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

Tentamensskrivning i
DA2004 Programmeringsteknik för matematiker
7.5 hp
2023-01-12

- **Skriv tydligt.** Svårlästa svar riskerar 0 poäng.
- Skriv bara på en sida av varje papper!
- Motivera alla svar (om inte annat anges)!
- 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.

1. Hur får man antalet nyckel-värde-par i en uppslagstabell `mydict`?

A. `size(mydict)`
B. `length(mydict)`
C. `len(mydict)`
D. `mydict.size()`
E. `mydict[length]`

2. Vad blir resultatet av `fun("DA2004",3)` givet koden till höger?

A. `DDDAA2004`
B. `4002AADDD`
C. `DA2000444`
D. `4440002AS`
E. `D3A221004`

```
def fun(s,n):
    if n == 0:
        return s
    else:
        return (s[0] * n + fun(s[1:],n-1))
```

3. Vilka av följande ord har med filhantering i Python att göra?

A. `open`
B. `input`
C. `close`
D. `raise`
E. `for`

4. Hur många kombinationer av tilldelningar av `True`/`False` för variablerna `x`, `y`, och `z` finns det som gör uttrycket `x and y and (not y or not z)` sant?

A. 0
B. 1
C. 2
D. 3
E. 4

5. Vad blir värdet på `s` efter att man kört koden till höger?

A. `hhheej`

B. `heejjj`

C. `jjjeeehhh`

D. `hhheeejjj`

E. Ingenting då man inte kan ha en **`while`** loop i en **`for`** loop.

```
s = ""
for c in "hej":
    i = 0
    while i < 3:
        s += i * c
        i += 1
```

6. Vilka slutsatser kan vi dra baserat på koden till höger?

A. `A` ärver från `B`

B. `B` ärver från `A`

C. `t` är ett klassattribut i `A`

D. `t` är ett instansattribut i `A`

E. `p` är ett attribut i `B`

```
class A(B):
    def __init__(self, r):
        self.t = r
        self.t += B.p
```

7. Vad ger `[(x,y) for x in range(3) for y in range(x)]` för resultat?

A. `[(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)]`

B. `[(1, 0), (2, 0), (2, 1), (3, 0), (3, 1), (3, 2)]`

C. `[0, 0, 2]`

D. `[(1, 0), (2, 0), (2, 1)]`

E. `[(0, 1), (0, 2), (1, 2)]`

8. Vilka av följande anrop ger `42` givet `d = {"hej": [ 9, {10 : 42}, 1, 42 ], 5 : [1,[42]]}` ?

A. `d["hej"][1][10]`

B. `d["hej"][3]`

C. `d["hej"][10]`

D. `d[5][1][0]`

E. `d[5][1]`

9. Vilka påståenden är sanna för Python 3?

A. Listor är muterbara.

B. Listor kan vara nycklar i uppslagstabeller.

C. Strängar är muterbara.

D. Strängar kan vara nycklar i uppslagstabeller.

E. Uppslagstabeller kan vara värden i uppslagstabeller.

10. Vad gäller för kodsnutten nedan till höger?

A. Ingenting händer då vi direkt fastnar i en oändlig loop.

B. När användaren skriver in `42` avslutas loopen och `Congratuations!` skrivs ut.

C. Programmet accepteras inte av Python då det inte går att skriva **`while`** `True` på detta sätt utan man måste ha ett booleskt test i en **`while`**.

D. Varje gång användaren skriver in `42` skrivs `Congratulations!` ut.

E. Det enda sättet att få loopen att avslutas är att skriva in något som inte är ett tal då **`int`**`(x)` anropet då leder till ett särfall.

```
while True:
    x = input("Write a number: ")

    if int(x) == 42:
        print("Congratulations!")
```

Del B: kodfrågor

Var snäll använd ett papper (eller fler) till varje fråga i del B.

11. På den här uppgiften ska man skriva ut en tabell med värden i procent, detta delas upp i tre deluppgifter. All kod för deluppgifterna kan skrivas på samma papper.

A. Skriv en funktion `float_to_percent(x)` som tar in ett flyttal `x` och returnerar en sträng med vad det motsvarar i procent, utan decimaler. (1p)

Exempelanvändning:

```
[In: ] float_to_percent(0.01)
[Out:] 1%
[In: ] float_to_percent(1.2)
[Out:] 120%
[In: ] float_to_percent(0.003)
[Out:] 0%
```

B. Skriv en funktion `dict_to_averages(d)` som tar in en uppslagstabell `d` där värdena är tal och returnerar en uppslagstabell med samma nycklar men där värdena nu är andelen som varje värde motsvarade av totalen av värdena. (1p)

Exempelanvändning:

```
[In: ] cb = {"Butter" : 150 , "Sugar" : 50 , "Oats" : 100 , "Cacao" : 15}
[In: ] print(dict_to_averages(cb))
[Out:] {'Butter': 0.47619047619047616, 'Sugar': 0.15873015873015872,
        'Oats': 0.31746031746031744, 'Cacao': 0.047619047619047616}
```

C. Skriv en funktion `dict_to_table` som tar in en uppslagstabell där värdena är tal och skriver ut en tabell med procent för varje värde. (1p)

Tips: använd `float_to_percent` och `dict_to_averages` ovan. Det är OK att använda dessa även om man inte löst uppgifterna.

