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

Tentamensskrivning i
DA2004 Programmeringsteknik för matematiker
7.5 hp
2022-04-21

- Tentan har flervalsfrågor där minst ett svarsalternativ är korrekt. Om man svarar fel eller inte har exakt antal rätta alternativ får man noll poäng på frågan.
- **Skriv tydligt.** Svårlästa svar riskerar 0 poäng.
- Skriv bara på en sida av varje papper!
- Man måste bli godkänd på del A (5 rätt på 10 frågor) för att del B ska rättas.
- **Hjälpmedel:** Ett A4 med så mycket information du vill. Du får skriva på båda sidorna.
- **Betygsgränser:** E: 10, D: 12, C: 14, B: 16, A: 18, av maximala 20.

Del A: flervalsfrågor

Var snäll samla svaren på del A på ett svarspapper. Varje fråga på del A är värd 1 poäng.

1. Betrakta kodsnutten till höger, vad blir värdet på `x` när man har kört koden?

A. 0
B. 6
C. 8
D. 10
E. Inget då programmet inte går att köra eftersom man måste ha en **`print`** när man fångat ett särfall med **`except`**.

```
s = "DA2004"
x = 0
for c in s:
    try:
        x += int(c)
    except:
        x += 1
```

2. Hur många kombinationer av tilldelningar av `True`/`False` för variablerna `x`, `y`, och `z` finns det som gör uttrycket till höger sant?

A. 1
B. 2
C. 3
D. 4
E. 5

```
(not x) or (y and z)
```

3. Vilka alternativ är korrekta för koden till höger?

A. Klassen `C` ärver från `A` och `B`
B. `x` är en klassvariabel
C. `x` är en instansvariabel
D. `f` är en metod som finns i `B` och `C`
E. Koden leder till ett felmeddelande då `B` och `C` saknar konstruktorer.

```
class A:
    def __init__(self):
        self.y = "42"

class B:
    def f(self):
        return 42

class C(A,B):
    x = 42
```

4. Vad gör `yield` i Python?

A. Det används för att skapa ett särfall.
B. Det används för att slå ihop två klasser till en.

C. Det används för att generera slumpmässiga tal.
D. Det används för att skapa en generator.
E. Det finns inget som heter `yield` i Python.

5. Hur många gånger anropas **`print`** när man kör koden till höger?

A. 0 gånger
B. 3 gånger
C. 6 gånger
D. 9 gånger
E. 12 gånger

```
i = 1
j = 5

while i < 4:
    while j < 8:
        print(i, ",", j)
        i = i + 1
        j = j + 1
```

6. Vad blir resultatet av `f('Lycka till!')` med koden till höger?

A. `'Lcatl!'`
B. `'yk il'`
C. `''`
D. `'L!'`
E. Inget då programmet kommer krascha då det saknas ett basfall.

```
def f(mylist):
    if len(mylist) <= 1:
        return mylist
    else:
        return mylist[0] + f(mylist[2:])
```

7. Om `x = ( 3, [4, 'e'], 'omtenta', { 'a': 1, 'e': 2 } )`, vilket eller vilka av följande alternativ kan man skriva för att komma åt ett `'e'`?

A. `x[1][1]`
B. `x[1][2]`
C. `x[2][3]`
D. `x[3]['e']`
E. `x[3][2]`

8. Vilket eller vilka påståenden gäller för Python?

A. Man får ha listor som *nycklar* i uppslagstabeller
B. Strängar är tupler av bokstäver
C. Typen **`int`** har ingen maxgräns för hur stora tal som kan hanteras
D. Funktioner i Python måste innehålla **`return`**
E. Pythonfunktionen **`open`** returnerar ett filnamn

9. Vad är värdet på `z` när man kört kodsnutten till höger?

A. 0
B. 5
C. 8
D. 10
E. 20

```
z = 0
for x in range(4):
    z += x
    for y in range(x):
        z += y
```

10. Vad skrivs ut när man kör koden till höger?

A. 13

B. 8

C. 3

D. 10

E. 5

```
a = 10

def f1(a,b=3):
    return a + b

def f2(var1, var2):
    return f1(var1 + var2,a)

var1 = 1
var2 = 2
print(f2(var1,var2))
```

Del B: koduppgifter

Var snäll använd ett papper till varje uppgift i del B.

11. Skriv en funktion `word_len(s)` som givet en sträng `s` skapar en uppslagstabell där nycklarna är orden i `s` och värdena är ordens längd. (1p)

Exempelanvändning:

```
[In: ] print(word_len('DA2004 Programmeringsteknik för matematiker'))
[Out:] {'DA2004': 6, 'Programmeringsteknik': 20, 'för': 3, 'matematiker': 11}
[In: ] print(word_len('Lycka till på tentan!'))
[Out:] {'Lycka': 5, 'till': 4, 'på': 2, 'tentan!': 7}
```

Tips: använd Python metoden `split()` som delar upp en sträng i en lista med ord.

12. Skriv en funktion `triangles(n)` som med hjälp av en listomfattning skapar alla triplar av positiva tal `(x,y,z)` med $x < y < z < n$ som uppfyller att $x^2 + y^2 = z^2$. (1p)

Exempelanvändning:

```
[In: ] print(triangles(20))
[Out:] [(3, 4, 5), (6, 8, 10), (5, 12, 13), (9, 12, 15), (8, 15, 17)]
```

13. Betrakta följande **`for`** loop,

```
for i in range(10):
    print(i)
```

Skriv ett ekvivalent program som skriver ut samma tal, men använder en **`while`** loop istället. (1p)

Obs: för poäng måste iterationen göras med hjälp av en **`while`** loop, så man får inte använda **`for`**, rekursion, listomfattningar, etc.

