Practicum woensdag

Collapse context

Your answer passed the tests! Your score is 100.0%. [Submission #670fc5d63baf09455ab80e17 (**2024-10-16 13:55:34**)]

Question 1: Midden

✓ Perfect

Gebruik tuples om coördinaten van punten voor te stellen. Schrijf een functie `midden` om het midden van twee punten voor te stellen. Denk eraan: $co(M) = \frac{co(A)+co(B)}{2}$.

```
def midden(m, n):
    return (m[0] + n[0])/2 , (n[1] + m[1]) / 2
```

Submit

Question 2: Midden uitgebreid

✓ Perfect

Gebruik tuples om n dimensionale coördinaten voor te stellen. Schrijf een functie `midden(A,B)` die de coördinaten van het midden M van het lijnstuk [A,B] teruggeeft. Hint: maak eerst een lijst aan en zet deze achteraf om naar een tuple.

```
def midden(X,Y):
    result = []
    for i in range(len(X)):
        result.append((X[i] + Y[i]) / 2)
    return tuple(result)
```

Submit

Question 3: Getallen tellen

✓ Perfect

Om te tellen hoeveel keer een getal voorkomt in een lijst, kan je de count gebruiken.

```
>>> l = [1, 2, 5, 3, 2, 17, 5, 6, 7, 3, 2, 1, 9, 3]
>>> l.count(2)
3
```

Schrijf een functie `tel_getallen` die een lijst van getallen krijgt en een nieuwe lijst teruggeeft met het aantal keer dat elk getal voorkomt. Maak enkel gebruik van lists en zorg ervoor dat de frequenties in de juiste volgorde worden toegevoegd aan de nieuwe lijst.

```
>>> l = [1, 2, 5, 3, 2, 17, 5, 6, 7, 3, 2, 1, 9, 3]
>>> frequenties = tel_getallen(l)
>>> print(frequenties)
[0, 2, 3, 3, 0, 2, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1]
>>> print("Het getal 5 komt", frequenties[5], "keer voor")
Het getal 5 komt 2 keer voor
>>> print("Het getal 9 komt", frequenties[9], "keer voor")
Het getal 9 komt 1 keer voor
```

```python
def tel_getallen(l):
    counted = []
    for i in range(0, max(l) + 1):
        if i not in counted: print(i, l.count(i))
        else: counted.append(i)
```

Submit

Question 4: Word counter (a)

✓ Perfect

Schrijf een functie `tel_woorden` die een lijst van woorden krijgt en een dictionary teruggeeft (sleutel = woord en waarde = het aantal keer dat het woord voorkomt). Er wordt geen output getoond op het scherm.

Je kan het eens testen met een grote file. Download alice_book.py en plaats het in dezelfde map als je main.py. Het bevat het volledige verhaal van "Alice in Wonderland" zonder leestekens en in kleine letters. Je zet dan het volgende bovenaan je code:

```
from alice_book import book

# woorden is een lijst van alle woorden uit het boek
woorden = book.split()
```

Voor je inzending in Inginious moet je enkel de functie `tel_woorden` insturen, de test-code doet de rest!

```python
def tel_woorden(woorden):
    word_times = dict()

    for word in woorden:
        if word in word_times:
            word_times[word] += 1
        else:
            word_times[word] = 1
    return word_times
```

Submit

Question 5: Word counter (b)

✓ Perfect

Schrijf een functie print_frequenties die als input de dictionary krijgt uit de vorige oefening en alle woorden afdrukt met de bijhorende frequentie. Pas je functies toe op "Alice in Wonderland". De eerste 10 lijnen van de output zien er als volgt uit:

```
project 87
gutenbergs 2
alices 17
adventures 11
in 428
wonderland 8
by 76
lewis 4
carroll 4
this 181
```

Je moet je functie uit de vorige oefening niet opnieuw schrijven, Inginious gebruikt voor deze code ook jouw code van de vorige vraag.

```
1  def print_frequenties(words):
2      for word in words:
3          print(word, words[word])
```

Submit

Question 6: Een dictionary maken

✓ Perfect

Schrijf een functie `list2dict` die twee lijsten krijgt en er een dict van maakt. De eerste list bevat de keys, de tweede de waarden. Als er iets mis is, geef je None terug. Gebruik geen ingebouwde functies zoals zip.

```
>>> list2dict(['Ten', 'Twenty', 'Thirty'],[10, 20, 30])
>>> {'Ten': 10, 'Twenty': 20, 'Thirty': 30}
```

```
1  def list2dict(keys, values):
2      result = {}
3      if len(keys) != len(values): return None
4
5      for j in range(len(keys)):
6          result[keys[j]] = values[j]
7      return result
```

Submit

Question 7: Minigolf

De proffen hebben besloten om op een namiddag minigolf te spelen. Ze hebben hun scores bijgehouden in een dictionary, waarbij de key de naam van een prof is en de value een lijst van het aantal slagen dat ze nodig hadden per hole. Schrijf een functie `bepaal_winnaar` die de naam teruggeeft van de winnaar. Let op: de winnaar is degene van wie de som van de punten bij alle holes het laagste is.

