Lambda-Funktionen (1)

- Auch der lambda-Operator ist uns schon begegnet. Wir haben ihn verwendet, um ein Sortierkriterium zu definieren. Beispiel aus Teil 5:

```
kontakte.sort(key=lambda x:x["name"])
```

- Eine **lambda-Funktion** ist eine anonyme Funktion, d.h. eine Funktion ohne Namen. Deshalb kann sie nicht unter Verwendung ihres Namens aufgerufen werden. Man kann sie aber z.B. als Sortier-Key verwenden.
- Eine lambda-Funktion wird mit dem Schlüsselwort lambda gefolgt von einer Parameterliste, einem Doppelpunkt und einem Ausdruck definiert:

lambda *parameterliste*: *ausdruck*

- Der Rückgabewert der lambda-Funktion ist der Wert des Ausdrucks. Beispiel:

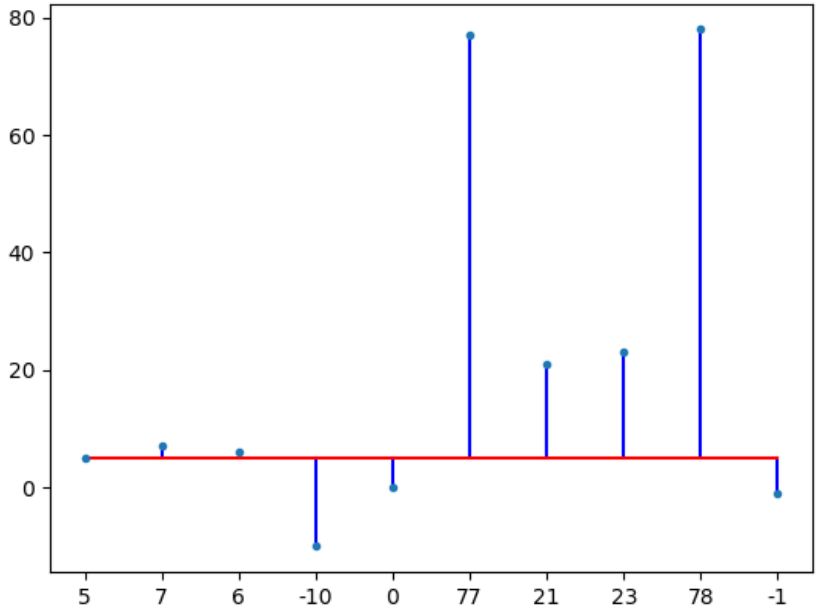
```
lambda x: abs(x - 5)
```

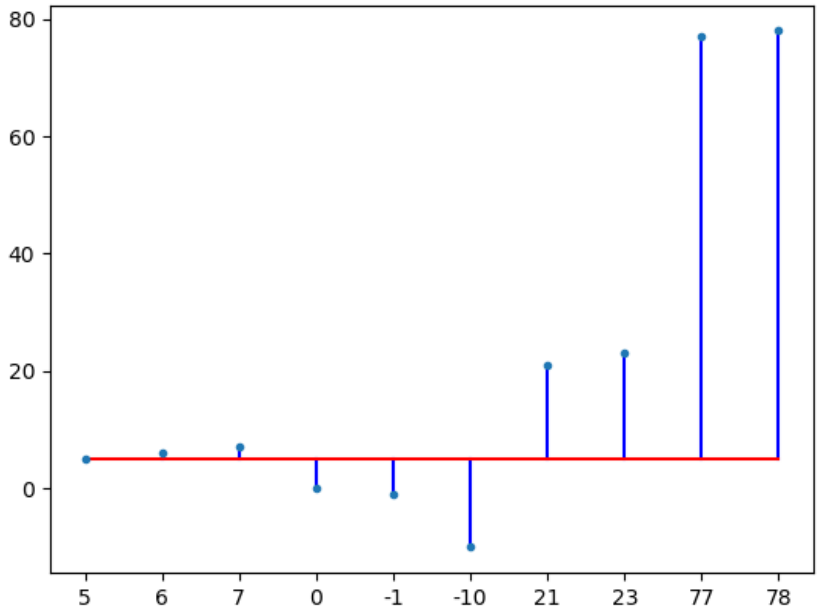
Lambda-Funktionen (2)

- Diese lambda-Funktion des letzten Beispiels berechnet den Abstand von x zu 5.
- Beispiel der Verwendung der lambda-Funktion als Sortier-Key:

```
zahlen = [5, 7, 6, -10, 0, 77, 21, 23, 78, -1]
zahlen.sort(key=lambda x: abs(x - 5))
print(zahlen)
```

- Die beiden Abbildungen zeigen die Abstände der Zahlen zu 5 vor und nach der Sortierung.





