MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Kursledare: Lars Arvestad
Examinator: Lars Arvestad

Tentamensskrivning i
DA2005 Programmeringsteknik och DA2004 … för matematiker
7.5 hp
2021-08-23

- Del A har flervalsfrågor där minst ett svarsalternativ är korrekt. Om man svarar fel eller inte har exakt rätt antal alternativ får man 0 poäng på frågan.
- Man måste bli godkänd på del A (4 poäng på 8 frågor) för att få göra del B.
- Del B består av ett antal frågor med varierande poäng (totalt 12) vilka ska lösas genom att man skriver Python-kod.
- Svaren till del B lämnas in i en `.py`-fil namngiven `anonymkod.py` där `anonymkod` är koden som man får i Ladok när man registrerar sig till tentan. Man **måste** även ha med sin anonyma kod i en kommentar i toppen av filen. Man ska **inte** skriva sitt riktiga namn någonstans i filen!

  Skriv också gärna anonymiseringskod i kommentar högst upp i filen.
- Var *noga* med att **namnge** *funktioner och klasser* rätt (på samma sätt som de är namngivna i uppgiften).
- Inga `import` får användas om de inte nämns eller finns med i uppgiften. Man får dock använda inbyggda funktioner som `len`, `range` och `map`.
- All kod avser **Python 3**, dvs *inte* t.ex. Python 2.7
- **Hjälpmedel:** Till del A får man ha ett A4 med så mycket information man vill. Du får skriva på båda sidorna. Del B är öppen bok då det är hemtenta och samma regler kring hjälpmedel gäller som för projekt och labbar.
- **Betygsgränser:** E: 10, D: 12, C: 14, B: 16, A: 18, av maximala 20.

Del A: flervalsfrågor (1p per fråga)

1. Vad är syftet med *moduler* i Python? Ett svar är rätt.

A. Det är ett sätt att definiera funktioner.
B. Det är ett sätt att organisera funktioner i logiska grupper.
C. Det är ett sätt att skriva kommentarer för funktioner.
D. Det är ett sätt att snabba upp beräkningar.
E. Det är ett sätt att strukturera sin felhantering.

2. Vad är värdet på `x` efter att man exekverat raderna till höger?

A. 0
B. 1
C. 4
D. 9
E. > 9

```
x=0
for y in range(3):
   for z in range(3):
      x = y + z
```

3. Vad innebär begreppet *typkonvertering* (även känt som *type casting*).

A. Man begränsar variabler till endast en typ. Till exempel kan man bestämma att `x` bara kan lagra heltalsvärden.
B. Man tilldelar en variabel ett värde av en ny typ. Till exempel kan man först göra `x = 1` och sedan `x = 7.3`.
C. Man tolkar om ett värde till en annan typ. Till exempel kan ett heltal göras om till ett flyttal.
D. Man gör en avrundning.
E. Man använder en if-sats till att avgöra om ett värde är giltigt eller inte.

4. Betrakta koden till höger, vad returneras från anropet `g(1)`?

A. `None`
B. 1
C. 2
D. 3
E. 6

```
a = None
def f(x):
    return 2*x
def g(x):
    a=f(x)
    return 3*a
```

5. Om `x = [{'a': 1, 'b': 2}, 3, [4, 5], (6, 7)]`, vad ska man skriva för att komma åt värdet 2?

A. `x[0][1]`
B. `x[1][2]`
C. `x['b'][0]`
D. `x['b'][1]`
E. `x[0]['b']`

6. Vad blir resultatet av `len(list(filter(lambda x: len(x)== 3, ['asp', 'björk', 'bok', 'gran', 'tall'])))`?

A. None
B. 2
C. 3
D. `['asp', 'bok']`
E. `[3, 5, 3, 4, 4]`

7. Vilka av nedanstående booleska uttryck är sanna om `x=True`, `y=True`, och `z=False`,

A. `x and y and z`
B. `not x or not y or not z`
C. `(x and y)or (x and z)or (y and z)`
D. `(x and not y)or (x and not z)or (y and not z)`
E. `not (x or y or z)`

8. Vad innebär begreppet *attribut* i objektorienterad programmering?

A. En variabel lagrad i ett objekt.
B. Namnet på en klass.
C. Det avser den klass man ärvt från, också kallat superklass.
D. Det är den metod som används för att initiera ett objekt.
E. Så kallas returvärdet från ett metodanrop.

