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

Tentamen i
DA2005 Programmeringsteknik *och*
DA2004 Programmeringsteknik för matematiker
7.5 hp
2024-10-03

- 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.
- Man måste bli godkänd på del A (4 rätt på 8 frågor) för att del B ska rättas.
- Del B består av frågor med varierande poäng (totalt 12 poäng).
- Inga `import` (Pythons standardbibliotek eller externa bibliotek) får användas om de inte nämns eller finns med i uppgiften. Man får använda inbyggda funktioner som `len`, `range` och `map`.
- All kod avser **Python 3**, dvs *inte* t.ex. Python 2.7
- **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. Vad skrivs ut om vi kör koden till höger?

```
xs = [1, 2, 3]
ys = [4, 5]

print(len([ x + y for x in xs for y in ys ]))
```

A. 0
B. 5
C. 6
D. 15
E. Inget av alternativen A.-D. då ett särfall lyfts.

2. Vad är värdet på `zs` efter att vi kört koden till höger?

```
xs = [1, 5, 4]
ys = [0, 5, 6]
zs = []

for i in range(3):
    x = xs[i]
    y = ys[i]
    if x >= y:
        zs.append(x)
    else:
        zs.append(y)
```

A. `[0, 5, 4]`
B. `[1, 5, 4]`
C. `[0, 5, 6]`
D. `[1, 5, 6]`
E. Inget av alternativen A.-D. då ett särfall lyfts.

3. Vilket alternativ beskriver bäst följande rad kod?

```
a = lambda f, x: f(x) * f(x)
```

A. Det är ett sätt att iterera.
B. Det är ett sätt att uttrycka rekursion.
C. Det är ett sätt att definiera en funktion.
D. Det är ett sätt att uttrycka beräkningar med vektorer.
E. Det är ett sätt att markera var huvudprogrammet börjar.

4. Vilken eller vilka variabeltilldelningar gör att uttrycket `not x and (x or not y)` evaluerar till `True`?

A. `x = False, y = False`
B. `x = False, y = True`
C. `x = True, y = False`
D. `x = True, y = True`
E. Ingen, det evaluerar alltid till `False`.

5. Vad skrivs ut av koden till höger?

A. 1
B. 3
C. 4
D. 12
E. 13

```
x = 1
for a in range(4):
    for b in range(3):
        x = x + 1
print(x)
```

6. Hur ska man skriva för att komma åt strängen `secret` i `d`?

A. `d[1]['pw'][2,0]`
B. `d[2]['pw'][3,1]`
C. `d['pw'][2][0]`
D. `d['pw'][2,0]`
E. `d['pw'][3][1]`
F. Går ej.

```
d = {"hej": 2, "pw": [8,93,["secret",1]]}
```

7. Vad blir det för utskrift av koden till höger?

A. 6, 1, -15
B. 15, 10, -15
C. 5, 10, -15
D. 15, 1, -15
E. Ingenting, det blir ett särfall.

```
x=1
def f(x, y):
    y = 5
    return x + y
x=10
y=-15
print(f(0,0), x, y)
```

8. Vad blir det för utskrift av koden till höger?

A. `[(0, 3), (1, 2), (2, 1)]`
B. `[(0, 3), (1, 2)]`
C. `[(0, 3)]`
D. `[(3, 0), (2, 1), (1, 2)]`
E. `[(3, 0), (2, 1)]`
F. `[(3, 0)]`

```
i, j, l = 3, 0, []
while j < i:
    l.append((i,j))
    i -= 1
    j += 1
print(l)
```

Del B: kodfrågor

Var snäll använd ett papper till varje fråga i del B. Delfrågor, som 10A och 10B, får gärna lösas på samma papper.

9. Skriv funktionen `multiplikationstabell(n)` som skriver ut produkterna av två tal *upp till och med* talet n. Utskriften ska göras i matrisform i stil med exemplet nedan. Du behöver inte bry dig om raka kolumner och liknande finesser. För ett tal i, där $1 \leq i \leq n$, ska en rad med produkter $i \times j$ skrivas ut, där $1 \leq j \leq n$. (2p)

Tips: Du kan undvika radbrytningar efter print-satser med hjälp av en extra parameter. Uttrycket `print(x, end='___')` innebär att variabeln x skrivs ut, men istället för att som vanligt avsluta med en radbrytning så bestämmer `end` vad som skrivs ut sist. I detta exempel används tre understreck, för synlighetens skull, istället för en radbrytning.

Exempel:

```
[In: ] multiplikationstabell(2)
[Out:]
1   2
2   4
[In: ] multiplikationstabell(3)
[Out:]
1   2   3
2   4   6
3   6   9
[In: ] multiplikationstabell(4)
[Out:]
1   2   3   4
2   4   6   8
3   6   9   12
4   8   12   16
```

Observera att, för multiplikationstabellen upp till $n = 2$, vi har följande produkter: $1 \times 1 = 1$, $1 \times 2 = 2$, radbrytning, sedan $2 \times 1 = 2$, och $2 \times 2 = 4$.

10.