Exempelanvändning:

```
[In: ] cb = {"Butter" : 150 , "Sugar" : 50 , "Oats" : 100 , "Cacao" : 15}
[In: ] dict_to_table(cb)
[Out:] Butter: 47%
       Sugar: 15%
       Oats: 31%
       Cacao: 4%
```

12. Skriv en högre ordningens funktion `zip_with(f,list1,list2)` som tar in en funktion `f` som tar två argument samt två listor av samma längd och producerar en lista där varje värde är resultatet av att anropa `f` med element för element i `list1` och `list2`. (1p)

Lägg även in en **`assert`** i koden som testar så att listorna verkligen har samma längd när `zip_with` anropas. (1p)

Exempelanvändning:

```
[In: ] print(zip_with(lambda x, y: x * y,[1,2,3],[4,5,6]))
[Out:] [4, 10, 18]
[In: ] print(zip_with(lambda x, y: x + y,["lycka","på"],["till","tentan"]))
[Out:] ['lyckatill', 'påtentan']
[In: ] print(zip_with(lambda x, y: x * y,[1,2,3],[4,5]))
[Out:] AssertionError: Input lists must have same length
```

13. Skriv en funktion `read_numbers` som läser in tal från användaren ända tills hen skriver in `+` eller `*`. Sen ska programmet antingen skriva ut summan eller produkten av de inlästa talen. Lägg även till lämplig felhantering med hjälp av **`try-except`** som gör att en körning inte kraschar om man skriver in något annat än ett tal, `+` eller `*`. (2p)

Tips: på den här uppgiften får man använda `prod` funktionen från `math` biblioteket vilken beräknar produkten av en lista av tal.

Exempelanvändning där det användaren skiver kommer efter :

```
[In: ] read_numbers()
[Out:] Skriv ett tal, + eller *: 3
       Skriv ett tal, + eller *: 2
       Skriv ett tal, + eller *: hej
       Felaktig indata, försök igen!
       Skriv ett tal, + eller *: 5
       Skriv ett tal, + eller *: +
       Summan är 10
[In: ] read_numbers()
[Out:] Skriv ett tal, + eller *: 4
       Skriv ett tal, + eller *: 2
       Skriv ett tal, + eller *: 3
       Skriv ett tal, + eller *: *
       Produkten är 24
```

14. På föreläsningarna hade vi som löpande exempel på objektorientering ett rollspel med klasser som `Hero` och `Monster`. Något vi inte gjorde var att skapa en klass för `Items` med exempel som `Sword`, `Shield`, `Potion`, etc. Detta ska ni nu göra och all kod för deluppgifterna på den här uppgiften kan skrivas på samma papper.

A. `Item`: huvudklass för olika saker (eng. *items*) i spelet. Den ska ha en konstruktor som tar in ett argument `lvl` och sätter instansattributen `required_level` vilken motsvarar den level som en karaktär måste vara för att kunna använda saken. (1p)

Exempelanvändning:

```
[In: ] generic_item = Item(10)
[In: ] print(generic_item.required_level)
[Out:] 10
```

B. Lägg till subklassen `Sword` som ärver från `Item` och har instansattributen `name` och `damage` vilka representerar vad det är för svärd och hur mycket skada det kan göra. Konstruktorn ska anropa konstruktorn i `Item` så att man även kan sätta den level som krävs för att använda svärdet. (1p)

Exempelanvändning:

```
[In: ] mydagger = Sword(5,"dagger",20)
[In: ] print(mydagger.required_level)
[Out:] 5
[In: ] print(mydagger.name)
[Out:] dagger
[In: ] print(mydagger.damage)
[Out:] 20
```

C. Lägg till en `__str__` metod i `Sword` som fungerar enligt nedan. (1p)

Exempelanvändning (med `mydagger` som ovan):

```
[In: ] print(mydagger)
[Out:] A dagger is a sword which requires level 5 and causes damage 20
[In: ] mylongsword = Sword(12,"long sword",100)
[In: ] print(mylongsword)
[Out:] A long sword is a sword which requires level 12 and causes damage 100
```