Tentamen Programmeringsteknik I 2020-08-08

Skrivtid: 14:00-19:00. 30 minuters extra marginal ges för inlämning efter skrivtiden. Inlämningen stänger alltså slutligt 19:30 för personer med ordinarie skrivtid.

Mer information: Information om hemtentamen finns i förväg på
`https://www.it.uu.se/edu/course/homepage/prog1/python/vt20/exam.html`.
Eventuella förtydliganden läggs även till där under tentans gång.

Tänk på följande

- Om du har en anmälningskod anger du den på första svarssidan (ovanför uppgift A1).
- Ladda i första hand upp din besvarade tenta i Studium. Maila i andra hand ifylld PDF till `carl.nettelblad@it.uu.se`. Kontrollera att du sparar PDF-filen på ett sådant sätt att din sparade fil fortfarande är ett redigerbart formulär. Om du inte lyckas fylla i PDF-filen, skriv svar på uppgifterna *i ordning med tydlig markering av varje deluppgift* i en textfil eller ett Worddokument. I sista hand kan avfotograferade eller scannade lösningar på papper godtas.
- Såvida inget annat anges får man bygga på lösningar till föregående uppgifter även om dessa inte har lösts.
- Det är tillåtet att införa extra metoder eller funktioner. Uttryck som ”skriv en funktion som” ska alltså *inte* tolkas så att lösningen inte får struktureras med hjälp av fler funktioner.
- Alla uppgifter gäller programmeringsspråket Python och programkod ska skrivas i Python. Koden ska vara läslig, dvs. den ska vara vettigt strukturerad och indenterad. Namn på variabler, funktioner, metoder, klasser etc. ska vara beskrivande men kan ändå hållas ganska korta.

  Observera att betyget kan påverkas negativt bland annat av:
  - onödiga variabler,
  - dålig läslighet,
  - upprepning av identisk kod,
  - underlåtenhet att utnyttja given eller egen tidigare skriven kod,
  - hög grad av ineffektivitet, som att upprepa ett omfattande funktionsanrop ett potentiellt stort antal gånger.

Observera

Skrivningen består av två delar. Lösningarna till uppgifterna på del A har svarsrutor i direkt anslutning. Rutorna är tilltagna i storlek så att de ska rymma svaren. Om du ändå inte får plats finns det extra tomsidor i slutet. Markera gärna i så fall att dessa använts.
Lämna rättningsrutorna tomma! För A1 finns extra utrymme att kommentera dina kryssvar om du vill förklara något.

Svaren på del B skrivs ett per speciellt anvisad sida, efter uppgiftstexterna.

Det är ditt ansvar att tentan blir elektroniskt inlämnad med svar på alla uppgifter du önskar besvara. Kontrollera att det verktyg du använder klarar av att spara ifyllda PDF-formulär, så du inte bara får en tom fil eller en fil där formulärfälten försvinner. Adobe Acrobat Reader klarar garanterat att spara och läsa våra filer. Se till att spara ditt formulär ofta och kontrollera på slutet att du verkligen laddar upp rätt fil!

Bedömning och betygsättning

För att bli godkänd (betyg 3) krävs att A-delen i huvudsak är rätt löst. Det betyder inte att varje uppgift behöver vara exakt rätt, men att du ska visa att du i huvudsak uppfyller kursens mål, som säger att studenten efter godkänd kurs ska kunna:

- redogöra för de grundläggande begreppen modul, funktion, klass, objekt och därtill hörande underbegrepp
- analysera och lösa problem med hjälp av programmeringskonstruktioner
- förklara vad ett givet program i Python utför
- använda befintliga moduler och skriva program med flera samverkande komponenter i Python
- använda en programutvecklingsmiljö
- testa och felsöka program

För betyget 4 krävs dessutom att minst hälften, och för betyg 5 alla, uppgifterna på B-delen är i stort sett rätt lösta. Vid bedömning för dessa betyg tas även hänsyn till kvaliteten på lösningarna på A-delen.

Observera att B-delen normalt sett endast rättas om A-delen är godkänd.

Hjälpmedel

En del viktiga aspekter av Python tas upp på separat referensblad.

