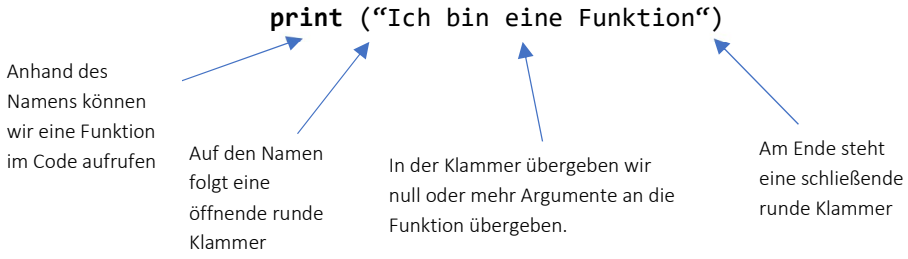
Lektion 6: Funktionen definieren und aufrufen

Oft ist es sinnvoll, einmal geschriebenen **Programmcode wiederzuverwenden**. Zudem ist es wichtig, dass Code nicht mehrfach vorhanden ist, da man sonst bei Änderungen leicht eine Stelle übersieht und der Code nicht mehr einheitlich funktioniert. Um das zu erreichen verwendet man Funktionen.

In Python kann man **eine neue Funktion mit dem Schlüsselwort "def" anlegen**. Man "definiert" also eine neue Funktion mit dem Namen der Funktion, mit runden Klammern und einem Doppelpunkt. Das ist - wie auch bei Kontrollstrukturen - der Kopf der Funktion. Im Rumpf der Funktion stehen die Befehle, die in der Funktion ausgeführt werden sollen.

Wenn man ein Skript ausführt, das nur eine Funktion enthält, geschieht zunächst noch nichts. Die Funktion wird nicht gestartet, sondern nur definiert und der Interpreter überspringt die Befehle der Funktion zunächst. **Will man die Befehle in der Funktion ausführen, so muss man die Funktion aufrufen.** Dazu schreibt man nur den Namen der Funktion gefolgt von runden Klammern. Erst bei einem solchen Aufruf springt der Interpreter in die Funktion hinein, führt die Befehle in der Funktion aus und springt dann wieder zu der Zeile des Aufrufs zurück.




```python
# ein Befehl
print("Erster Befehl")


# definiere eine Funktion mit diesem Funktionskopf
# die Funktion heißt "meine_erste_funktion"
def meine_erste_funktion():
    # Befehle im Rumpf der Funktion
    print("-> Erster Befehl der Funktion!")
    print("-> Zweiter Befehl der Funktion!")


# noch ein Befehl
print("Zweiter Befehl")

# rufe die Funktion auf
meine_erste_funktion()

# noch ein Befehl
print("Dritter Befehl")

```

```
Erster Befehl
Zweiter Befehl
-> Erster Befehl der Funktion!
-> Zweiter Befehl der Funktion!
Dritter Befehl
```

FACHOBERSCHULE BERUFSOBERSCHULE BAMBERG	Informatik		Datum:
	Name:	Klasse:	

Funktionen Argumente übergeben

Einige Funktionen haben Sie bereits verwendet, z.B. print(…), str(…), int(…) und range(…). Ohne das Mitgeben einer Zeichenkette wäre die "print()"-Funktion allerdings recht nutzlos. Woher sollte die Funktion dann wissen, welcher Text ausgegeben werden soll? Die "print()"-Funktion hat also ein sogenanntes **Argument**. Funktionen mit einem Argument erwarten, dass **beim Aufruf eine Variable mitgegeben wird** und können diese dann verwenden.

So könnte z. B. die gewünschte Temperatur als "soll_temperatur" an eine Funktion übergeben werden und diese Temperatur wird dann von der Heizung eingeregelt.

```
1  # gerade beträgt die Temperatur 21.4°C
2  aktuelle_temperatur = 21.4
3
4
5  # definiere eine neue Funktion mit dem Namen "setze_soll_temperatur"
6  # und gebe die aktuelle und die gewünschte Soll-Temperatur mit
7  def setze_soll_temperatur(ist_temperatur, soll_temperatur):
8      # regle die Temperatur ein, indem die Heizung eingeschaltet wird,
9      # wenn es aktuell noch kälter als die Soll-Temperatur ist
10     if ist_temperatur < soll_temperatur:
11         print("Heizung wird eingeschaltet")
12     # und indem die Heizung ausgeschaltet wird, wenn es warm genug ist
13     else:
14         print("Heizung wird ausgeschaltet")
15
16
17 # rufe die Funktion auf, die Temperatur wird auf 22.2°C geregelt
18 gewuenschte_temperatur = 22.2
19 setze_soll_temperatur(aktuelle_temperatur, gewuenschte_temperatur)
```

```
Heizung wird eingeschaltet
```

Man kann einer Funktion auch mehrere Argumente übergeben, dazu muss man diese nur durch Kommas trennen. Allerdings müssen immer alle Argumente übergeben werden, die die Funktion erwartet, sonst gibt der Interpreter einen Fehler aus.

