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

Tentamensskrivning i
DA2005 Programmeringsteknik
7.5 hp
2021-02-03

- 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 utav 8 frågor med varierande poäng (totalt 12p).
- Inga externa bibliotek får användas om de inte nämns eller finns med i uppgiften, man får dock använda inbyggda funktioner som `len`, `range` och `map`.
- **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

1. Vilken *typ* har returvärdet från funktionen `input()`?

A. **`int`**
B. **`float`**
C. **`str`**
D. Det beror på vad man förser funktionen med som input
E. Inget av ovanstående

2. Vilka av följande påståenden är korrekta? (se kod till höger)

A. `d1` är av datatyp uppslagstabell.
B. Vi kan använda tupler som nycklar i uppslagstabeller.
C. Vi kan använda listor som nycklar i uppslagstabeller.
D. `d2` har `15` och `'l33t'` som nycklar.
E. Ett särfall lyfts om vi kör koden.

```
key1 = ('hej', 5)
key2 = [5, 'hej']
d1 = { key1 : [42, 13], 'hej' : 'a' }
d2 = { key2 : 15, 5 : 'l33t' }
print(len(d1) + len(d2))
```

3. Givet tilldelningarna i koden till höger, vilka av nedanstående uttryck utvärderas till `True`?

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

```
x = True
y = False
```

4. Vad är resultatet om vi skriver ut satsen `[i**2 for i in [j for j in range(6)if j % 2 == 1]]`?

A. `[0, 4, 16]`
B. `[0, 1, 4, 9, 16, 25]`
C. `[1, 9, 25]`
D. `[0, 4, 16, 64]`
E. `[1, 9, 25, 49]`

5. Vilka av slutsatser kan vi dra av koden till höger? (se kod till höger)

A. `A` är en superklass till `B`

B. `A` är en subklass till `B`

C. `hello` är ett attribut till klassen `B`

D. `HELLO` är ett klassattribut

```
class B:
    HELLO = 'hello'
    def __init__(self, hello='good bye'):
        self.hello = hello

class A(B):
    def __init__(self, hello=None):
        super().__init__()
        if not hello:
            pass
        else:
            self.hello = hello
```

6. Vad skrivs ut om vi kör koden till höger?

A. `[1, 3, 6, 8]`

B. `[2, 4, 7, 9]`

C. `[(3, 4), 7, [9]]`

D. `[(1, 2), (6, 7), 8]`

E. Ett särfall lyfts.

```
var = [ [(1, 2), (3, 4)], (6, 7), [[8], [9]] ]
res = map(lambda y: y[1], var)
print(list(res))
```

7. Vad blir resultatet av koden till höger?

A. `6`

B. `4`

C. `5`

D. `3`

```
def f1(a):
    c = f2(a, b)
    return a + c

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

a = 1
b = 2
print(f1(b))
```

8. Vad blir resultatet av koden till höger?

A. `4`

B. `7`

C. `17`

D. `21`

E. Ett särfall lyfts.

```
def f(x):
    if len(x) == 0:
        return 0
    else:
        return sum(x) + f(x[1:])

a = [1, 2, 4]
print(f(a))
```

Del B: kodfrågor

9. a **(1p)** Betrakta funktionen `flip` nedan. Nämn minst två datatyper förutom **`list`** som funktionen fungerar på.

b **(1p)** Skriv om funktionen så att den använder en **`for`**-loop istället för **`while`** och döp den till `flip_for` . Din funktionen ska ha samma beteende som `flip` .

```
def flip(s):
    i = 0
    s_str = ''
    while i < len(s):
        s_str += str(s[-(i+1)])
        i = i + 1
    return s_str
```

10. **(2p)** Låt bokstäverna A,C,G och T motsvara respektive tal 1,2,3,4. Skriv en funktion `dna_sum(s)` som tar en sträng med bokstäver A,C,G och T och skriver ut summan av strängen som ett *heltal*. Om strängen som ges som argument innehåller andra bokstäver än A,C,G och T, så ska dessa *inte* räknas med i summan. Om argumentet inte är en sträng så ska funktionen lämna ett felmeddelande (se nedan) och returnera.

