Introductie in Programmeren

Oefententamen 2

2018

Schrijf meteen jouw naam en studentnummer hieronder:

Je mag de vragen in Engels of Nederlands beantwoorden.

Alle code op dit tentamen is in de programmeertaal Python 3.

Dit is een "gesloten boek"-tentamen. Je mag voor het invullen je pen of potlood gebruiken, maar verder niets. Schrijf duidelijk en niet te groot.

Het laatste blaadje dient als kladpapier. Deze mag je eraf scheuren. Laat het weten als je meer kladpapier nodig hebt.

Je hoeft geen comments in je code te schrijven. Je mag alle aangeleverde functies uit de lectures of de opdrachten gebruiken in je antwoorden. Pseudocode kan ook punten opleveren, maar doorgaans minder dan "echte" code!

Schrijf jouw antwoorden op het tentamen.

Tijd: 2 uur

/ 23 punten

Concepten (2 punten)

0. (1 punt) Wat wordt er geprint na het uitvoeren van dit stukje code?

```
print(9 / 2)
```

1. (1 punt) Wat wordt er geprint na het uitvoeren van dit stukje code?

```
print('A' * 2)
```

Comprehensions (3 punten)

2. (1 punt) Wat wordt er geprint na het uitvoeren van dit stukje code?

```
words = ["dit", "is", "een", "lijst", "van", "woorden"]
print([x for x in words if len(x) > 3])
```

3. (1 punt) Het volgende stukje code produceert een lijst `L`. Met welke list comprehension bereiken we het zelfde resultaat?

```
L = []
for c in "foobarbaz":
    if c in "aeiou":
        L.append(c)
```

4. (1 punt) Schrijf een list comprehension die de lijst `[1,4,9,16,25,36]` produceert. Maak hierbij gebruik van `range()`.

Verzamelingen (4 punten)

5. (1 punt) Noem één verschil tussen een `list` en een `set`, dat niet het verschillende type haakjes is.

6. (1 punt) Noem één verschil tussen een `list` en een `tuple`, dat niet het verschillende type haakjes is.

7. (1 punt) Wat wordt er geprint na het uitvoeren van dit stukje code?

```
data = [1,2,3,1,2,1,2,3]
x = {}
for datum in data:
    if datum in x:
        x[datum] += 1
    else:
        x[datum] = 1
print(x)
```

8. (1 punt) Het volgende stukje code geeft een error. Hoe kan je ervoor zorgen dat in variabele S de door de gebruiker ingevoerde string komt te staan, met als eerste letter een `"C"`?

```
>>> S = input("What is your name?")
Jasper
>>> S[0] = "C"
TypeError: 'str' object does not support item assignment
```

Class (8 punten)

Om verbindingen tussen punten bij te houden hebben we onderstaande class ontworpen. Het implementatiewerk dat is voor jou! In ons probleem kan elk punt verbonden zijn met één ander punt. De vraag is dan, gegeven een punt A en B, zijn deze verbonden? Dat kan direct zijn, maar ook indirect via een punt C:

```
A -> C -> B
```

Of via meerdere punten:

```
A -> C -> D -> B
```

Mocht je nu ook Inleiding Logica volgen, dan herken je dit vast als een transitieve relatie!

```
class Connections:
    def __init__(self):
        self.connections = {}

    def add(self, a, b):
        self.connections[a] = b

    def __str__(self):
        out = ""
        for key in self.connections:
            out += f"{key}->{self.connections[key]} "
        return out

    def is_connected(self, a, b):
        # TODO

    def __add__(self, other):
        # TODO
```

Kijk, `__init__`, `add` en `__str__` zijn al voor je geïmplementeerd:

```
>>> cons = Connections()
>>> cons.add("a", "b")
>>> cons.add("b", "c")
>>> print(cons)
a->b b->c
```

Nu is het aan jou om `is_connected` te implementeren zodat ook dit voorbeeld werkt:

```
>>> cons = Connections()
>>> cons.add("a", "b")
>>> cons.add("b", "c")
>>> cons.add("c", "d")
>>> cons.is_connected("a", "d")
True
>>> cons.is_connected("a", "e")
False
```

8. (4 punten) Implementeer `is_connected` hieronder. De methode `is_connected` moet een `bool` returnen. Je `returnt` `True` als de twee punten verbonden zijn en `False` als deze niet verbonden zijn.

```
def is_connected(self, a, b):
```