14. Skriv en högre ordningens funktion `take_while(p,mylist)` som tar in en funktion `p` som returnerar `True` eller `False`, samt en list `mylist`. Funktionen ska sedan returnera de första elementen i `mylist` som gör att testfunktionen `p` blir `True`. Så fort testfunktionen `p` blir `False` för något element i `mylist` ska alltså funktionen avslutas och inga fler värden returneras. (1p)

Exempelanvändning:

```
[In: ] print(take_while(lambda x: x > 3,[4, 5, 6, 2, 10]))
[Out:] [4, 5, 6]
[In: ] print(take_while(lambda x: x > 3,[1, 2, 3, 4, 5, 6, 2, 10])))
[Out:] []
```

15. Tribonacci talen är en talsekvens som påminner på Fibonacci talen, men istället för att börja med 0 och 1 och sen addera de två tidigare talen i sekvensen så börjar man med 0, 0 och 1, och sen adderar man de *tre* tidigare talen för att få nästa. Skriv en *rekursiv* funktion `tribonacci(n)` som returnerar det n:e Tribonacci talet räknat från `0`. (1p)

Exempelanvändning:

```
[In: ] print(tribonacci(0))
[Out:] 0
[In: ] print(tribonacci(1))
[Out:] 0
[In: ] print(tribonacci(2))
[Out:] 1
[In: ] print(tribonacci(3))
[Out:] 1
[In: ] print(tribonacci(4))
[Out:] 2
[In: ] print(tribonacci(5))
[Out:] 4
[In: ] print(tribonacci(6))
[Out:] 7
```

Obs: för poäng måste `tribonacci` vara rekursiv.

16. På ordinarie tenta 2022-03-14 skulle man skriva en klass för orcher (monster i Sagan om Ringen) med instansattributen `hp` och `strength` vilka representerar en orchs liv och styrka. Man skulle även ha en metod `beats` som givet en annan orch kollar vem som är starkast och returnerar `True` om orchen är lika stark eller starkare än den andra orchen och `False` annars. Nedan följer en möjlig lösning:

```
class Orc:

    def __init__(self,hp,strength):
        self.hp = hp
        self.strength = strength

    def beats(self,other):
        return self.strength >= other.strength
```

Uppgiften är nu att lägga till en metod `dec_hp` till `Orc` som tar in ett tal `n` och minskar orchens `hp` med `n`. (1p)

17. Lägg till en metod `fight` till `Orc` som givet en annan orch kollar vem som vinner en strid med hjälp av `beats` metoden och sedan använder `dec_hp` för att minska den som förlorats liv med vinnarens styrka. (1p)

18. Vi kallar en $n \times n$ heltalsmatris *komplett* om den innehåller siffrorna $1, \ldots, n^2$ så att varje siffra förekommer en gång och summan av varje rad, kolumn och diagonal är samma. Man kan bevisa att denna summa är samma för alla kompletta matriser av en specifik storlek, exempelvis för $n = 4$ är summan alltid 34. Ett exempel på en matris som inte är komplett än, men vilken kan bli det om man byter ut nollorna mot tal på rätt sätt är:

7	4	0	10
0	9	0	8
14	0	0	1
2	0	12	0

Ni ska nu implementera lite funktionalitet som kan användas som en del i en implementation av ett program för att komplettera 4×4 matriser. Matrisen ska representeras som en lista med fyra listor med fyra tal i. Exempelvis en tom matris är representerad som `[ [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0] ]` och matrisen ovan skrivs som `[ [7, 4, 0, 10], [0, 9, 0, 8], [14, 0, 0, 1], [2, 0, 12, 0] ]`.

(a) Skriv en funktion `matrix_to_string(m)` som givet en 4×4 matris `m` på formen ovan *returnerar* en sträng där `0` har bytts ut mot `'..'`, rutorna med värden ≥ 10 läggs till som de är, och alla andra värden läggs till

som siffran med ett mellanslag framför. Denna typ av utskrift görs för att formateringen ska bli bra då varje siffra kommer ta upp två tecken. Slutet på varje lista med fyra tal ska även markeras med ny rad, dvs `'\n'`. (1p)

Obs: man kan här anta att `m` är på korrekt form och bara innehåller tal mellan `0` och `16`.

Exempelanvändning:

```
[In: ] matrix = [ [7, 4, 0, 10], [0, 9, 0, 8], [14, 0, 0, 1], [2, 0, 12, 0] ]
[In: ] print(matrix_to_string(matrix))
[Out:]  7 4..10
       .. 9.. 8
       14.... 1
        2..12..
```

(b) Skriv en funktion `is_possible_move(m,x,y)` som returnerar `True` om positionen `(x,y)` i matrisen `m` är tom (dvs `0`) och `False` annars. Funktionen ska även testa så att `x` och `y` ligger mellan `0` och `3`, om de inte gör det ska ett `ValueError` lyftas. (1p)

Obs: funktionen ska bara kolla så att positionen i matrisen är tom och att `x` och `y` har giltiga värden. Den ska alltså *inte* göra något mer avancerat som att kolla så att talet inte redan ligger på raden, kolumnen eller diagonalen.

(c) Nedan följer en väldigt enkel funktion för att stoppa in ett nytt tal i matris. Ett antagande som görs i koden är att `(x,y)` är en tom ruta i matrisen, detta är dock inte testat eller dokumenterat på något sätt. Lägg till en assertion som med hjälp av `is_possible_move` försäkrar oss om att funktionen används på rätt sätt. (1p).

```
def make_move(m,x,y,v):
    m[x][y] = v
    return m
```

Obs: lösningarna på delproblemen på uppgift 18 bör skrivas på samma svarspapper. Delproblemen kan betraktas som oberoende av varandra och man kan få poäng på någon av dem även om man inte löst de andra.