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

Tentamen i
DA2005 Programmeringsteknik *och*
DA2004 Programmeringsteknik för matematiker
7.5 hp
2025-04-25

- 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. Vilka påståenden är sanna för Python 3?

A. Listor är muterbara.
B. Listor kan vara nycklar i uppslagstabeller.
C. Listor kan vara värden i uppslagstabeller.
D. Strängar kan vara nycklar i uppslagstabeller.
E. Strängar kan vara värden i uppslagstabeller.

2. Vad blir resultatet av anropet `g(2,1)` om funktionerna är definierade som till höger?

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

```
def f(a, b):
    return a * g(b)

def g(a, b=0):
    if b == 0:
        return a
    else:
        return f(a,b) * b
```

3. Vad är resultatet av uttrycket `int(float("0.1"))`?

A. Ett särfall då man inte kan konvertera strängar till flyttal.
B. Flyttalet `0.0`
C. Flyttalet `0.1`
D. Heltalet `0`
E. Heltalet `1`

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

A. `A` är en subklass till `B`.
B. `B` är en subklass till `A`.
C. `t` är ett instansattribut i klassen `A`.
D. Konstruktorn i `B` tar inga argument utöver `self`.
E. Konstruktorn kastar ett särfall då parametern `r` måste ha samma namn som `t`.

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

5. Vad innehåller `d` efter att man kört koden nedan?

```
s = ["Good","luck","on","the","exam"]
d = {}

for w in s:
    d[w] = len(w)
```

A. `{ 2 : ["on"], 3 : ["the"], 4 : ["Good", "luck", "exam"] }`
B. `{ 2 : "on", 3 : "the", 4 : "exam"}`
C. `{ 2 : "on", 3 : "the", 4 : "Good", 4 : "luck", 4 : "exam"}`
D. `{ "Good" : 4, "luck" : 4, "on" : 2, "the" : 3, "exam" : 4 }`
E. Inget då man får ett särfall då strängar inte kan vara nycklar i uppslagstabeller.

6. Vilka av följande kodsnuttar ger listan `[ 0, 2, 4, 6, 8 ]`?

A. `list(range(0,10,2))`
B. `[ x for x in range(10) if x % 2 == 0]`
C. `[ x for x in range(10) if x % 2 == 1]`
D. `list(map(lambda x: x % 2 == 0,range(10)))`
E. `list(filter(lambda x: x % 2 == 0,range(10)))`

7. Vad innehåller `out` efter att man kört koden till höger?

A. `"2D25"`
B. `225`
C. `"225"`
D. `9`
E. `"AA2"`

```
out = ""
s = "DA2025"

for x in s:
    try:
        out += s[int(x)]
    except:
        pass
```

8. Vilka av följande alternativ är typer i Python 3?

A. `char`
B. `str`
C. `module`
D. `dict`
E. `importer`

Del B: kodfrågor

Var snäll använd ett papper till varje fråga i del B. Delfrågor får gärna lösas på samma papper.

9. Betrakta **`for`**-looparna nedan:

```
for x in range(5):
    for y in range(5):
        print(str(x),str(y),str(x+y))
```

Skriv om dem med hjälp av **`while`**-loopar istället. För poäng måste koden endast använda sig av **`while`**-loopar på lämpligt sätt och utdata ska vara samma som om man kör koden ovan. (1p)

10. Skriv en funktion `conjunctify(lst)` som tar in en lista `lst` med strängar och returnerar en sträng med komma mellan alla ord utom de två sista där `and` ska skrivas in mellan istället. (2p)

Exempelanvändning:

```
[In: ] print(conjunctify(["Flour","milk","butter","eggs","salt"]))
[Out:] Flour, milk, butter, eggs and salt
[In: ] print(conjunctify(["Flour","milk"]))
[Out:] Flour and milk
[In: ] print(conjunctify(["Flour"]))
[Out:] Flour
```

11. Skriv ett program som frågar användaren `What is the answer to the ultimate question of life, the universe and everything?` och förväntar sig `42`. Om man skriver in ett annat tal än `42` ska programmet skriva ut `Wrong` och man får frågan igen. Skriver man `42` ska programmet skriva ut `Congratulations!` och sluta fråga. För full poäng måste man använda sig av **`try`** och **`except`** så att programmet inte kraschar om användaren skriver in något annat än tal. (2p)

Exempelkörning:

```
What is the answer to the ultimate question of life, the universe and everything? Food
You must input a number
What is the answer to the ultimate question of life, the universe and everything? 10
Wrong
What is the answer to the ultimate question of life, the universe and everything? 5
Wrong
What is the answer to the ultimate question of life, the universe and everything? 42
Congratulations!
```

