Endprüfung Informatik (Lösungen)

Aufgabe 0

```python
x = int(input("Geben Sie x ein: "))
y = int(input("Geben Sie y ein: "))

if x > y:
    print(x)
elif x == y:
    print(x)
else:
    print(y)
```

(a) Angenommen x=3 und y=5, was ist der Output von diesem Programm? (1 Punkt)

> 1P: 5 oder y

(b) Angenommen x=1 und y=-7, was ist der Output von diesem Programm? (1 Punkt)

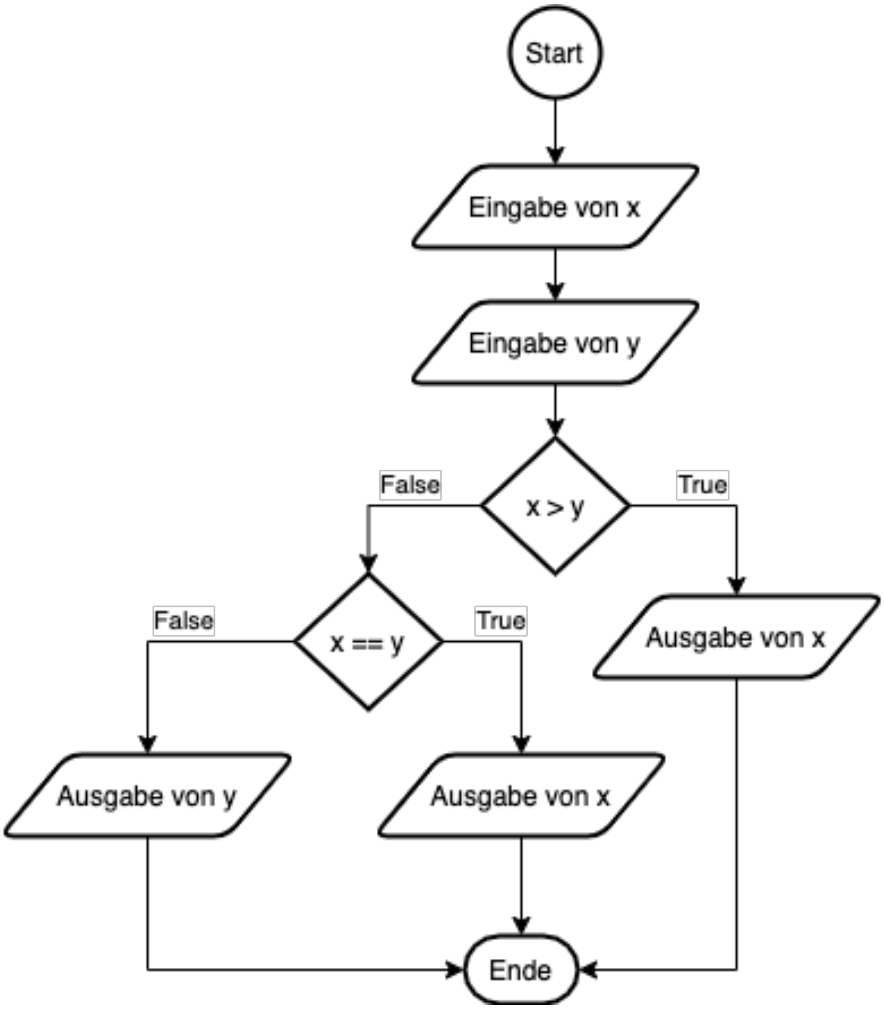
> 1P: 1 oder x

(c) Was macht dieses Programm? Begründen Sie stichwortartig. (1 Punkt)

> 1P: Maximum ausrechnen, welche Zahl grösser ist etc.

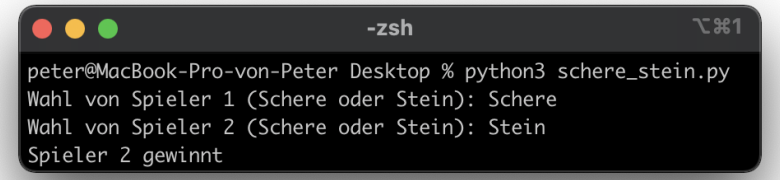
(d) Erstellen Sie ein Flussdiagramm dazu (2 Punkte)

> 2P: Vollständig korrekt oder ein kleiner Fehler z. B. "" oder ==
> 1.5P: Mehrere Fehler
> 1P: Logik-Fehler z. B. falsche oder zu viele Verzweigungen
> 0.5P: Sieht aus wie ein Flussdiagramm