Del B: kodfrågor (2p per fråga)

9. Skriv en funktion `f` för att skriva ut alla nycklar och längder på nycklarnas associerade värden i en uppslagstabell (eng: *dictionary*), en nyckel och dess värde per rad. Du ska anta att både nycklar och dess värden är strängar. Ex: Med uppslagstabellen

```
barr_fakta = {'gran': 'korta barr',
              'tall': 'långa barr'}
```

ska utdata bli

```
gran 10
tall 10
```

men din lösning ska förstås hantera alla uppslagstabeller på samma form.

10. Skriv funktionen `spell_check_helper` som tar två parametrar: en sträng `word` och en uppslagstabell `bigrams` som har strängar av längd två som nycklar och `True` som värden. Funktionen ska returnera `True` om `word` består av bokstavspar som återfinns i `bigrams` och `False` om minst ett bokstavspar saknas i `bigrams`.

Tester:

```
[In] : d = build_bigrams(['baka', 'kaka'])
[In] : d
[Out]: {'ba': True, 'ak': True, 'ka': True}
[In] : spell_check_helper('kaka', d)
[Out]: True
[In] : spell_check_helper('baka', d)
[Out]: True
[In] : spell_check_helper('baka kaka', d)
[Out]: False
[In] : spell_check_helper('vaka', d)
[Out]: False
[In] : spell_check_helper('bak', d)
[Out]: True
[In] : spell_check_helper('', d)
[Out]: False
[In] : spell_check_helper('b', d)
[Out]: False
```

Ovan är `build_bigrams` är en tänkt hjälpfunktion som returnerar en uppslagstabell med bokstavspar givet en lista med ord.

11. I den här uppgiften ska du skriva kod för att simulera ett sommarkryp! Skapa en klass `Bug` som har attribut för koordinater x och y, samt en räknare, `n_steps`, som håller reda på hur många steg krypet har tagit.

När man instantierar ett kryp (dvs ett objekt från `Bug`) ska dess position sättas till origo $(0, 0)$ och stegräknaren ska sättas till 0.

Det ska finnas tre metoder för klassen:

- `step`, som tar ett steg i en av riktningarna vänster, höger, upp eller ned, slumpmässigt valt. Krypen kan alltså bara gå i fyra riktningar, inte vilken riktning som helst. Steglängden är 0.1.
- `walk`, som tar antalet steg som parameter, och sedan utför det antalet steg. Metoden ska använda `step`-metoden, för slumpad vandring.
- `distance_from_origin`, som returnernar avståndet till origo som krypet är på, med hjälp av Pythagoras sats.

Du får använda modulerna `math` och `random`.

Exempel:

```
[In] : b = Bug()
[In] : b.walk(999)
[In] : b.distance_from_origin()
[Out]: 5.470831746635972    # A random value, of course
```

Tester:

```
[In] : b = Bug()
[In] : b.distance_from_origin()
[Out]: 0
[In] : b.step()
[In] : b.distance_from_origin() > 0
[Out]: True
[In] : b.n_steps
[Out]: 1
[In] : b.walk(999)
[In] : b.distance_from_origin() > 0
[Out]: True
[In] : b.n_steps
[Out]: 1000
```

12. Skriv klassen `Harkrank` som en *subklass* till `Bug`. Ändra `Bug`, om det behövs, så att den har ett attribut `step_size` som är 0.1 i basklassen, men sätts till 1.0 i konstruktorn för `Harkrank`, och används för att bestämma hur långa stegen är.

Du ska också ändra beteendet för `walk` i `Harkrank`: den tar aldrig fler än 50 steg åt gången, oavsett vad man ber den göra. Så för att ta 100 steg måste man anropa `walk` minst två gånger.

`Harkrank` ska vara mycket begränsad. Den ska ärva attribut och metoder från `Bug`.

Det går bra att svara på fråga 11 och 12 samtidigt. Dvs, du behöver inte ha två versioner av `Bug` i din inlämnade fil.