A. Skriv funktionen `standardise_word(word)` som givet en sträng som innehåller ett ord returnerar en en sträng där endast bokstäver är kvar och versaler har blivit gemena. Du behöver inte verifiera att det faktiskt finns ett ord i strängen. Det räcker med att plocka bort tecken som inte är bokstäver och ändra versaler (stora bokstäver) till gemena (små). Tips: använd `lower()`. (2p)

Exempel:

```
[In: ] standardise_word('')
[Out:] ''
[In: ] standardise_word('ko')
[Out:] 'ko'
[In: ] standardise_word('Hej!')
[Out:] 'hej'
[In: ] standardise_word('  x  ')
[Out:] 'x'
[In: ] standardise_word(' Binde-Streck.')
[Out:] 'bindestreck'
[In: ] 'Carl XVI Gustaf'.lower()
[Out:] 'carl xvi gustaf'
```

B. I delfråga A skulle `standardise_word` returnera tomma strängen om den fick tomma strängen som indata. Nu ska du istället betrakta det som ett fel. Det ska också vara ett fel om den standardiserade strängen blir tom.

Visa med kod hur man aktiverar felet `ValueError` (inbyggt i Python) om det standardiserade ordet är tomt. (1p)

Exempel:

```
[In: ] standardise_word('')
[Out:] ValueError: No word in input string
[In: ] standardise_word('  !  ')
[Out:] ValueError: No word in input string
[In: ] standardise_word('...')
[Out:] ValueError: No word in input string
[In: ] standardise_word('Lycka!')
[Out:] 'lycka'
```

Obs: det uppstår normalt lite mer text vid ett ValueError, här visas endast den sista raden på felutskriften.

11. Skriv funktionen `read_text(filename)` som läser en textfil given av `filename` och returnerar en lista av Word-objekt, givna av klassen nedan, som återger textfilens innehåll. Varje Word-objekt ska innehålla ett ord, standardiserat med funktionen från fråga 10 och radnummer i textfilen. (2p)

```
class Word:
    def __init__(self, word, line_number):
        self.word = standardise_word(word) # Den funktion som avses i tidigare fråga!
        self.line = line_number

    def __repr__(self):
        return f'Word({self.word}, {self.line})'
```

Exempel:

```
[In: ] read_text('robert_broberg.txt')
[Out:] [Word(bättre, 0), Word(vara, 0), Word(ute, 0), Word(på, 0), Word(hal, 0), Word(is,
    0), Word(och, 1), Word(ha, 1), Word(det, 1), Word(glatt, 1), Word(än, 2), Word(att,
    2), Word(gå, 2), Word(i, 2), Word(lera, 2), Word(och, 2), Word(sörja, 2)]
```

I detta exempel innehåller filen `robert_broberg.txt` en fras skriven på tre rader:

> Bättre vara ute på hal is
> och ha det glatt
> än att gå i lera och sörja!

12. Skriv funktionen `most_common_words(wordlist)` som tar en lista med Word-objekt och returnerar en lista med de ord (som vanliga strängar) som är vanligast. Det kan finnas flera ord som är vanligast, i den mening att de har samma antal förekomster. (2p)

Exempel: Antag att filen `gertrude_stein.txt` innehåller den berömda frasen ”A rose is a rose is a rose.” Då ska funktionen identifiera att ”a” och ”rose” förekommer tre gånger vardera, vilket är högst ordfrekvens i filen.

```
[In: ] gs = read_text('gertrude_stein.txt')
[In: ] most_common_words(gs)
[Out:] ['a', 'rose']
```

13. Utöka klassen Word (ovan) med metoden `letters()` som returnerar en lista med de bokstäver som används i ordet, utan upprepningar. Ordet ”mamma” innehåller ”m” och ”a” och ska därför returnera en lista med exakt de två bokstäverna. Ordningen på bokstäverna spelar inte någon roll. (1p)

Exempel:

```
[In: ] Word('a', 0).letters()
[Out:] 'a'
[In: ] x = Word('aaaa', 0)
[In: ] x.letters()
[Out:] 'a'
[In: ] Word('abba', 0).letters()
[Out:] 'ab'
[In: ] Word('mamma', 0).letters()
[Out:] 'ma'
```

14. Visa hur man löser följande uppgifter med hjälp av *högre ordningens funktioner*.

A. Givet en sträng vill vi ha en *lista* med de bokstavskoder (heltal) som representerar bokstäverna. Man får bokstavskoder med funktionen **`ord`** som tar en bokstav som argument. Till exempel har uttrycket **`ord`**`('A')` resultatet 65 och **`ord`**`('B')` resultatet 66. Men **`ord`** fungerar inte på längre strängar. Så hur göra jag för att omvandla ”Lavender Haze” till en lista med bokstavskoder? (Svaret blir `[76, 97, 118, 101, 110, 100, 101, 114, 32, 72, 97, 122, 101]`, där 32 representerar mellanslag.) (1p)

B. Givet en lista med bokstavskoder i variabeln *taytay*, hur kan jag få fram de bokstavskoder i *taytay* som motsvarar versaler? Man kan använda att versalerna för det engelska alfabetet har bokstavskoder i en serie, och vi struntar i de icke-engelska bokstäverna i denna uppgift. (1p)