2024_portfolio

June 3, 2024

1 Portfolio Prüfung WWI2023B

Bitte bearbeiten Sie alle Aufgaben direkt hier im Notebook und geben Sie die .ipynb-Datei am Ende der Portfolio-Prüfung hier ab. Für Aufgaben die ohne Code erstellt werden, steht nach der Aufgabenstellung ein Markdown Antwortfeld zur Verfügung. Für Aufgaben bei denen ein Code verlangt wird, befindet sich nach der Aufgabenstellung ein interaktives Code-Feld (ggf. mit schon vorab ausgefüllten Code-Fragementen).

Viel Erfolg

1.1 Aufgabe 1 Python Basics

Ersetzen Sie die Fragezeichen im unten stehenden Code so, dass der Befehl `print(squared)` die Ausgabe `[1, 4, 9, 16, 25]` erzeugt. Verwenden Sie zur Erzeugung der Quadratzahlen die Variable `numbers`. Eine direkte Zuweisung von [1,4,6,9,16] an `squared` ist nicht erlaubt.

[ ]:
```
numbers = [1, 2, 3, 4, 5]
squared = [??]
print(squared)
```

(____ / 2 Punkte)

Ersetzen Sie die Fragezeichen im unten stehenden Code so, dass die Ausgabe der Code-Zeile `Hlo` lautet.

[ ]:
```
my_string = 'Hello WWI'
print(my_string[?:?:?])
```

(____ / 2 Punkte)

Gegeben ist das unten stehende Python-Dictionary. Ergänzen Sie die Funktion `calculate_average_grade()`, welche die Durchschnittsnote jedes Studenten zurückgibt. Die Funktion `calculate_average_grade(student_grades)` sollte ein Dictionary `av_grades` zurückgeben, in dem die Schlüssel die Namen der Schüler sind und die Werte ihre Durchschnittsnote.

[ ]:
```
student_grades = {
    'Alice': [85, 90, 92],
    'Bob': [70, 80, 75],
    'Charlie': [95, 85, 90]
}
```

```
def calculate_average_grade(list_of_grades):
    av_grades = {}

    pass # das "pass" sollten Sie durch geeigneten Code ersetzen

    return av_grades

print(calculate_average_grade(student_grades)) # Ausgabe: {'Alice': 89.0, 'Bob':
 ↪ 75.0, 'Charlie': 90.0}
```

(____ / 5 Punkte)

Schreiben Sie eine Python-Funktion namens `sum_of_digits`, die die Summe der Ziffern (die Quersumme) einer gegebenen Zahl berechnet.

Beispielausgaben:

`sum_of_digits(123)`: $1 + 2 + 3 = 6$

`sum_of_digits(456)`: $4 + 5 + 6 = 15$

[ ]:
```
def sum_of_digits(digits):
    pass # das "pass" sollten Sie durch geeigneten Code ersetzen

print(sum_of_digits(123))
print(sum_of_digits(456))
```

(____ / 4 Punkte)

2 Aufgabe 2 Komplexität von Algorithmen

Definieren Sie die Funktionen `funktion1, funktion2, funktion3` so dass sie die angegeebene Komplexität besitzen. Es soll gelten: | Funktion | Aufwand | |————|———| | funktion1 | $O(n)$ | | funktion2 | $O(n^2)$ | | funktion3 | $O(log(n))$ |

[ ]:
```
def funktion1(n):
    pass # das "pass" sollten Sie durch geeigneten Code ersetzen

def funktion2(n):
    pass

def funktion3(n):
    pass
```

(____ / 9 Punkte)

Sie können sich den Aufwand der von Ihnen definierten Funktionen (falls sie korrekt codiert sind) mit der folgenden Code-Zelle visualisieren lassen.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import timeit

ns = np.linspace(1, 1000, 50, dtype=int)

ts1 = [timeit.timeit(stmt=f'funktion1({n})',
                     #setup= f'',
                     globals=globals(),
                     number=3)
       for n in ns]

ts2 = [timeit.timeit(stmt=f'funktion2({n})',
                     #setup= f'',
                     globals=globals(),
                     number=3)
       for n in ns]

ts3 = [timeit.timeit(stmt=f'funktion3({n})',
                     #setup= f'',
                     globals=globals(),
                     number=3)
       for n in ns]