Tester:

```
[In] : h = Harkrank()
[In] : h.step_size
[Out]: 1
[In] : h.n_steps
[Out]: 0
[In] : h.walk(100)
[In] : h.n_steps
[Out]: 50
[In] : h.walk(999)
[In] : h.n_steps
[Out]: 100   # Det blev 50 steg till
```

13. Alla har väl spelat *Minesweeper*? Spelet förstörde kontorsproduktiviteten på 90-talet! Din uppgift är att skriva funktionen `minesweeper_map` som skapar en karta för Minesweeper. Funktionen ska ta två parametrar, `n` och `p`, och returnera en lista med strängar som ger en karta med storlek $n \times n$ för spelet. Listan ska alltså ha `n` element och bestå av strängar av längden `n`. Strängarna består av mellanslag (för ”tomt”) och bokstaven ”o”, som representerar en mina. Minorna ska placeras med sannolikheten `p`, som är mellan 0 och 1.

Du får använda modulen `random`.

Exempel:

```
[In] : m = minesweeper_map(5, 0.1)
[In] : m
[Out]: ['o    ', '     ', '     ', ' o o ', 'o    ']  # Olika varje gång!
[In] : for line in m: print(line)
[Out]:
o



 o o
o
```

Tester:

```
[In] : minesweeper_map(3, 1)
[Out]:['ooo', 'ooo', 'ooo']
[In] : minesweeper_map(3, 0)
[Out]:['   ', '   ', '   ']
[In] : minesweeper_map(0, 0)
[Out]:[]
```

14. I den här uppgiften ska du rätta fel. Funktionen `mine_counter`, nedan, tar en minesweeper-karta (t.ex., given av `minesweeper_map` i uppgift 13) och returnerar en lista av listor, representerande en matris, där element *i*, *j* räknar hur många minor som finns inom avstånd 1 från position *i*, *j* i kartan. Till exempel, om kartan är 3×3 och helt tom så blir resultatet en noll-matris:

```
[In] : c = mine_counter(['   ', '   ', '   ])
[In] : c
[Out]: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Om vi lägger en mina i mitten på kartan så blir resultatet istället en ett-matris, eftersom varje ruta är inom avstånd 1 från minan:

```
[In] : c = mine_counter(['   ', ' o ', '   ])
[In] : c
[Out]: [[1, 1, 1], [1, 1, 1], [1, 1, 1]]
```

Prova även med en eller flera minor i en 5×5-matris:

```
[In] : ex3 = ['     ', '     ', '  o  ', '     ', '     ']
[In] : res3 = mine_counter(ex3)
[In] : res3
[Out]: [[0, 0, 0, 0, 0], [0, 1, 1, 1, 0], [0, 1, 1, 1, 0], [0, 1, 1, 1, 0], [0, 0, 0,
     0, 0]]
[In] : ex4 = ['     ', '  o  ', ' ooo ', '  o  ', '     ']
[In] : res4 = mine_counter(ex4)
[In] : res4
[Out]: [[0, 1, 1, 1, 0], [1, 3, 4, 3, 1], [1, 4, 5, 4, 1], [1, 3, 4, 3, 1], [0, 1, 1,
     1, 0]]
```

Lägg märke till att det finns fem minor i `ex4` och mittpunkten är nära dem alla.

Men koden nedan har ett fel som visar sig på en del kartor! Ett särfall uppstår och programmet avbryts. Exempelkartorna ovan är ”snälla” och demonstrerar inte problemet.

Korrigera koden nedan så att man kan använda `mine_counter` på alla kartor!

```
def increment_neighborhood(res, i, j):
    for delta_i in [-1, 0, 1]:
        for delta_j in [-1, 0, 1]:
            res[i + delta_i][j + delta_j] += 1


def mine_counter(matrix):
    res = list()
    for line in matrix:
        res.append([0 for c in line])

    for i, line in enumerate(matrix):
        for j, c in enumerate(line):
            if c == 'o':
                increment_neighborhood(res, i, j)

    return res
```

Tips: Du kan kopiera över koden till en egen fil, men tyvärr följer inte indenteringen med. En separat kodfil läggs upp på kurswebben. Och här nedan är de snälla testfallen enkelt tillgängliga. Skapa några egna också, för hand eller med `minesweeper_map`, som triggar felet.

```
ex1 = ['   ', '   ', '   ']
ex2 = ['   ', ' o ', '   ']
ex3 = ['     ', '     ', '  o  ', '     ', '     ']
ex4 = ['     ', '  o  ', ' ooo ', '  o  ', '     ']
```