
- Then write a program that try every possible combination of our set of character with all possible length (from 1 to 5)
- Record and print the time to find the solution
- Write at every step the number length you are testing and time taken so far.
- To continue, calulate the number of combination according to the length of the password and the collection of character selected.
- Hint: check itertools module