Programmieren in Python | Beispielfagen

1 Grundlagen

1.1 Allgemeines

a) Welche Statements gibt es in Python **nicht**?

- ☐ if
- ☐ for
- ☐ def
- ☐ else if
- ☐ stop

b) Die Variablen a, und b sind mit numerischen Werten belegt. Welche der folgenden Ausdrücke werden garantiert zu **True** ausgewertet?

- ☐ `a < b`
- ☐ `(a < b) or (b < a)`
- ☐ `(a <= b) and (b <= a)`
- ☐ `(a == b) or (a < b) or ((b < a) and True)`

c) Welche der folgenden Ausdrücke werden zu **False** ausgewertet?

- ☐ `[el[1] for el in ["ab","cd"]] == ["a", "b"]`
- ☐ `len({"key_1": { "key_2": [4,5,6]}}) == 2`
- ☐ `str(2) == 2`

d) Die Datei `script.py` enthält folgenden Python Code.

```
vals = [1.3, 1, 3.3, 3, 5, 4]
for idx in range(len(vals)-1):
  if vals[idx] > vals[idx+1]:
    vals[idx], vals[idx+1] = vals[idx+1], vals[idx]
print(vals)
```

Listing 1: script.py

Wie lautet die Konsolenausgabe des Codes `script.py`?

1.2 Funktionale Programmierung

```python
def function(word: str):
    if len(word) > 0:
        print(word)
        function(word=word[:-1])
    else:
        return

function(word = "DHBW")
```

Listing 2: recursive.py

a) Wie lautet die Konsolenausgabe der Datei `recursive.py`

```python
def function_1(x: int) -> int:
    for i in range(x):
        print(i)
    return function_2(y=float(x))

def function_2(y: float) -> int:
    return int(y ** 2)

def function_3(z: int) -> int:
    print(z+1)
    return z

print(function_3(z=function_1(x=2)))
```

Listing 3: basicmath.py

b) Wie lautet die Konsolenausgabe der Datei `basicmath.py`

Die Funktion `best_student` bekommt eine Liste von Studenten übergeben. Sie findet den besten Studenten und gibt diesen zurück. Leider ist die vorgegebene Logik fehlerbehaftet.

```
1  class Student:
2      def init(name: str, grade: float):
3          self.name = name
4          self.grade = grade
5
6  def best_student(students: [Student]) -> int:
7      best = students[0]
8      for student in students.items():
9          if student.grade > best.grade:
10             student = best
11     return best
12
13 best = best_student([Student("Joe",2),
14             Student("Bob",1),
15             Student("Lea",3)])
16
17 print(f"best student: {best.name}, grade: {best.grade}")
```

Listing 4: grade.py

c) Markieren Sie alle 6 Fehler in den Zeilen 1 bis 11 in der `grade.py` Datei. Verbessern Sie den Code, indem Sie eine funktionsfähige Version im Antwortkasten implementieren. Die Zeilen 13 bis 17 sind fehlerfrei und müssen dabei nicht nochmals aufgeschrieben werden

d) Wie lautet die Konsolenausgabe ihres verbesserten und ausführbaren Codes ?

1.3 Objektorientierte Programmierung

Beziehen Sie sich in den nachfolgenden Aufgaben auf den Quellcode aus der `student.py` Datei. Dieser Code soll einen Studenten und das Lernen im Python Kurs Modellieren. Dabei kann ein Student mehrere Kurse belegen. Beim Lernen verbessert sich die Note des Kurses um einen Notenpunkt, wobei dieser Vorgang mit Energie (0 - 100%) bezahlt werden muss. Daher kann eine 1 nie ohne einmal zu schlafen erreicht werden. Beim Schlafen füllt sich die Energie des Studenten wieder auf 100 % auf.

```python
class Subject:
    def __init__(self, name: str):
        self.name = name
        self.grade = 6


class Student:
    def __init__(self, name: str, subjects: [Subject]):
        self.name = name
        self.energy = 100
        self.subjects = subjects

    def has_python_course(self):
        return "Python" in [subject.name for subject in self.subjects]

    def get_python_course(self):
        pass

    def study_python(self):
        if self.has_python_course():
            python_course = self.get_python_course()
            pass

    def __study(self):
        self.energy -= 25

    def has_energy_to_study(self):
        pass

    def sleep(self):
        self.energy = 100
```

Listing 5: student.py

a) Erstellen Sie eine Liste mit den Kursen **Mathe, Python** und **Java**

b) Erstellen Sie einen Studenten mit Namen **"Joe"**, welcher die in a) generierten Kurse belegt.

