Seminar 3

ES12003

2025–26

1 Home exercises

Exercise 1.1

Write a program that for a given string (e.g. `s = 'abcd'`) returns a new string where the first and last characters have been exchanged.

Solution 1.1

Using string indexing (recall that Python uses zero-based indexing, meaning the first character has index 0, the second 1, and so on):

```
s = 'abcd'
s[-1] + s[1:-1] + s[0] # works independently of the length of s
```

In words, you add together/concatenate three strings:

1. the last character of `s` (negative indices are used to count from the end of the string)
2. a substring that starts at the second character of `s` and includes all characters up to but not inclusive of the last character of `s`
3. the first character of `s`

Exercise 1.2

Write a program to get a string made of four copies of the last two characters of a user-supplied string (the latter can have any length greater than two).

Solution 1.2

```
1 string = input("Enter a string: ") # abcdefg
2 substring = string[-2:]
3 print(substring * 4)
```

`string[-2:]` extracts a substring that starts at the character before the last. Since `end` is not included, it is assumed to be equal to the length of the string. On the last line, the multiplication operator (`*`) is used to repeat `'fg'` the specified number of times.

Exercise 1.3

Write a program to remove the n^{th} character from a nonempty string (e.g., if you remove the fifth character from `starting`, you end up with `staring`).

Solution 1.3

```
1 s = 'starting' # example word
2 n = 5 # remove the fifth character
3 first_part = s[:n-1]
4 last_part = s[n:]
5 print(s, 'vs', first_part + last_part)
```

On line 3, we create a string starting at index `0` and ending at index `n-2` (i.e., `star`). On line 4, we create a string that includes all characters from the character at index n to the end of `s` (i.e., `ing`). Then, we print the concatenation of `first_part` and `second_part`, effectively removing the fifth character (that is the character at index `n-1`).

Exercise 1.4

Write a program to remove characters that have odd index values in a given string.

Solution 1.4

We remove characters at odd indices using slicing with step.

```
1 s = 'eoeoeoeoeoeoeoeo'
2 print(s[::2]) # eeeeeeee
```

When slicing strings, the step parameter controls how many characters to skip between each character in the slice. Thus, `s[::2]` takes every second character (indices $0, 2, 4, 6, \ldots$).[1]

Exercise 1.5

Predict the results of the following expressions and check them using JupyterLab.

```
1 not 1 < 2 or 4 > 2
```

```
2 not (1 < 2 or 4 > 2)
```

```
3 0 and 1
```

```
4 0 or 1
```

```
5 type(complex(2,3).real) is int
```

Solution 1.5

The operator `not` has higher precedence than the operator `or` (check the documentation on operator precedence). Thus, line 1 amounts to `False or True`, resulting in `True`. You could rewrite this line as `(not 1 < 2) or (4 > 2)`. Line 2, instead, amounts to `not True` (expressions inside parentheses are evaluated first).

Lines 3 and 4: the integer `0` is always `False`, while every other number is `True`. Logic operations between two numbers (Boolean values) return numbers (Boolean values). The `not` unary operator always returns Boolean value (try, e.g., `not 0`).

Line 5: `.real` returns the real part of the complex argument, which is a `float` (you can get the imaginary number by `.imag`).

Exercise 1.6

Create a program that checks if a given word is a palindrome.

[1] In programming, it is often possible to achieve the same result using different methods. On www.w3resource.com you may find another—albeit more complicated—solution to the same exercise.

Solution 1.6

Assuming that the palindrome is a word, like madam or racecar, the following program should do the job.

```
1 w = input("Enter a word: ") # e.g. level
2 w == w[::-1]
```

When you write `w[::-1]` and `w` is a string, you're using slicing to reverse the string. The slice notation has three parts: `start`, `stop`, and `step`. By leaving out the `start` and `stop` and setting the `step` to `-1`, you're telling Python to take the string from end to beginning, one character at a time. So, `w[::-1]` gives you a new string that is the reverse of the original. Finally, line 2 simply tests whether the specified word is identical to its inverse.

2 Class exercises

Exercise 2.1

Evaluate the golden ratio constant

$$\phi = \frac{1 + 5^{\frac{1}{2}}}{2}$$

and print it with three decimal places using an f-string.[2]

Exercise 2.2

According to Wikipedia, palindromes often consist of a sentence or phrase, e.g., "Do geese see God?" or "Was it a car or a cat I saw?". Punctuation, capitalisation, and spaces are usually ignored. Can you modify the code in home exercise 1.6 so that, besides words, sentences or phrases can be tested?

[2] F-strings are discussed on page 17 of the Topic 1 slides. The W3Schools tutorial on Python string formatting provides a clear overview of f-strings.