Bijvoorbeeld:

punten = { "Calders": [2, 4, 3, 2, 3, 4, 4, 4, 3, 3], # Som is 32 "Hofkens": [3, 3, 2, 4, 2, 4, 3, 3, 4, 3], # Som is 31 "Laenens": [2, 3, 4, 2, 3, 4, 4, 3, 4, 3], # Som is 32 "Symens": [2, 3, 2, 4, 2, 2, 3, 2, 2, 4], # Som is 26 }

```
>>> print(bepaal_winnaar(punten))
```

Symens

```
1
```

Submit

Question 8: Element counter

✓ Perfect

Schrijf een (heel korte) functie `tel_elementen` die het aantal verschillende elementen uit een lijst teruggeeft. De body van je functie mag maar 1 regel zijn.

```
1  def tel_elementen(ele):
2      return len(set(ele))
```

Submit

Question 9: Priemgetallen en fibonaccigetallen (a)

✓ Perfect

Schrijf een functie `priemgetallen(n)` die een lijst van de eerste `n` priemgetallen teruggeeft. Het eerste priemgetal is 2. De code voor `is_priem` en `volgend_priemgetal` werden reeds toegevoegd zodat je die kan hergebruiken.

```
1  def priemgetallen(n):
2      result = []
3      current_priem = 2
4
5      while len(result) != n:
6          result.append(current_priem)
7          current_priem = volgend_priemgetal(current_priem)
8
9      return result
```

Submit

Question 10: Priemgetallen en fibonaccigetallen (b)

✓ Perfect

Schrijf een functie `fibonacci(n)` die een lijst van de eerste `n` fibonacci-getallen teruggeeft. Het n-de fibonacci-getal is de som van zijn twee voorgangers. Begin met de lijst `[1,1]` en breidt deze uit tot de gewenste lengte. Hergebruik je code van vorige sessie!

```
>>> fibonacci(6)
>>> [1,1,2,3,5,8]
```

```
def fibonacci(n):
    result = [1,1]

    for i in range(2, n):
        result.append(result[i-2] + result[i-1])
    return result
```

Submit

Question 11: Priemgetallen en fibonaccigetallen (c)

✓ Perfect

Schrijf een heel korte (max. 3 regels in de body van de functie) functie `priem_en_fib(n)` die een set van getallen teruggeeft die zowel priem als fibonacci zijn. `n` duidt op de eerste `n` priemgetallen en de eerste `n` fibonacci-getallen uit de vorige oefeningen. Maak gebruik van de correcte set operaties.

```
def priem_en_fib(n):
    return set(fibonacci(n)) & set(priemgetallen(n))
```

Submit

Question 12: Priemgetallen en fibonaccigetallen (d)

✓ Perfect

Schrijf een heel korte (max. 3 regels in de body van de functie) functie `priem_of_fib(n)` die een set van (unieke) getallen teruggeeft die priem of fibonacci zijn (of beide). `n` duidt op de eerste `n` priemgetallen en de eerste `n` fibonacci-getallen uit de vorige oefeningen.

```
def priem_of_fib(n):
    return set(fibonacci(n)) | set(priemgetallen(n))
```

Submit

Question 13: Priemgetallen en fibonaccigetallen (e)

✓ Perfect

Schrijf een heel korte (max. 3 regels in de body van de functie) functie `priem_xof_fib(n)` die een set van getallen teruggeeft die priem of fibonacci zijn (maar niet beide). `n` duidt op de eerste `n` priemgetallen en de eerste `n` fibonacci-getallen uit de vorige oefeningen.

```
1  def priem_xof_fib(n):
2      return set(fibonacci(n)) ^ set(priemgetallen(n))
```

Submit

Question 14: Taakbeheer Systeem

✓ Perfect

Hieronder vind je de code voor een taakbeheer systeem. Er zitten echter nog een paar bugs verstopt. Zoek en verbeter de 3 fouten met behulp van de debugger in PyCharm.

```
1  # Function to add a new task
2  def add_task(tasks, task):
3      tasks.append(task)
4
5  # Function to update a task's status
6  def update_task(tasks, task_name, status):
7      for task in tasks:
8          if task['name'] == task_name:
9              task['status'] = status
10
11 # Function to print all pending tasks
12 def print_pending_tasks(tasks):
13     for task in tasks:
14         if task['status'] == 'pending':
15             print(task['name'])
16
17 # Function to check if a task is complete
18 def is_task_complete(tasks, task_name):
19     for task in tasks:
20         if task['status'] == 'complete' and task['name'] == task_name:
21             return True
22
```

Submit