DA2004 Programmering för matematiker, 2019-03-18

- **Skriv tydligt.** Svårlästa svar riskerar 0 poäng.
- Skriv bara på en sida av varje papper!
- 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 (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 du *längden* på en lista `lista`?

a. `lista`
b. `lista[0]`
c. `len(lista)`
d. `lista.len()`
e. Inget av ovanstående.

2. Hur skapar man en modul m i Python, så att man i sitt program kan skriva `import m`?

a. Skapa filen m.py och använd nyckelorder *module* för att markera att det är en modul.
b. Skapa filen m.py och definiera en klass i filen.
c. Skapa filen m.py och definiera en klass i filen. Klassen ska ärva från basklassen "module".
d. Skapa filen m.py och skriv högst upp `export f1, f2` osv, där f1 och f2 är funktioner i m.py som ska exporteras.
e. Skapa filen m.py och gör ingenting: alla Python-filer kan användas som om de vore moduler.

3. Vilket resultat skrivs ut av koden till höger?

a. 2
b. 3
c. 5
d. 6
e. Ingenting, det blir ett särfall.

```
def a(x, y):
    return lambda z: x + y + z

f = a(2,3)
print(f(1))
```

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

a. 2
b. 3
c. 4
d. 5
e. Ingenting, det blir ett särfall.

```
def f(x):
    return x*y

def g(y):
    return x*y

x = 1
y = 2
print(f(2) + g(1))
```

5. Vad använder man **`assert`** till?

a. Man skriver in villkor som kan avslöja fel i koden.
b. Man fångar särfall (*exceptions*) som uppstått på grund av **`raise`**.
c. Man använder det för multipla returvärden.
d. Man använder det för att instantiera objekt från klasser.
e. Ingenting, instruktionen finns inte i Python.

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

a. **`str`**
b. **`int`**
c. **`list`**
d. **`dict`**
e. Det beror på vad användaren skriver.

7. Vilka påståenden är sanna?

a. Funktioner kan returnera funktioner.
b. Funktioner kan inte vara utan parametrar.
c. Funktioner kan anropa sig själva.
d. Funktioner kan ha flera **`return`**-satser.
e. Inget av ovanstående.

8. Vad är resultatet av anropet `f([1,[2,3], [4,5]])` om man definierat f som nedan?

a. Siffran 1 skrivs ut.
b. Siffrorna 1, 2, och 4 skrivs ut på var sin rad.
c. Siffrorna 1 till 5 skrivs ut på var sin rad.
d. Siffran 1 och listorna `[2,3]` samt `[4,5]` skrivs ut på var sin rad.
e. Ett särfall

```
def f(lis):
    if type(lis) == list:
        for e in lis:
            f(e)
    else:
        print(lis)
```

9. Vilket eller vilka exempel är korrekt skrivna uppslagstabeller (*dictionaries*).

a. `{}`
b. `()`
c. `{'a':'b', 'c':'d'}`
d. `(('a','b'), ('c','d'))`
e. Inget av ovanstående.

10. Vilken eller vilka boolska variabeltilldelningar gör att utrycket

```
(x and y) or (x and z)
```

evaluerar till `True`?

a. `x=False, y=False, z=True`
b. `x=False, y=True, z=False`
c. `x=True, y=False, z=True`
d. `x=True, y=True, z=False`
e. `x=True, y=True, z=True`

Del B: kodfrågor

11. Vad är skillnaden mellan `=` och `==` i Python?

12. Vad är resultatet av anropet `g('hubba')` efter att koden nedan har definierats? Motivera ditt svar.

```
def g(s):
    if len(s) > 0:
        return g(s[1:]) + s[0]
    else:
        return s
```

13. Under vilken eller vilka omständigheter blir resultatet utskriften "Hoppsan!" när man exekverar följande kod?

```
try:
    s = input('Vilken multiplikationstabell? ')
    for i in range(1,13):
        print(s, '*',i,'=',i*int(s))
except:
    print('Hoppsan!')
```