Aufgabe 1

Programmieren Sie ein “Schere-Stein-Papier” Spiel, aber OHNE “Papier”. Dabei soll zuerst Spieler 1 eine Wahl treffen und danach Spieler 2. Das Programm soll dann als Output entweder “Spieler 1 gewinnt”, “Spieler 2 gewinnt” oder “Unentschieden” ausgeben. Es gelten die gleichen Regeln wie im normalen Spiel d. h. Stein schlägt Schere (Stein gewinnt). (3 Punkte)




```python
wahl1 = input("Wahl von Spieler 1 (Schere oder Stein): ")
wahl2 = input("Wahl von Spieler 2 (Schere oder Stein): ")

if wahl1 == "Schere":
    if wahl2 == "Schere":
        print("Unentschieden")
    elif wahl2 == "Stein":
        print("Spieler 2 gewinnt")

if wahl1 == "Stein":
    if wahl2 == "Schere":
        print("Spieler 1 gewinnt")
    elif wahl2 == "Stein":
        print("Unentschieden")
```

> 3P: Korrektes und funktionsfähiges Programm ("" vergessen ist ok)
> 2.5P: Ein (kleiner) Fehler z. B. 1x falsches/fehlendes Typecasting, and/or Syntax
> 2P: Richtige Logik/Überlegung, aber einige Fehler bzw. fehlender Code
> 1P: Es sieht aus wie Python

Aufgabe 2

```python
x = 6
i = 0
while i < 4:
    x = x + i
    i = i + 1
    print(x)
```

(a) Was sind die Outputs von diesem Programm? (2 Punkte)

> (0.5P für jede Zahl in der richtigen Reihenfolge)
> 0.5P: 6
> 0.5P: 7
> 0.5P: 9
> 0.5P: 12

(b) Schreiben Sie das Programm mit einer For-Schleife statt einer While-Schleife (1 Punkt)

```python
x = 6
for i in range(4):
    x = x + i
    print(x)
```

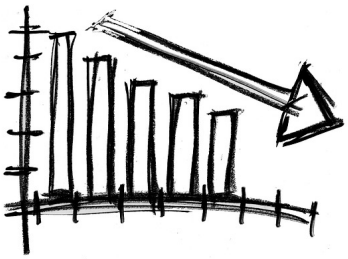
> 1P: Vollständig korrekt

(c) Warum sollten Sie wenn immer möglich For-Loops nehmen? Begründen Sie stichwortartig oder anhand eines Beispiels. (1 Punkt)

> 1P: For-Loops sind weniger fehleranfällig

Aufgabe 3

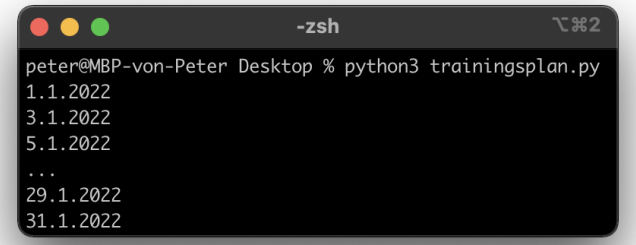
(a) Ihr Smartphone verliert jedes Jahr ca. die Hälfte des Wertes (je nach Marke). Erstellen Sie ein Programm, bei welchem Sie den Preis des Smartphones eingeben können und der Wert für jedes Jahr (bis 5 Jahre) ausgegeben wird. (3 Punkte)




```python
wert = int(input("Geben Sie den Preis ein: "))
for i in range(1, 6):
    wert = wert / 2
    print(str(i) + " Jahr: " + str(wert))
```

> 3P: Korrektes und funktionsfähiges Programm ("" vergessen ist ok)
> 2.5P: Ein (kleiner) Fehler z. B. 1x falsches/fehlendes Typecasting
> 2P: Richtige Logik/Überlegung, aber einige Fehler bzw. fehlender Code
> 1P: Es sieht aus wie Python

(b) Sie wollen fit werden und erstellen dazu einen Trainingsplan. Da Sie nicht jeden Tag trainieren sollten, entscheiden Sie sich dazu, **jeden zweiten Tag** ins Gym zu gehen. Erstellen Sie ein Programm, welches ihnen alle Trainingsdaten im Januar ausgibt, z. B. (3 Punkte)




```python
i = 1
while i <= 31:
    print(str(i) + ".1.2022")
    i = i + 2
```

> 3P: Korrektes und funktionsfähiges Programm ("" vergessen ist ok)
> 2.5P: Ein (kleiner) Fehler z. B. 1x falsches/fehlendes Typecasting
> 2P: Richtige Logik/Überlegung, aber einige Fehler
> 1P: Es sieht aus wie Python