Basic Exercises

March 21, 2022

1 Play with Python

Tutor: Francesca Cairoli

Email: francesca.cairoli@phd.units.it

Here you can find some basic exercises to get familiar with Python language.

Solving these exercises is **not** mandatory. If you feel like these exercises are too easy you can skip them. If, on the other hand, their solution is not that straightforward, we suggest you take some time to solve them.

In order to properly solve these exercises, look into the reference material shared in Moodle or search for help online and learn to read the manuals of numpy, scipy, matplotlib and so on.

If you are still stuck, do not hesitate to write me an email.

1. Define a function that takes as input a number and, depending on whether the number is even or odd, print out an appropriate message to the user.

```
[ ]: def even_odd(number):
         '''
         CODE HERE
         '''
```

```
[ ]: even_odd(5)
```

2. Define a function that takes as input a list and returns a new list containing all the elements of the original list that are less than 5.

What happens if the list contains elements that are not numbers? `example_list = [1, 2.4, 25, 'c', 77, "whatever"]`

```
[6]: def less_than_five(input_list):

         '''
         CODE HERE
         '''

         return
```

[ ]:
```
my_list = [] # choose a list
print(less_than_five(my_list))
```

3. Define a function that takes as input a number and returns a list of all the divisors of that number.

[21]:
```
def divisors(input_list):

    '''
    CODE HERE
    '''

    return
```

Extra: define a function that tells you if a certain input number is a prime number.

[ ]:
```
def is_prime(number):
```

4. Define a function that takes a string as input and print out whether this string is a palindrome or not. (A palindrome is a string that reads the same forwards and backwards.)

[8]:
```

```

5. Define a function that generates a random number between 1 and 100 (including 1 and 100) and ask the user to guess the number. The game should continue until the user guesses the number correctly or until they type "exit". You should tell the user whether they guessed too low, too high, or exactly right. Keep track of how many guesses the user has taken, and when the game ends, print this out.

Hint: use `x = input("your message")` to get inputs from the user, `x` is going to be a string. Remember to cast it to a number.

[ ]:
```

```

6. Matrix multiplication: given a matrix A a matrix B and a vector x, all defined as numpy arrays, figure out what is the difference between the `*` operation and the `@` operation

`A*x != A@x`

`A*B != A@B`

[13]:
```
import numpy as np
```

[20]:
```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
B = np.eye(3) #identity matrix
x = np.array([0.1,0.2,0.3])
y = np.ones(3)
z = np.zeros(3)
```

```
[ ]:
```

6.1. Find `x` such that `Ax = b`

```
[ ]:
```

7. Dictionaries:

7.1. Write a Python program to convert two lists into a dictionary in a way that item from `list1` is the key and item from `list2` is the value

`keys = ['Ten', 'Twenty', 'Thirty']`

`values = [10, 20, 30]`

Expected output: `{'Ten': 10, 'Twenty': 20, 'Thirty': 30}`

```
[ ]:
```

7.2. Merge two dictionaries into one:

`dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}`

`dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}`

```
[ ]:
```

7.3. Check if a certain key or a certain value exists in a dictionary

```
[ ]:
```

8.Plots: Use matplotlib library to plot the graph of a simple function, e.g. $f(x) = sin(x)$.

```
[23]: import matplotlib.pyplot as plt
```

```
[ ]:
```

9. Solving differential equations: Use `scipy.integrate` library to solve the following diffential equation and plot the solution:

$$\frac{dx}{dt} = e^{-3x} + 2x + 4$$

```
[ ]: from scipy.integrate import odeint
```