14. Här nedan är en funktion för att beräkna den itererade logaritmen, även känd som $\log^*$, i valfri bas.

$$\log_b^*(n) = \begin{cases} 0 & n \leq 1 \\ 1 + \log_b^*(\log_b(n)) & n > 1 \end{cases}$$

Koden bryter mot en viktig princip som vi har tagit upp i kursen (och som är allmän praxis inom programmering). Skriv om koden så att den följer kursens tumregler.

```
from math import log
def logstar():
    '''
    Compute the iterated logarithm of the variable 'n' in base 'base'.
    To compute log*(10) in base 2, write:
       n = 10
       base = 2
       res = logstar()
    '''
    log_levels=0
    my_number = n
    while my_number > 1:
        my_number = log(my_number, base)
        log_levels += 1
    return log_levels
```

15. Skriv funktionen `prefix_sums` som tar en sekvens med tal och returnerar en lista med prefix-summorna. Dvs, givet tal $x_1, \ldots, x_n$, beräkna $y_i = \sum_{j=1}^{i} x_j$ för $1 \leq i \leq n$. Funktionen ska fungera så här:

```
In [1]: prefix_sums([])
Out [1]: []

In [2]: prefix_sums([1,1,1,1,1])
Out [2]: [1, 2, 3, 4, 5]

In [3]: prefix_sums(range(5))
Out [3]: [0, 1, 3, 6, 10]
```

16. Skriv en funktion `count_nucleotides(s)` som räknar antalet A, C, G, och T i strängen s, som antas representera DNA. Du kan anta att s bara innehåller tillåtna bokstäver, som versaler. Exempel på användning, inifrån Spyder:

```
In [1]: count_nucleotides('')
Out [1]: {'A': 0, 'C': 0, 'G': 0, 'T': 0}

In [2]: count_nucleotides('AACAATT')
Out [2]: {'A': 4, 'C': 1, 'G': 0, 'T': 2}
```

17. Funktionen `capitalize` (nedan) tar en sträng s som parameter och returnerar en ny version av s där varje ord börjar med stor bokstav. Det är meningen att funktionen ska fungera så här:

```
In [1]: capitalise('abc')
Out [1]: 'Abc'

In [2]: capitalise('a b c')
Out [2]: 'A B C'

In [3]: capitalise('tillfällige ständige sekreteraren')
Out [3]: 'Tillfällige Ständige Sekreteraren'
```

Tyvärr blir det något fel: koden är inte korrekt. Förklara i ord vad det är som är fel!

```
import string
def capitalise(s):
    '''
    Input:   a string
    Returns: a new string in which every word start with an upper-case letter.
    '''
    cs = ''
    time_for_upper_case = True
    for c in s:
        if c in string.ascii_letters and time_for_upper_case:
            c = c.upper()     # Change character to upper case
            time_for_upper_case = False
        elif c == ' ':
            time_for_upper_case = True
    return cs
```

Notera att `string.ascii_letters` finns i modulen string och innehåller bokstäverna i det engelska alfabetet. Metoden `upper` är definierad för strängar.

18. Vad menas med *attribut* i samband med objektorienterad programmering? Man avser:

a. de klasser som definierats i ett program.
b. de objekt som skapats i ett program.
c. variabler associerade med en klass.
d. funktioner associerade med en klass.
e. inget av ovanstående.

19. I figur 1 finns en liten början till en klass för hantering av DNA-sekvenser, dvs strängar på formen `'ACGTAAAGTC'` (till exempel) av varierande längd.

Visa hur man instantierar ett objekt av klassen `DNA` till variabeln `gene`.

20. Gör klassen lite mer användbar genom att lägga till en metod `composition` som räknar och returnerar de olika nukleotiderna i en DNA-sekvens. Du kan använda dig av funktionen som definierats i fråga 16, även om du inte har löst den uppgiften.

Din lösning ska vara skriven så att man kan göra anropet

```
gene.composition()
```

om `gene` är ett objekt av klassen `DNA`.

```
class DNA:
    def __init__(self, s):
        self._sequence = s
```

Figur 1: Kodutkastet till frågorna 19 och 20.