Exempel på input och resulterande output:

```
[In] : dna_sum('AAA')
[Out]: 3
[In] : dna_sum('GGTA')
[Out]: 11
[In] : dna_sum('T')
[Out]: 4
[In] : dna_sum('')
[Out]: 0
[In] : dna_sum('ABaa')
[Out]: 1
[In] : dna_sum([1, 2, 3])
[Out]: ValueError: <dna_sum: argument must be a string>
```

11. **(2p)** Skriv en rekursiv variant `dna_sum_recursive(s)` utav funktionen `dna_sum(s)` ovan. `dna_sum_recursive(s)` ska ha samma beteende som `dna_sum(s)` .

12. **(2p)** Pythons inbyggda funktion **`hash()`** räknar ut ett lagringsvärde i form av ett heltal för ett godtyckligt icke-muterbart objekt i Python. Testa **`hash()`** för till exempel `42` eller `2.5` . Skriv en funktion `min_index(list_in)` som tar en lista `list_in` som har numeriska värden som element och returnerar platsen (index) på elementet i `list_in` med *lägst* lagringsvärde. Om listan är tom ska funktionen returnera `-1` .

Notera: ni ska **endast** **`int`** och **`float`** . Vissa andra datatyper, t.ex. **`str`** , är inte garanterade att ha samma lagringsvärde.

Exempel på input och resulterande output:

```
[In] : print(min_index([1,56565,-3]))
[Out]: 2
[In] : print(min_index([0.8,10**3, 2.5]))
[Out]: 1
[In] : print(min_index([]))
[Out]: -1
```

13. **(2p)** I koden nedan ser du två klasser definierade (`Car` och `MC`) som ärver från basklassen `Vehicle` . Konstruktorn till `Vehicle` ska som argument ta antalet hjul fordonet har (`n_wheels, int`) och fordonets maxhastighet (`max_velocity, float`). Vidare ska ett instansattribut `helmet_needed` sättas till `False` om fordonet har fler än två hjul och `True` annars, och ett instansattribut som specificerar typen av fordon (`vehicle_type`). Skriv färdigt klassen `Vehicle` så att följande kod:

```
class Vehicle(object):
    def __init__(self, n_wheels, max_velocity):
        # rest of constructor here

    # possible other functionality in the base class

class Car(Vehicle):
    def __init__(self, max_velocity):
        super().__init__(4, max_velocity)
        self.vehicle_type = 'Car'

class MC(Vehicle):
    def __init__(self, max_velocity):
        super().__init__(2, max_velocity)
        self.vehicle_type = 'MC'

car = Car(180)
mc = MC(130)
print(car)
print(mc)
```

ger outputen:

```
<Car | Helmet needed: False, Max velocity: 180 km/h>
<MC | Helmet needed: True, Max velocity: 130 km/h>
```

14. **(2p)** Du har en text-fil med data där första kolumnen är ett heltal och andra kolumnen är ett flyttal. Kolumnerna är separerade av ett kommatecken. Ett exempel är nedanstående (som vi kallar `data.txt`):

```
1,3.12
4,4.8
6,32.3
2,-2.1
56,13
42,-99
```

Skriv en funktion `select_data(file_name, filter_float)` som tar en sträng som första argument och ett flyttal som andra argument. Funktionen ska med hjälp av **`open, map, filter, lambda`** läsa in datat i filen med namn `file_name`, filtrera det inlästa datat så att alla rader som har ett värde i andra kolumnen som är *större än* `filter_float` behålls, och skriva det behållna datat till en fil med namn `output.txt`.

Notera: ni ska använda er av funktionell programmering med **`map, filter, lambda`** och ska **ej** använda er av **`for, while`**-loopar.

Exempel på användning och output:

```
[In]: select_data(data.txt', 8.3)

# produces 'output.txt' with content
6,32.3
56,13

[In]: select_data('data.txt', -3)

# produces 'output.txt' with content
1,3.12
4,4.8
6,32.3
2,-2.1
56,13
```