Na even in de Python-documentatie te hebben gezocht, blijkt dat je de methode `__add__` kunt implementeren om de `+` operatie mogelijk te maken. Zo vertaalt `cons1 + cons2` naar `cons1.__add__(cons2)`
Aan jou de taak om `__add__` te implementeren zodat ook dit voorbeeld werkt:

```
>>> cons1 = Connections()
>>> cons1.add("a", "b")
>>> cons2 = Connections()
>>> cons2.add("b", "c")
>>> cons3 = cons1 + cons2
>>> print(cons3)
a->b b->c
>>> print(cons1)
a->b
>>> print(cons2)
b->c
```

9. (4 punten) Implementeer hieonder de methode `__add__`. Deze methode accepteert een andere `Connections` (`other`) als argument en `return`t een nieuwe `Connections`. De nieuwe `Connections` is de optelling van alle verbindingen. Je mag hierbij vanuit gaan dat er geen verbindingen worden overschreven. Ofwel als `A` verbonden is met `B`, dat `A` dan geen verbinding zal hebben in de andere `Connections`.

```
def __add__(self, other):
```

Debuggen (6 punten)

Elk uur werd er een meting gedaan van het C02 gehalte in de lucht in a1.16 op het Science Park. Deze metingen zijn opgeslagen in `C02.txt`. Dit bestand bevat twee kolomen aan data. De eerste kolom is het uur van de meting, en de tweede kolom het ppm (parts per million) C02 in de lucht. Na 10 metingen ziet `C02.txt` er zo uit:

```
0,512
1,640
2,593
3,580
4,581
5,613
6,840
7,889
8,863
9,891
```

Het volgende programma leest het bestand `C02.txt` in en stopt elke kolom in een lijst. Het werkt echter niet helemaal.

air_quality.py

```
1)  hours = []
2)  ppm = []
3)
4)  with open("C02.txt", "r") as f:
5)      for line in f:
6)          hour = int(line.split(",")[0])
7)          hours.append(hour)
8)
9)      for line in f:
10)         c02 = int(line.split(",")[1])
11)         ppm.append(c02)
12)
13) print(hours)
14) print(ppm)
```

Bij een test van de code krijgen we de volgende uitkomst

```
$ python air_quality.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[]
```

11. (2 punten) Leg uit wat er fout gaat en hoe je dit kan oplossen.

Het programma hieronder was bedoeld om alle elementen te vinden die in dezelfde plek in beide lijsten zitten. Het werkt echter niet helemaal

```
common.py
1) first = [0,2,4,6,8]
2) second = [0,1,2,3,4]
3)
4) for i in first:
5)     if first[i] == second[i]:
6)         print(first[i])
```

Bij een test van de code gebeurt het volgende:

```
$ python common.py
0
Traceback (most recent call last):
  File "common.py", line 5, in <module>
    if first[i] == second[i]:
IndexError: list index out of range
```

12. (2 punten) Leg uit wat er fout gaat en hoe je dit kan oplossen.

Een woord is een palindroom als je het woord kan omdraaien zonder dat het woord veranderd. Voorbeelden zijn `pap`, maar ook bijvoorbeeld `koortsmeetsysteemstrook`. Hieronder staat code om te kijken of een woord een palindroom is. Het werkt echter niet helemaal.

is_palindrome.py

```
1)  def is_palindrome(word):
2)      if len(word) <= 1:
3)          return True
4)
5)      if word[0] != word[-1]:
6)          return False
7)
8)      return is_palindrome(word)
9)
10) print(is_palindrome("koortsmeetsysteemstrook"))
```

Bij een test van de code gebeurt het volgende:

```
$ python is_palindrome.py
Traceback (most recent call last):
  File "is_palindrome.py", line 10, in <module>
    print(is_palindrome("koortsmeetsysteemstrook"))
  File "is_palindrome.py", line 8, in is_palindrome
    return is_palindrome(word)
  File "is_palindrome.py", line 8, in is_palindrome
    return is_palindrome(word)
  File "is_palindrome.py", line 8, in is_palindrome
    return is_palindrome(word)
  [Previous line repeated 994 more times]
  File "is_palindrome.py", line 2, in is_palindrome
    if len(word) <= 1:
RecursionError: maximum recursion depth exceeded in comparison
```

13. (2 punten) Leg uit wat er fout gaat en hoe je dit kan oplossen.

Kladpapier

Wij kijken niet naar wat je op dit vel schrijft!