c) Implementieren Sie die Methode `has_energy_to_study()` der Student-Klasse. Diese gibt **True** zurückgeben, wenn die Energie des Studenten mehr als 25 % beträgt. Andernfalls wird **False** zurückgegeben.

d) Implementieren Sie die Methode `get_python_course()` Student-Klasse. Gehen Sie dabei wie folgt vor: In einer Schleife wird so lange die liste der subjects durchsucht, bis das Subject mit dem Namen "Python"gefunden wurde. Die Referenz dieses Objects wird dann zurück gegeben.

e) Vervollständigen Sie die Methode `study_python` der Student-Klasse. Gehen Sie dabei wie folgt vor: Überprüfen sie zunächst mit einer und-Verknüpfung, ob der Student über genügend Energie zum Lernen verfügt und ob im Fach Python noch keine 1 (beste Note) erreicht wurde. Ist beides gegeben verringern Sie die Note des Kurses **Python** um einen Notenpunkt und führen die private Methode **study()** aus.

f) Implementieren Sie den Lernvorgang von Joe im Kurs Python bis er eine 1 (beste Note) erreicht hat. **Achtung: Joe muss zwischen den Lern-Sessions auch eine Pause zum Erholen einlegen.**

2 Data Science

2.1 Numpy

```
1 import numpy as np
2
3 numpy_array = np.zeros((2, 3))
4 print(numpy_array)
5
6 numpy_array[1,2] = 1
7 numpy_array[1,1] = (numpy_array[1,2]+3) ** (1/2)
8 print(numpy_array)
```

Listing 6: numpyscript.py

a) Wie lautet die Konsolenausgabe aus Zeile 4?

b) Wie lautet die Konsolenausgabe aus Zeile 8?

2.2 Pandas

Im nachfolgenden Code wird ein Datensatz erstellt, welcher Informationen über Einkäufe in einem Supermarkt beinhaltet. Auf jeder Rechnung wird genau eine Sorte von Items abgerechnet. Zu jeder Rechnungsnummer gehören die Anzahl des gekauften Items, sowie der Stückpreis dieses Items.

```
import pandas as pd

data = {"Invoice_ID": [1234, 1235, 1236, 1237, 1238],
        "Units": [4, 3, 1, 3, 3],
        "Unit_Price": [1.0, 3.0, 1.5, 4.0, 5.0]}
df = pd.DataFrame(data=data)
df = df.set_index("Invoice_ID")
print(df)
```

Listing 7: pandasscript.py

a) Wie lautet die Konsolenausgabe, welche beim Ausführen der Datei `pandasscript.py` erzeugt wird?

b) Erweitern Sie die Variable **df** um eine eine Spalte **Total_Amount**, welche für jede **Invoice_ID** das Produkt aus **Units** und **Unit_Price** beinhaltet.

c) Nachdem Sie das DataFrame **df** in b) um die Spalte **Total_Amount** erweitert haben, wird folgende Zeile Code ausgeführt.

```
print(df.groupby("Units").mean())
```

Wie lautet die Konsolenausgabe?

d) Was ist der durchschnittliche Stückpreis für den Kauf von 3 Items auf einer Rechnung?

e) Anschließend wird folgender Code ausgeführt.

```
1 print(df.loc[[1234,1235,1236]]["Total_Sum"].sum())
```

Beschreiben Sie in wenigen Sätzen was mit dieser Zeile Code umgesetzt wird.

f) Was ist die Konsolenausgabe des in e) ausgeführten Codes?

2.3 Matplotlib

Abbildung 1: Unemployment Rate

```python
import pandas as pd
import matplotlib.pyplot as plt

data = {'year': [1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020],
        'unemployment_rate': [8, 7.2, 6.9, 7, 6.5, 6.2, 5.5, 6.3, 4.5]}

df = pd.DataFrame(data=data)

plt.bar(df['unemployment_rate'], df['year'], color='blue', marker='x')
plt.title('unemployment rate vs year', fontsize=14)
plt.xlabel('unemployment rate ', fontsize=0)
plt.ylabel('year', fontsize=1.4)
plt.show()
```

Listing 8: Matplotlib Chart

a) Der Code Matplotlib Chart soll die in Abbildung 1 dargestellte Arbeitslosenrate über die Jahre 1940 - 2020 erzeugen. Leider ist der Code ab Zeile 9 lücken- und fehlerhaft. Verbessern Sie den Code indem Sie die korrekte Version im Antwortkasten implementieren. Die Zeilen 1 bis 7 sind fehlerfrei und müssen deshalb nicht nochmals aufgeschrieben werden.