Eftersom detta är en hemtenta tillåts du använda tillgängliga elektroniska och fysiska resurser. Det innefattar att köra kod i Python och söka i dokumentation, kurshemsidor och liknande resurser på nätet. Det innefattar **inte** att ställa frågor eller på andra sätt kommunicera med andra människor. Tentamen ska göras enskilt. Om du utgår från ett specifikt exempel i din kod ska du ange källan till detta. Kopiering av kod utan att ange källa kan komma att betraktas som plagiat – utgångspunkten är att du ska formulera dina egna svar i sin helhet. Misstanke om fusk eller plagiat kan, precis som vid annan examination, anmälas till universitetets disciplinnämnd.

Om du inte minns exakt vad en viss funktion heter eller hur en del av språkets syntax ser ut kan du påpeka detta och beskriva vilka antaganden du gör. Vi försöker både bedöma hur väl du har löst problemet och hur väl du kan hantera Pythons språk och funktionalitet.

Lycka till!

Anmälningskod:

Del A (obligatorisk för alla)

A1. Kryssa för rätt alternativ, endast ett om frågan inte uttryckligen nämner att flera kan vara möjliga.

a) Vad är sant om en sträng?

- ☐ 1) En och samma sträng kan högst finnas en gång som värde i ett lexikon
- ☐ 2) En sträng kan ha värdet `None`
- ☐ 3) En sträng är oföränderlig
- ☐ 4) Funktionen `float` omvandlar en sträng till ett värde av heltalstyp
- ☐ 5) Strängar är en sorts listor

b) Vad är `sin` i följande kod?

```
import math

x = math.sin(3.14159)
```

- ☐ 1) En funktion
- ☐ 2) En variabel
- ☐ 3) En parameter
- ☐ 4) En modul
- ☐ 5) En klass
- ☐ 6) Det går inte att säga

c) Vad är `classlib` i följande kod?

```
import classlib
```

- ☐ 1) En funktion
- ☐ 2) En variabel
- ☐ 3) En parameter
- ☐ 4) En modul
- ☐ 5) En klass
- ☐ 6) Det går inte att säga

d) Vilken metod i klassen `str` returnerar en ny sträng utan inledande blanktecken, men i övrigt oförändrad? Se `https://docs.python.org/3/library/stdtypes.html#string-methods`

- ☐ 1) `str.find`
- ☐ 2) `str.lstrip`
- ☐ 3) `str.partition`
- ☐ 4) `str.split`
- ☐ 5) `str.strip`
- ☐ 6) `str.rpartition`
- ☐ 7) `str.rfind`
- ☐ 8) `str.rsplit`
- ☐ 9) `str.rstrip`

e) Vad kallar man av konvention den första parametern till en metod?

- ☐ 1) `me`
- ☐ 2) `__init__`
- ☐ 3) `this`
- ☐ 4) `self`
- ☐ 5) `append`
- ☐ 6) `0`

f) Vad bör `x` ha för värde för att följande kod ska fungera?

```
with open('ourfile.txt', x) as file:
    data = file.readlines()
```

- ☐ 1) `''`
- ☐ 2) `'r'`
- ☐ 3) `'w'`
- ☐ 4) `None`
- ☐ 5) `False`
- ☐ 6) `True`

g) Vad är sant för varje magisk metod?

- ☐ 1) En funktion som anropas utan att ingå i klassen
- ☐ 2) En metod med valfritt antal parametrar
- ☐ 3) Alla metoder är magiska
- ☐ 4) En metod med ett specifikt namn som anropas underförstått av Python
- ☐ 5) En metod som kan representera ett objekt som en sträng
- ☐ 6) En metod som initierar ett objekt
- ☐ 7) En metod som modifierar ett objekt

h) Hur många olika listor skapar följande kod?

```
a = [1, (2, 3)]
b = a
c = a[1]
d = a[1][0]
e = a[1][:]
f = a.copy()
```

- ☐ 1) 1
- ☐ 2) 2
- ☐ 3) 3
- ☐ 4) 4
- ☐ 5) 5

i) Vilken typ har följande uttryck?

```
'[{"1":1}, 2.5, 3, 4]'
```

- ☐ 1) dict
- ☐ 2) float
- ☐ 3) int
- ☐ 4) list
- ☐ 5) str

Kommentar till mina svar på A1:

Rättning:

A2. Funktionen `todict` ska ta emot en lista med ord (strängar) som parameter och returnera ett lexikon. Nycklarna i detta lexikon ska vara det första tecknet ur varje sträng, skrivet med versal. Värdet som kopplas till varje nyckel ska vara en *lista* med alla ord som börjar på det tecknet. Skriv klart funktionen. Tänk på att `str` har en metod `upper` som returnerar en ny sträng där alla tecken är versaliserade.

```
def todict(lst):
    """Create a dictionary with single uppercase letters used as keys, for each
    entry indexing all words from lst starting with that letter as a list."""
```

Rättning:

A3. Skriv en funktion `reciprocal(lst)` som tar emot `lst` som en parameter och returnerar en ny lista där varje element är 1 delat med motsvarande element ur `lst`. Listan `[1, 4, 5]` ska alltså ge listan `[1, 0.25, 0.2]`.

Skriv också, utanför funktionen, exempelkod som använder funktionen på listan `[5.0, 250.0, 10.0]` och skriver ut resultatet.

Rättning:

A4. Nedan finns innehållet i en funktion. Föreslå hur början av funktionsdefinitionen (dess ”huvud”) ska se ut med namn och parametrar. Skriv även en lämplig docstring.

```
res = 1
if y < 0:
    x = 1/x
    y = -y
for _ in range(y):
    res *= x
return res
```

Rättning:

A5. Definiera en ny klass `Student` som beskriver en student som har ett namn och ett betyg. Dess initieringsmetod ska ta in två parametrar `first` och `last`, men lagra hela namnet som en sträng i ett enda attribut (en instansvariabel) `name`. Dessutom ska det finnas ett attribut `grade` som från början har värdet `None`.

Rättning:

A6. Skriv en metod `setGrade` som sätter en students betyg. Betyget måste dock vara giltigt, annars ska inget hända. Giltiga betyg är 0 (d.v.s. U), 3, 4, 5. Dessutom får man aldrig sänka ett tidigare satt betyg. Om inget parametervärde anges ska betyget 3 vara underförstått. Man ska alltså exempelvis kunna skriva `s.setGrade(5)` eller `s.setGrade()` för ett studentobjekt `s`.

Rättning:

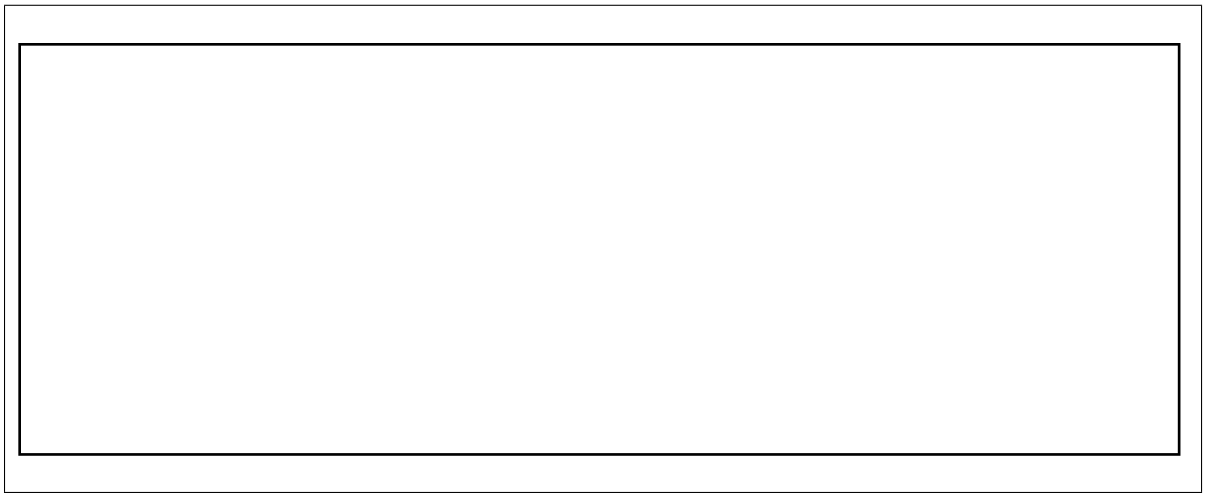
A7. Skriv klart metoden `__str__` för `Student`, som ska representera studenten som en sträng. Detta ska vara på formen namn:betyg, t.ex. `'Viola Frilin:4'`. Om betyg saknas ska tecknet - (bindestreck) användas. Värdet 0 ska översättas till U.

```
def __str__(self):
    """Represents the student object as a string, name:grade. Missing grades
    represented by dash (-), 0 values as the letter U."""
```

Rättning:

A8. Nedanstående program innehåller ett antal mindre skrivfel. Skriv den korrekta koden med korta kommentarer om vad som var fel.