fig, (ax1, ax2, ax3) = plt.subplots(1,3)
ax1.plot(ns, ts1, 'ob')
ax2.plot(ns,ts2,'or')
ax3.plot(ns,ts3,'og')
```

3 Aufgabe 3 Array-List

Gegeben ist der folgende Code für eine array-backed-list:

[ ]:
```
# Array Backed List
import numpy as np
class ArrayList:
    def __init__(self):
        self.data = np.empty(1, dtype=object)
        self.size = 0

    def append(self, value):
        if self.size == len(self.data):
            ndata = np.empty(len(self.data)*2, dtype=object)
            for i in range(len(self.data)):
                ndata[i] = self.data[i]
            self.data = ndata
        self.data[self.size] = value
```

```
        self.size += 1

    def __setitem__(self,idx,value):
        """Implements `x = self[idx]`"""
        assert isinstance(idx, int), 'Index must be an integer'
        if idx < 0:
            idx += self.size
        if idx < 0 or idx >= self.size:
            raise IndexError('list index out of range')
        self.data[idx] = value

    def __len__(self):
        """Should return the number of elements in the ArrayList"""
        return self.size

    def __repr__(self):
        return str(self.data[:self.size])
```

3.1 a) Zugriff

Vervollständigen Sie die Definition und ergänzen Sie die Funktion `__getitem__(self, idx)` welche das Element an einem bestimmten Index zurückliefert.

[ ]:
```
class ArrayList(ArrayList):
    def __getitem__(self, idx):
        """Implements `x = self[idx]`"""
        pass # Sie sollten das "pass" durch geeigneten Code ersetzen.
```

Sie können ihre Funktion mit folgendem Code testen:

[ ]:
```
my_list = ArrayList()
for i in range(50):
    my_list.append(i)

print(my_list[0])  # Erwartete Ausgabe: 0
print(my_list[11])  # Erwartete Ausgabe: 11
print(my_list[-1]) # Erwartete Ausgabe: 49
```

(____ / 5 Punkte)

3.2 b) Löschen

Implementieren Sie eine Methode `remove(item)` für die obige ArrayList-Klasse. Diese Methode soll ein Element aus der ArrayList entfernen, falls es vorhanden ist.

Die Methode soll wie folgt funktionieren: - Wenn das angegebene Element `item` in der ArrayList enthalten ist, sollen **alle** Vorkommen des Elements aus der Liste entfernt werden. - Wenn das Element nicht vorhanden ist, soll die Methode keine Änderungen an der ArrayList vornehmen.

[ ]:
```
class ArrayList(ArrayList):
    def remove(self, item):
        """Removes all occurrences of the specified value from the ArrayList"""
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methode mit folgendem Code testen:

[ ]:
```
my_list = ArrayList()
my_list.append(10)
my_list.append(20)
my_list.append(30)
my_list.append(20)
print(my_list)  # Ausgabe: [10, 20, 30, 20]

my_list.remove(10)
print(my_list)  # Ausgabe: [20, 30, 20]

my_list.remove(20)
print(my_list)  # Ausgabe: [30]

my_list.remove(50) # Keine Änderung, da 50 nicht in der Liste enthalten ist
print(my_list)  # Ausgabe: [30]
```

(____ / 10 Punkte)

3.3 c) Suchen in der Liste

Implementieren Sie eine Methode `contains(value)` für die obige ArrayList. Diese Methode soll überprüfen, ob die ArrayList das angegebene Element `value` enthält.

Die Methode soll wie folgt funktionieren: - Rückgabe `True`, wenn die ArrayList das angegebene Element `value` enthält. - Rückgabe `False`, wenn die ArrayList das angegebene Element `value` nicht enthält.