Die Funktion wird (wie du bereits gesehen hast) am Ende des Rumpfs verlassen und der Interpreter springt zum Aufruf zurück.

Schlüsselwort return

Es gibt noch eine weitere Möglichkeit, Funktionen zu verlassen. **Trifft der Interpreter auf das Schlüsselwort "return", so kehrt er sofort zum Aufruf zurück.** Dabei kann man auch einen Wert aus der Funktion zurückgeben. Der Rückgabewert kann eine Zahl sein, eines der übergebenen Argumente oder eine Variable, die in der Funktion neu angelegt wurde. Man kann Kontrollstrukturen und Funktionen kombinieren und damit **komplexe Funktionalität in Funktionen implementieren**.

```python
# definiere die Maximum-Funktion, welche die größere Zahl der beiden
# Argumente zurückgibt
def maximum(x, y):
    if x > y:
        return x
    else:
        return y


# rufe die Funktion auf, die größere der beiden Zahlen wird
# zurückgegeben
maximum_wert = maximum(3.25, 4.71)
print("Maximum:", maximum_wert)

# Aufgabe: verändere die Kommazahlen im Funktionsaufruf.
# Welchen der Werte erhältst du?

```

```
Maximum: 4.71
```

FACHOBERSCHULE BERUFSOBERSCHULE BAMBERG	Informatik		Datum:
	Name:	Klasse:	

Funktionen Übung: Notenschnitt

Schreibe ein Python-Programm, das den Notenschnitt in einem Fach berechnet. Wir entwickeln hierzu einen Prototyp, den wir schrittweise verbessern können.

631_Notenschnitt.py

Es gibt zwei Arten von Fächern: Schulaufgabenfächer und Kurzarbeitsfächer. Programmiere zwei Funktionen, die bei Übergabe der entsprechenden Noten, den jeweiligen Schnitt für ein Schulaufgabenfach bzw. für ein Kurzarbeitsfach zurückgeben.

Die Berechnungsregeln sind wie folgt:

1. Schulaufgabenfach: Notenschnitt=((Note Schulaufgabe×1)+((Note Kurzarbeit×2+mündliche Note×1)/3))/2
2. Kurzarbeitsfach: Notenschnitt=(Note Kurzarbeit×2+mündliche Note×1)/3

Vorläufige Annahme: Die Schüler erhalten jeweils nur eine mündliche Note.

Teste deine Funktionen, in dem Du diese aufrufst und Du deine Noten als Argumente übergibst.

632_Notenschnitt_main.py

Schreibe ein Hauptprogramm mit der Funktion main(), das den Benutzer fragt, ob der Notenschnitt für ein Schulaufgabenfach oder ein Kurzarbeitsfach berechnet werden soll. Basierend auf der Auswahl des Benutzers, sollen die entsprechenden Noten abgefragt und der Notenschnitt berechnet und ausgegeben werden.

Für die weitere Verbesserung unseres Notenschnitt-Programms, benötigen wir Kenntnisse über die Datenstrukturen Dictionaries und Listen (siehe Kapitel 8 und 9 des Python Grundkurses)

633_Notenschnitt_Faecher.py

Definiere eine geeignete Datenstruktur (z.B. ein Dictionary), die speichert, ob ein bestimmtes Fach ein Schulaufgabenfach oder ein Kurzarbeitsfach ist. Das Fach Informatik ist z.B. eine Kurzarbeitsfach, das Fach Mathematik ist z.B. ein Schulaufgabenfach. Das Programm soll dann ermöglichen, dass der Nutzer den Namen des Faches eingibt und die Zuordnung zu Kurzarbeitsfach bzw. Schulaufgabenfach automatisch erfolgt.

634_Notenschnitt_Faecher_muendlich.py

Die vorläufige Annahme, dass es pro Fach nur eine mündliche Note gibt, soll aufgehoben werden.

Programmiere eine Funktion erstelle_muendliche_notenliste(), die dem Nutzer erlaubt mehrere mündliche Noten einzugeben (z.B. durch ein Leerzeichen getrennt) und diese Noten als Zahlenwert in einer Liste abspeichert und zurückgibt. Passe zudem die weiteren Funktionen zur Notenschnittberechnung entsprechend an.

Tipps: Hilfreiche Funktionen für die Implementierung: split(), append(), sum(), len()