12. Skriv ett program `sum_file(f)` som givet ett filnamn `f` till en fil innehållande tal utspridda över flera rader med mellanrum mellan skriver ut summan av talen. (2p)

Lägg till felhantering med hjälp av **`try-except`** så att programmet inte kraschar om man försöker öppna en fil som inte finns. (1p)

Exempelanvändning:

Givet filen `tal.txt` innehållande:

```
1 2
3
4 5 6 7
8 9
```

så ska följande hända:

```
[In: ] print(sum_file("tal.txt"))
[Out:] 45
```

Om man istället försöker öppna `oj.txt` som inte finns så ska följande hända:

```
[In: ] print(sum_file("oj.txt"))
[Out:] Error: the file does not exist
```

13. På ordinarie tenta skulle ni skriva kod för att representera låtar och album med hjälp av klasser. En exempellösning kunde se ut på följande sätt:

```
class Song:

    def __init__(self,a,n,l):
        self.artist = a
        self.name = n
        self.length = l

    def __str__(self):
        return self.artist + " - " + self.name

class Album:

    def __init__(self,a,n,ss):

        for s in ss:
            if s.artist != a:
```

```
                raise ValueError("All songs must have the same artist as the album")

        self.artist = a
        self.name = n
        self.songs = ss
```

Den här uppgiften går ut på att utöka funktionaliteten på två sätt.

A. Lägg till en `__ge__` metod till `Song` som låter en jämföra låtar med avseende på låtlängd. (2p)

Exempelanvändning:

```
[In: ] s1 = Song("Bruce Springsteen","The Ties That Bind",(3,33))
[In: ] s2 = Song("Bruce Springsteen","Hungry Heart",(3,19))
[In: ] s3 = Song("Bruce Springsteen","The River",(4,59))
[In: ] print(s1 >= s2)
[Out:] True
[In: ] print(s2 >= s3)
[Out:] False
```

B. Lägg till en metod `length` till `Album` som beräknar längden på albumet genom att summera längden på låtarna. Längden ska returneras som ett par `(m,s)` där `m` och `s` är albumets längd i minuter och sekunder. Kom ihåg att en minut har 60 sekunder. (2p)

Exempelanvändning:

```
[In: ] a = Album("Bruce Springsteen","The River",[s1,s2,s3])
[In: ] print(a.length())
[Out:] (11,51)
```

MATEMATISKA INSTITUTIONEN	Exam in
STOCKHOLMS UNIVERSITET	DA2005 Programming Techniques *and*
Avd. Beräkningsmatematik	DA2004 Programming Techniques for Mathematicians
Teacher: Anders Mörtberg	7.5 hp
Examiner: Lars Arvestad	2025-04-25

- The exam contains multiple-choice questions, where at least one option is correct. If you answer incorrectly or do not select all of the correct options, you will receive 0 points on the question.
- You must pass part A (4 correct answers out of 8 questions) for part B to be graded.
- Part B consists of questions with varying points (a total of 12 points).
- No import (Python's standard library or external libraries) is allowed unless mentioned or included in the task. You may use built-in functions like len, range, and map.
- All code is for **Python 3**, i.e., *not* e.g. Python 2.7
- **Help:** An A4 sheet with as much information as you like. You may write on both sides.
- **Grading:** E: 10, D: 12, C: 14, B: 16, A: 18, out of a maximum of 20.

Part A: Multiple choice questions

Please collect the answers to part A on one answer sheet.

1. Which of the following statements are true for Python 3?

A. Lists are mutable.
B. Lists can be keys in dictionaries.
C. Lists can be values in dictionaries.
D. Strings can be keys in dictionaries.
E. Strings can be values in dictionaries.

2. What is the result of calling `g(2,1)` if the functions are defined as shown to the right?

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

```
def f(a, b):
    return a * g(b)

def g(a, b=0):
    if b == 0:
        return a
    else:
        return f(a,b) * b
```

3. What is the result of the expression `int(float("0.1"))`?

A. An exception as strings cannot be converted to floating-point numbers.
B. The floating-point number `0.0`
C. The floating-point number `0.1`
D. The integer `0`
E. The integer `1`

4. What conclusions can we draw based on the code to the right?

A. `A` is a subclass of `B`.
B. `B` is a subclass of `A`.
C. `t` is an instance attribute in class `A`.
D. The constructor in `B` takes no arguments other than `self`.
E. The constructor throws in an exception as the parameter `r` must have the same name as `t`.

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

5. What does `d` contain after running the code below?

```
s = ["Good","luck","on","the","exam"]
d = {}

for w in s:
    d[w] = len(w)
```

A. `{ 2 : ["on"], 3 : ["the"], 4 : ["Good", "luck", "exam"] }`
B. `{ 2 : "on", 3 : "the", 4 : "exam"}`
C. `{ 2 : "on", 3 : "the", 4 : "Good", 4 : "luck", 4 : "exam"}`
D. `{ "Good" : 4, "luck" : 4, "on" : 2, "the" : 3, "exam" : 4 }`
E. Nothing, as it results in an exception as strings cannot be keys in dictionaries.

