Universität Bremen

M.Sc. Neurosciences

Module MN-F3: Programming

Exercises #2 / Tutorial on 30.10.2025

Dr. Udo Ernst, Dr. David Rotermund, Dr. Joscha Schmiedt

1 Finger-flexing

a) Define a function `f` which computes $f(x) = \exp(-x) - 5\sin(x)$.

b) Define a function `sign` which takes a `number` as its argument and computes the following expression:

$$\text{sign(number)} := \begin{cases} -1 & : \text{number} < 0 \\ 1 & : \text{number} > 0 \\ 0 & : \text{otherwise} \end{cases} \tag{1}$$

c) Write a function `mean` that computes and returns the mean of the items in a list.

d) Write a function `meanvar` that computes and returns both mean and variance of the items in a list. Re-use `mean` to simplify your code. Test your function on the list `alist = [1, 17, 42]` and print the results.

e) **For-loop:** Factorials are defined as $n! = 1 \cdot 2 \cdot 3 \ldots n-1 \cdot n$. Write a program that computes $n!$ for arbitrary integer $n > 1$ using a `for`-loop. Do not use any pre-defined function such as `math.factorial` for that purpose.

f) **While-loop:** Find the minimum integer number n_{min} for which $n_{min}! > 10^{10}$. **Vectorization for experts:** Rewrite your code for computing $n!$ using NumPy such that it does not require a `for`-loop. What is the result for $n = 180$?

g) OPTIONAL for EXPERTS: **Log-i-tech^{TM}:** Rewrite your code such that it computes $\log(n!)$ instead. What is the corresponding result for $n = 180$?

2 Combinatorics for statistics: how high can you get?

Your goal in this exercise is to compute the binomial coefficient

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad ,$$

which you already know from the binomial distribution introduced in the statistics course.

a) Do the computation by using three subsequent `for`-loops. Compare your result with the `math`-function `math.comb` for the values $n = 18, k = 4$ and $n = 180, k = 40$.

b) OPTIONAL for EXPERTS:
In the previous exercise, you already learned that sometimes it is better to compute with logarithms as long as this is possible. Use this knowledge to rewrite your code. Do you obtain results which are more consistent with Numpy's function?

3 Data analysis: a first example

To study the interplay between loops and conditional execution, we consider a simple data analysis problem. Let's say we recorded some experimental trials whose lengths in seconds are stored in a list

```
t_rec = [1.2, 4.6, 3.1, -51.4, 7.1, 4.4, 5.2, 3.2, 1.4, 6.0]
```

A second list contains flags which tell us if a trial contains artifacts, or is clean

```
artifacts = [False, False, False, False, True, False, False, True, True, False]
```

We put ourselves into the position of a neuroscientist who wants to process that data and writes a program for it. Here we develop the code step by step, making it safer and cleaner by every extension.

a) At the start of your program, check if the two lists contain the same number of elements. Print a message if that's the case. If not, print an error message and stop the execution of the code. For this purpose you can use the following command:
`raise ValueError("String with an appropriate error message")`

b) After this check, write a loop that goes through all the trials. Compute the total length of available trials without artifacts, and print this number and also the percentage of valid trials in the whole list. When hitting a trial containing an artifact, print a message telling the user in which trial this was the case.

c) When observing the output from the previous task, you will wonder about the result. Sometimes the data you have contain unexpected values, and it is in your responsibility to detect such cases and raise errors or discard such data. You also have to inform the user with a message when this happens.

d) Sometimes you only need a specific amount of data available to you. Extend your program such that it stops going through the loop when it has a predefined amount of total recording time (let's say 15 seconds) accumulated.

Use functions for your program where appropriate.