The purpose of this training assignment is to repeat list comprehensions, to give some examples for slicing, work more with functions and investigate sets in Python. This assignment has 9 tasks.

**Warming-up Exercises**

You should *not* run the code before you have written down what it will do when executed.

**Task 1**

Assume we have stored the following values in a list:

\[
L = [0, 1, 2, 1, 0, -1, -2, -1, 0]
\]

What is the outcome of the following commands?

- \(L[0]\)
- \(L[-1]\)
- \(L[:-1]\)
- \(L + L[1:-1] + L\)
- \(L[2:2] = [-3]\)
- \(L[3:4] = []\)
- \(L[2:5] = [-5]\)

**Task 2**

What parts of the code below generate an error, and why?

```python
def f(x):
    return sin(x)

# does this work?
x = 3.
print(f())

# and this?
```
```python
print(f)

# what about:
y = 2*pi
print(f(y))
```

**Task 3**

What is the return value of this function?

```python
def f(m):
    L = [n-m/2 for n in range(m)]
    return 1 + L[0] + L[-1]
```

Can you write another function which has the same output?

What happens if you instead use integer division? That is, replacing in your code `/` by `//`.

**Programming Exercises**

**Task 4**

Let a distance table between some villages be given by the following data

```
  0  20  30  40
20  0  50  60
30  50  0  70
40  60  70  0
```

Construct a list (of lists) which contains this data. Call this list `distance`.

Construct from this list a list `reddistance` which contains only the relevant data in the following form

```
20  30  50
30  50  70
40  60  70
```

Use slicing for this task. Solve this task in two ways:

1. by using for-loops
2. by using *only* list comprehension and slicing (no for loops at all)

**Task 5**

Let $A$ and $B$ be sets. The set $(A \setminus B) \cup (B \setminus A)$ is called the *symmetric difference* of the two sets.
Write a function, which performs this operation. Compare your results to the result of the command `A.symmetric_difference(B)`.

**Task 6**

Study also other operations on sets. You find a complete list of those by using the “tab”-key in Spyder / IPython after `A`, where `A` is a previously defined set in Python. In particular study the method `update` and `intersection_update`. Explain what the difference between the two commands `intersection` and `intersection_update` is.

**Task 7**

Test in Python by considering a couple of examples the statement that the empty set is a subset of any set.
If you have no time left to make the following exercise during the training exercise session, you should do it at home, so that you are able to pose questions during the next lecture.

**Bisection Method**

A continuous function which changes its sign in an interval \([a, b]\) has at least one root in this interval. Such a root can be found by the **bisection method**.

This method starts from the given interval. Then it investigates the sign changes in the subintervals \([a, \frac{a+b}{2}]\) and \([\frac{a+b}{2}, b]\). If the sign changes in the first subinterval, \(b\) is redefined to be

\[
 b := \frac{a + b}{2}
\]

otherwise \(a\) is redefined in the same manner to be

\[
 a := \frac{a + b}{2}
\]

and the process is repeated until \(b - a\) is less than a given tolerance. Note, a sign change is characterized by the condition

\[
 f(a)f(b) < 0.
\]

**Task 8**

- Implement this method as a function `bisec`. It needs the initial interval \([a, b]\) and the tolerance as arguments.
- `bisec` should return the final interval and its midpoint.
- Test the method with the function `arctan` and also with the polynomial

\[
 f(x) = 3x^2 - 5
\]

in the interval \([-0.5, 0.6]\) and alternatively in \([-1.5, -0.4]\).

If you are uncertain of your results it might be helpful to plot the functions in the given intervals.

**Task 9**

In the next exercise we will work more with functions. A nice thing in Python is that functions are objects like everything else, and can be passed to other functions as input. Try changing your `bisec` method so that it also has \(f\) as an input argument, i.e. the first line could look like
```python
def bisec(f, interval, tol):

Then you can call it like
```
bisec(arctan, [1, 2], 1e-3)  # Not a good initial guess
```