6. Which of the following code snippets result in the list `[ 0, 2, 4, 6, 8 ]`?

A. `list(range(0,10,2))`
B. `[ x for x in range(10) if x % 2 == 0]`
C. `[ x for x in range(10) if x % 2 == 1]`
D. `list(map(lambda x: x % 2 == 0,range(10)))`
E. `list(filter(lambda x: x % 2 == 0,range(10)))`

7. What does `out` contain after running the code to the right?

A. `"2D25"`
B. `225`
C. `"225"`
D. `9`
E. `"AA2"`

```
out = ""
s = "DA2025"

for x in s:
    try:
        out += s[int(x)]
    except:
        pass
```

8. Which of the following are types in Python 3?

A. `char`
B. `str`
C. `module`
D. `dict`
E. `importer`

Part B: Coding questions

Please use one piece of paper for each question in Part B. Subquestions can be solved on the same paper.

9. Consider the **`for`** loops below:

```
for x in range(5):
    for y in range(5):
        print(str(x),str(y),str(x+y))
```

Rewrite them using **`while`** loops instead. For full points, the code must use **`while`** loops appropriately, and the output should be the same as if the original code is run. (1p)

10. Write a function `conjunctify(lst)` that takes a list `lst` of strings and returns a string with commas between all words except the last two where `and` should be written instead. (2p)

Example usage:

```
[In: ] print(conjunctify(["Flour","milk","butter","eggs","salt"]))
[Out:] Flour, milk, butter, eggs and salt
[In: ] print(conjunctify(["Flour","milk"]))
[Out:] Flour and milk
[In: ] print(conjunctify(["Flour"]))
[Out:] Flour
```

11. Write a program that asks the user `What is the answer to the ultimate question of life, the universe and everything?` and expects `42`. If a number other than `42` is entered, the program should print `Wrong` and ask the question again. If `42` is entered, the program should print `Congratulations!` and stop asking. For full points, you must use **`try`** and **`except`** so that the program does not crash if the user enters anything other than a number. (2p)

Example run:

```
What is the answer to the ultimate question of life, the universe and everything? Food
You must input a number
What is the answer to the ultimate question of life, the universe and everything? 10
Wrong
What is the answer to the ultimate question of life, the universe and everything? 5
Wrong
What is the answer to the ultimate question of life, the universe and everything? 42
Congratulations!
```

12. Write a program `sum_file(f)` that, given a filename `f` for a file containing numbers spread across multiple lines with spaces between them, prints the sum of the numbers. (2p)

Add error handling using **`try-except`** so that the program does not crash if you try to open a file that doesn't exist. (1p)

Example usage:

Given the file `numbers.txt` containing:

```
1 2
3
4 5 6 7
8 9
```

the following should happen:

```
[In: ] print(sum_file("numbers.txt"))
[Out:] 45
```

If you instead try to open `nonexistent.txt` which does not exist, the following should happen:

```
[In: ] print(sum_file("nonexistent.txt"))
[Out:] Error: the file does not exist
```

13. In the regular exam, you would write code to represent songs and albums using classes. An example solution could look like the following:

```
class Song:

    def __init__(self,a,n,l):
        self.artist = a
        self.name = n
        self.length = l

    def __str__(self):
        return self.artist + " - " + self.name

class Album:

    def __init__(self,a,n,ss):

        for s in ss:
            if s.artist != a:
                raise ValueError("All songs must have the same artist as the album")
```

```
        self.artist = a
        self.name = n
        self.songs = ss
```

This task is about extending the functionality in two ways.

A. Add a `__ge__` method to `Song` that allows comparing songs based on their length. (2p)

Example usage:

```
[In: ] s1 = Song("Bruce Springsteen","The Ties That Bind",(3,33))
[In: ] s2 = Song("Bruce Springsteen","Hungry Heart",(3,19))
[In: ] s3 = Song("Bruce Springsteen","The River",(4,59))
[In: ] print(s1 >= s2)
[Out:] True
[In: ] print(s2 >= s3)
[Out:] False
```

B. Add a `length` method to `Album` that calculates the length of the album by summing the length of the songs. The length should be returned as a pair `(m,s)` where `m` and `s` are the album's length in minutes and seconds. Remember that a minute has 60 seconds. (2p)

Example usage:

```
[In: ] a = Album("Bruce Springsteen","The River",[s1,s2,s3])
[In: ] print(a.length())
[Out:] (11,51)
```