Lambda-Funktionen (3)

- Man kann eine lambda-Funktion auch einer Variablen zuweisen. Beispiel:

```
dist = lambda x, s: abs(x - s)
```

- Dann kann man die Funktion aufrufen. Beispiel:

```
print(dist(-10, 5))
```

- Das bringt aber nichts, weil man die Funktion gleich mit def definieren könnte.
- Man kann auch eigene Funktionen mit Funktionsparametern definieren. Beispiel:

```
def print_only(zahlen_liste, f):
    for zahl in zahlen_liste:
        if f(zahl):
            print(zahl, end=" ")
    print()


print_only(zahlen, lambda x: x > 0)
```

Lambda-Funktionen (4)

- Die Sortierfunktionen können auch ohne key-Parameter aufgerufen werden, da key ein Default-Parameter ist. Das kann man auch in selbst definierten Funktionen erreichen. Beispiel:

```
def print_only(zahlen_liste, f=lambda x: True):
    for zahl in zahlen_liste:
        if f(zahl):
            print(zahl, end=" ")
    print()

print_only(zahlen, lambda x: x > 0)        # Ausgabe: 5 7 6 77 21 23 78
print_only(zahlen)                         # Ausgabe: 5 7 6 -10 0 77 21 23 78 -1
```

Lambda-Funktionen (5)

- In Teil 3 hatten wir als Sortier-Key eine sign-Funktion:

```
def sign(x):
    if x < 0:
        return -1
    elif x > 0:
        return 1
    else:
        return 0
zahlen = [5, 7, 6, -10, 0, 77, 21, 23, 78, -1]
zahlen.sort(key=sign)
print(zahlen)                    # Ausgabe: [-10, -1, 0, 5, 7, 6, 77, 21, 23, 78]
```

- Mit einer lambda-Funktion geht das auch kürzer:

```
zahlen = [5, 7, 6, -10, 0, 77, 21, 23, 78, -1]
zahlen.sort(key=lambda x: -1 if x < 0 else (1 if x > 0 else 0))
print(zahlen)                    # Ausgabe: [-10, -1, 0, 5, 7, 6, 77, 21, 23, 78]
```

Lambda-Funktionen (6)

- In der lambda-Funktion auf der letzten Folie wurde ein if-else-Ausdruck mit folgender Syntax verwendet:

  wert1 if *bedingung* else *wert2*

  Anstelle des else-Wertes kann wieder ein if-else-Ausdruck eingesetzt werden:

  wert1 if *bedingung1* else (*wert2* if *bedingung2* else *wert3*)

- Else-Zweige sind hier zwingend erforderlich. Sonst hätte das Resultat keinen Wert, falls die Bedingung False ist.
- Die Parameterliste einer lambda-Funktion kann auch leer sein. Beispiel aus four_gui.py:

```
def zug(spalte): …
commands = [lambda :zug(0), lambda :zug(1), lambda :zug(2), lambda :zug(3), lambda :zug(4), lambda :zug(5), lambda :zug(6)]
buttons[i][j] = tk.Button(bg="white", width=5, height=2, command=commands[j])
```

- 🕮 Kapitel 30.1 in (Klein 2018)

Lambda-Funktionen (7)

Aufgaben

1. Im Praktikum wurde die Flugroute aufsteigend nach den x-Koordinaten der Ziele sortiert:

```
destinations.sort(key=itemgetter(0))
```

   Mit lambda-Funktionen geht das auch ohne itemgetter. Wie?

2. Geben Sie gleichwertige lambda-Funktionen für die folgenden Funktionen an:

```
def say_hello(name):
    return "Hello " + name


def distance(x1, y1, x2, y2):
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

   Weisen Sie die lambda-Funktionen zwei Variablen say_hello und distance zu und rufen Sie damit die lambda-Funktionen für beliebig gewählte Argumente auf.