[ ]:
```
class ArrayList(ArrayList):
    def contains(self, value):
        """Checks if the ArrayList contains the specified value"""
       pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methode mit folgendem Code testen:

[ ]:
```
my_list = ArrayList()
my_list.append(10)
my_list.append(20)
my_list.append(30)
print(my_list.contains(20)) # Ausgabe: True
print(my_list.contains(40)) # Ausgabe: False
```

(____ / 6 Punkte)

4 Aufgabe 4 Linked List

Gegeben ist der unten stehende Code für eine Linked List.

[ ]:
```
class LinkedList:
    class Node:
        def __init__(self, val, next):
            self.val = val
            self.next = next

    def __init__(self):
        self.top = None
        self.size = 0

    def prepend(self, value):
        newNode = LinkedList.Node(value,self.top)
        self.top = newNode
        self.size += 1

    def __iter__(self):
        node = self.top
        while node:
            yield(node.val)
            node = node.next

    def __repr__(self):
        return '[' + ', '.join(str(x) for x in self) + ']'
```

5 a) Definition vervollständigen

Der Liste fehlt noch die Methoden `__len__()` und `append()`. Implementieren Sie die `__len__`-Methode, welche die Anzahl an Elementen in der Liste zurück gibt. Erweitern Sie die `append(value)`-Methode so, dass sie ein Element mit dem Wert `value` ans Ende der Liste hängt.

[ ]:
```
class LinkedList(LinkedList):
    def __len__(self):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen

    def append(self, value):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methoden mit folgenden Code testen:

[ ]:
```
my_linked_list = LinkedList()
my_linked_list.append(10)
print(my_linked_list) # Ausgabe [10]
```

```
my_linked_list.append(20)
print(my_linked_list) # Ausgabe [10, 20]
my_linked_list.append(30)
print(my_linked_list) # Ausgabe [10, 20, 30]
```

(____ / 8 Punkte)

6 b) Stack

Die obige Klasse LinkedList enthält bereits einige grundlegende Methoden. Ihre Aufgabe ist es, diese Klasse in einen Stack umzuwandeln, indem Sie die Methoden push, pop und peek implementieren. - Implementieren Sie die Methode push(value), die ein neues Element mit dem angegebenen Wert oben auf den Stack legt. - Implementieren Sie die Methode pop(), die den Wert des obersten Elements vom Stack entfernt und zurück gibt. - Implementieren Sie die Methode peek(), die den Wert des obersten Elements auf dem Stack zurück gibt, ohne es zu entfernen.

[ ]:
```
class Stack(LinkedList):

    def push(self, value):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen

    def pop(self):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen

    def peek(self):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methoden mit folgenden Code testen:

[ ]:
```
my_stack = Stack()
my_stack.push(10)
my_stack.push(20)
my_stack.push(30)

print(my_stack) # Ausgabe [30, 20, 10]
print(my_stack.peek())  # Ausgabe: 30
print(my_stack.pop())   # Ausgabe: 30
print(my_stack.pop())   # Ausgabe: 20
print(my_stack.peek())  # Ausgabe: 10
```

(____ / 9 Punkte)

7 c) Liste sortieren

Implementieren Sie eine Funktion `sort()`, welche die Elemente der LinkedList nach ihrer Größe sortiert. Das kleinste Element sollte an erster Stelle stehen, und das größte Element sollte an letzter Stelle stehen.

[ ]:
```
class LinkedList(LinkedList):

    def sort(self):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methoden mit folgenden Code testen:

[ ]:
```
my_unsorted_list = LinkedList()
my_unsorted_list.prepend(10)
my_unsorted_list.prepend(5)
my_unsorted_list.prepend(20)
my_unsorted_list.prepend(15)

print(my_unsorted_list)        # Vor der Sortierung: [15,20, 5, 10]
my_sorted_list = my_unsorted_list.sort()
print(my_sorted_list)        # Nach der Sortierung: [5, 10, 15, 20]
```

(____ / 15 Punkte)

8 d) Liste umkehren

Implementieren Sie eine Methode `reverse()`, welche die Reihenfolge der Elemente einer LinkedList umkehrt. Das erste Element sollte zum letzten Element werden, das zweite Element zum vorletzten usw.

[ ]:
```
class LinkedList(LinkedList):
    def reverse(self):
        pass # das "pass" sollten Sie durch geeigneten Code ersetzen
```

Sie können Ihre Methoden mit folgenden Code testen:

[ ]:
```
my_list = LinkedList()
my_list.prepend(10)
my_list.prepend(5)
my_list.prepend(20)
my_list.prepend(15)

print(my_list) # Vor der Umkehrung: [15, 20, 5, 10]
my_list.reverse()
print(my_list)        # Nach der Umkehrung: [10, 5, 20, 15]
```

(____ / 15 Punkte)