Aufgabenblatt 3: Funktionen

Aufgabe 1: Satzbau

a) Implementieren Sie ein Funktion namens `sentence`, welche nach Übergabe eines Subjekts, eines Prädikats und eines Objekts aus diesen drei Satzbestandteilen einen Satz konstruiert und diesen ausgibt.

b) Erstellen Sie mit Hilfe der Funktion ein Python-Progamm, welches die drei Satzteile einliest und anschließend ausgibt.

Lösung:

In [1]:
```
# Aufgabe 1
```

Aufgabe 2: Celsius nach Fahrenheit

Schreiben Sie ein Python-Script, welches den Benutzer dazu auffordert einen Celsius-Wert (Fließkommazahl) einzugeben und anschließend den entsprechenden, auf zwei Nachkommastellen gerundeten, Fahrenheit-Wert ausgibt. Der Aufruf des Scripts, bei dem der Benutzer den Celsius-Wert -12.715 eingibt, soll dabei folgende Ausgabe erzeugen:

Celsius: -12.715
Fahrenheit: 9.11

Hinweis: Zum Runden können Sie die Funktion round verwenden. Konsultieren Sie hierzu help(round) in einem Python-Feld. \n Hinweis: Strings können wie folgt zu Fließkommazahlen konvertiert werden:

`float('2.3450')`

`> 2.345`

Versucht man einen String, der keiner Fließkommazahl entspricht, zu konvertieren, wird die Ausführung des Programms durch eine Ausnahme zum Absturz gebracht:

float('not a number')
Traceback (most recent call last):
File "", line 1, in
ValueError: could not convert string to float: 'not a number'

In Ihrem Python-Script dürfen Sie dies ignorieren. Sie können also annehmen, dass der Benutzer stehts eine Fließkommazahl eingibt.

Lösung:

In [2]:
```
# Aufgabe 2
```

Aufgabe 3: Bibliotheken und Apps

In dieser Aufgabe sollen Sie die Mantelfläche eines Kegels berechnen und dabei Funktionen und mehrere Dateien verwenden.

Es können nicht nur Funktionen aus Modulen importiert werden. Auch Funktionsdefinitionen aus anderen Dateien lassen sich importiert werden. Die Datein müssen dabei im Python-Format abgespeichert werden und den Namen: `Dateiname.py` haben.

Beispiel:

`foo.py`

```
def my_function():
     print("Meine Funktion")

print("Das ist deine Funktion")
```

`lee.py`

```
from foo import my_function
my_function
```

Wird die Datei lee.py ausgeführt (in Jupyter oder auf in vscode), dann wird alles, was in der Datei foo.py steht, ausgeführt. Die Ausgabe ist folglich:

```
Das ist deine Funktion
Meine Funktion
```

Um dies zu verhindern, teilt man seinen Code meistens in zwei Arten von Python-Dateien auf:

- Python-Dateien, die selbst keine ausführbare Programme darstellen, sondern nur Funktionalität bereitstellen, die von anderen Python-Dateien importiert wird. Diese Dateien enthalten nur Definitionen und keine anderen Anweisungen. Eine Ansammlung von einer oder mehreren solcher Dateien nennt man auch eine Bibliothek (englisch Library, kurz lib).
- Python-Dateien, die den Einstiegspunkt in ein Programm darstellen. Diese Dateien enthalten auch Anweisungen, die nicht Teil einer Definition sind, aber können dafür auch nicht sinnvoll in andere Dateien importiert werden. Solche Dateien nennt man auch Applikationen (englisch applications, kurz app).

Diese Aufteilung bringt den Vorteil, dass man die Funktionalität aus den Bibliotheks-Dateien potentiell in mehreren Anwendungen wiederverwenden kann, ohne das Verhalten der Anwendungen auf ungewünschte Weise zu beeinflussen. Ein Beispiel für eine Bibliotheks-Datei ist das math-Modul.

a) Erstellen Sie die Datei cone_area_lib.py und definieren Sie dort eine Funktion cone_area, welche den Radius und die Höhe eines Kegels als Argumente vom Typ float entgegen nimmt und die Mantelfläche des Kegels als Wert vom Typ float zurückgibt. Die Funktion soll keine Seiteneffekte haben, d.h. sie soll ausschließlich die Argumente verwenden um die Mantelfläche zu berechnen und diese zurückgeben, aber keine Funktionen wie print oder input aufrufen. Beispielaufruf:

```
cone_area(3.0, 5.0)
> 54.96
```

b) Erstellen Sie die Datei cone_area_app.py und importieren Sie dort die Funktion cone_area aus der Datei cone_area_lib.py. Verwenden Sie die Funktionen cone_area, input und print um das gleiche Verhalten wie bei Aufgabe 2.3 zu erzeugen, d.h. das Ausführen von cone_area_app.py soll für einen Kegel mit Radius 3 und Höhe 5 folgende Ausgabe erzeugen: $ python3 cone_area_app.py Radius: 3.0 Höhe: 5.0 Mantelfläche: 54.96

Lösung:

In [3]:
```
# Aufgabe 3
```