portfolioTwoRepetition_completed

July 17, 2023

1 Portfolio Prüfung Wiederholung WWI2022B

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

Verwenden Sie jeweils eine Zeile Code um die im Kommentar beschriebene Funktionalität zu erzeugen.

[ ]:
```
# geben Sie "WWI rockt" als Output der Codezelle aus
print("WWI rockt")
```

```
WWI rockt
```

[ ]:
```
# Erstellen Sie eine Liste mit den Zahlen 1, 2, 3, 4 und weise Sie sie der␣
↪Variablen "zahlen" zu.
zahlen= [1,2,34]
```

[ ]:
```
# Erstellen Sie eine Range aller geraden Zahlen von 2 bis 100 (einschließlich)␣
↪und weisen Sie sie der Variablen "nummern" zu.
nummern = range(2,101,2)
```

(____ / 3 Punkte)

1.2 Aufgabe 2 Linked Queue

Gegeben ist der folgende Code für eine verlinkte Queue:

[ ]:
```
class LinkedQueue:
    class Node:
        def __init__(self,val, next=None):
            self.value = val
            self.next = next
```

```
    def __init__(self) -> None:
        self.head = self.tail = None

    def enqueue(self, value):
        # enqueues an element at the end of the queue
        if self.tail:
            self.tail.next = self.tail = LinkedQueue.Node(value)
        else:
            self.head = self.tail = LinkedQueue.Node(value)

    def __iter__(self):
        node = self.head
        while node:
            yield node.value
            node = node.next

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

1.2.1 a) Dequeue und Empty

Implementieren Sie die Funktionen `dequeue()`, welche das erste Element der Liste zurückgibt und aus der Liste löscht. Implementieren Sie zusätzlich die Funktion `empty()`, welche `True` zurückliefert, wenn die Liste leer ist, ansonsten `False`.

[ ]:
```
class LinkedQueue(LinkedQueue):
    def dequeue(self):
        assert not self.empty()
        retValue = self.head.value
        self.head = self.head.next
        if self.head is None:
            self.tail = self.head

        return retValue


    def empty(self):
        return self.head == None
```

(____ / 6 Punkte)

[ ]:
```
myQueue= LinkedQueue()
for i in range(5):
    myQueue.enqueue(i)

while not myQueue.empty():
    print(myQueue.dequeue())
print(myQueue.empty())
```

```
0
1
2
3
4
True
```

1.2.2 b) Mittelwert aller Elemente

Implementieren Sie eine Funktion `mean(self)`, welche den Mittelwert aller Elemente der LinkedQueue berechnet und zurückgibt. Gehen Sie davon aus, dass die Liste nur Knoten mit Zahlen als Werten enthält.

Beispiel: Der Mittelwert der LinkedQueue mit den Werten [1,2,3,4] beträgt 2,5.

[ ]:
```python
class LinkedQueue(LinkedQueue):
    def mean(self):
        actNode = self.head
        size = 0
        sum = 0
        while actNode:
            sum += actNode.value
            size +=1
            actNode = actNode.next

        return sum / size
```

(____ / 3 Punkte)

[ ]:
```python
myQueue = LinkedQueue()
for i in range(5):
    myQueue.enqueue(i)
print(myQueue.mean())
```

```
2.0
```

1.2.3 c) Einfügen vor

Implementieren Sie eine Funktion `addBefore(self, value, targetNodeValue)`, welche einen Knoten mit dem Werte `value` direkt vor dem ersten Knoten mit dem Wert `targetNodeValue` einfügt. Geben Sie eine Meldung aus, falls die Liste leer ist oder kein Knoten der Liste den Wert `targetNodeValue` besitzt.

[ ]:
```python
class LinkedQueue(LinkedQueue):
    def addBefore(self, value, targetNodeValue):
        assert not self.empty()

        # head value == targetNodeValue so prepend
        if self.head.value == targetNodeValue:
            newNode = LinkedQueue.Node(value, self.head)
```

```
            self.head = newNode
            return

        # check all other nodes
        node = self.head
        while node.next:
            if node.next.value == targetNodeValue:
                newNode = LinkedQueue.Node(value,node.next)
                node.next = newNode
                return
            node = node.next

        #if we end up here, we didn't found the targetNode so raise an Keyerror
        raise(KeyError("targetNodeValue not found"))
```

(____ / 6 Punkte)

[ ]:
```
myQueue = LinkedQueue()
for x in range(4):
    myQueue.enqueue(x)

print(myQueue)
myQueue.addBefore(10,1)
print(myQueue)
```

```
[0, 1, 2, 3]
[0, 10, 1, 2, 3]
```

1.3 Aufgabe 3 Komplexität von Algorithmen

Schreiben Sie jeweils eine beliebige Funktion, welche eine Python Liste als Eingabe erhält und die vorgegeben Laufzeitkomplexität hat.

Beispiel:

Erstellen Sie eine Funktion mit konstanten Aufwand $O(1)$:

[ ]:
```
myList = [x for x in range(10)]
# define a function with O(1) complexity
def functionExample(myList):
    return myList[0]
```

1.3.1 a) Linearer Aufwand

Definieren Sie eine Funktion mit linearem Aufwand $O(n)$.

[ ]:
```
myList = [x for x in range(10)]
# define a function with O(n) complexity
def function1(myList):
```

```
    pass
```

(____ / 3 Punkte)

1.3.2 b) Quadratischer Aufwand

Definieren Sie eine Funktion mit quadratischem Aufwand $O(n^2)$.

[ ]:
```
myList = [x for x in range(10)]
# define a function with O(n^2) complexity
def function2(myList):
    pass
```

(____ / 3 Punkte)

1.3.3 c) Exponentieller Aufwand

Definieren Sie eine Funktion mit exponentiellem Aufwand, z.B. $O(2^n)$ oder $O(3^n)$.

[ ]:
```
myList = [x for x in range(10)]
# define a function with e.g. O(2^n) or O(3^n) complexity
def function3(myList):
    pass
```

(____ / 3 Punkte)

1.3.4 d) Logarithmischer Aufwand

Definieren Sie eine Funktion mit logarithmischem Aufwand $O(\log(n))$.

[ ]:
```
myList = [x for x in range(10)]
# define a function with O(log(n)) complexity
def function4(myList):
    pass
```

(____ / 3 Punkte)

1.4 Abgabe

Bitte denken Sie daran, ihre Notebook-Datei auf den Abgabe-Server hochzuladen.

https://privacy.dhbw-stuttgart.de/wwi2022b.html