PHOT 110: Introduction to programming

Practical 13: exercises on data & dictionaries

Michaël Barbier, Spring semester (2024-2025)

1. Dictionary

You can create a dictionary using curly brackets

```
position = {"x": 1, "y": 1}
```

Perform the following tasks:

- Create a dictionary with four variables (position in space-time): “x”: 0, “y”: 1, “z”: 0, “t”: 0.
- Add this dictionary to an empty list.
- Then create a function that allows to add another dictionary of space-time coordinates
- Afterwards, create another function that allows you to convert the list of space-time coordinates to a single dictionary of 4 arrays x, y, z, and t, containing the coordinates.

2. Create a module

Put the functionality of previous exercise in a module. Remember to use a separate script-file for the module.

- Test each of the functions of the module. - Afterwards, try to import the module in a main script which plots the space-time coordinates in a 3D parametric curve plot.

```
def add_position(positions, x, y, z, t):
  # ...
  # code to add extend the list
  # ...
  return positions

def convert_to_dict(positions):
```

```
  # ...
  # code to convert the list of
  # dictionaries into a dictionary of lists
  # ...
  return coords
```

3. Drawing the Koch curve

Fractals are easiest defined by recursive algorithms. Look at the following code which draws (one side of) the Koch curve.

- Afterwards try to adapt the code to plot the whole Koch curve (the three sides of the snowflake)

```
import turtle

def koch(a, order):
    if order > 0:
        for t in [60, -120, 60, 0]:
            koch(a/3, order-1)
            turtle.left(t)
    else:
        turtle.forward(a)

# Draw the Koch curve using Turtle
turtle.setworldcoordinates(-10, -10, 100, 100)
turtle.goto
turtle.pensize(3)
turtle.pencolor("blue")
turtle.speed(10) # possible issue
koch(100, 4)
```