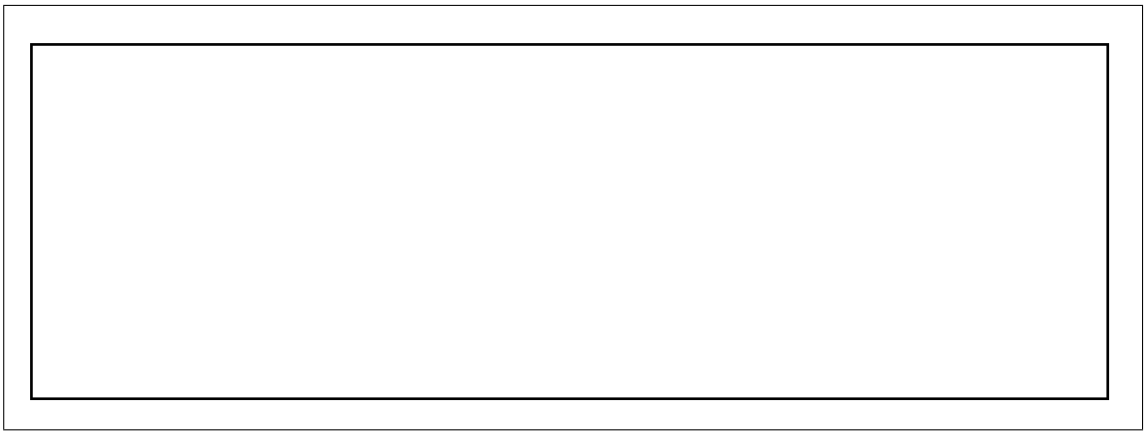
```
tupleoftuples = ((1,2), (3,4), (5,9)]
t = Tupleoftuples
for a b in t
    print('{a} + {b} = {a+b}')
```



Rättning:

A9. Ibland talar man om redigeringsavstånd mellan två strängar. Det är hur många ändringar man behöver göra för att den ena strängen ska bli den andra strängen. I det här fallet begränsar vi oss till strängar av samma längd och redigeringsavståndet definieras som hur många tecken man måste *byta ut* för att den ena ska bli den andra. Till exempel är redigeringsavståndet mellan `'fyra'` och `'mura'` 2, medan redigeringsavståndet mellan `'fyra'` och `'myra'` är 1, precis som avståndet mellan `'fyra'` och `'fyla'`. Skriv klart funktionen `editDist` nedan som räknar ut denna typ av redigeringsavstånd.

```
def editDist(a, b):
    """Compute the substitution edit distance between two strings a and b of
    equal length."""
```



Rättning:

A10. Skriv en funktion som utifrån ett startord och en lista med giltiga ord av samma längd returnerar alla som har exakt redigeringsavstånd 1. Ordet `'fyra'` och listan `['myra', 'boll', 'fyla', 'mura']` ska ge listan `['myra', 'fyla']`. Se till att välja lämpligt namn på funktionen och dess parametrar, liksom att dokumentera funktionen med docstring.

Rättning:

Del B (för betyg 4 och 5)

Svaren till dessa uppgifter ges på separata svarssidor längst bak. Se till att skriva varje uppgift på anvisat blad.

B1. Antag att listan `students` innehåller ett antal objekt av klassen `Student` från tentamens A-del. Skriv kod för att skapa en ny lista `students2` som bara innehåller de studenter som har något av de giltiga betygen 3, 4, 5 (och alltså inte 0 eller `None`) och där dessa studenter är sorterade i stigande ordning efter betyg.

Använd Pythons funktionalitet i språk och standardbibliotek för att uttrycka detta tydligt och kompakt. Det kan innebära att undvika att skapa tillfälliga listor och att inte gå igenom hela listor med uttryckliga loopar, om det finns andra möjligheter.

B2. Klassen `Square` nedan beskriver ett kvadratiskt rutnät med punkter, representerat som en lista med rader, där varje rad i sin tur består av en lista av strängar med längd 1. Klassen är given, förutom metoden `plot`, som ska anropas med tre parametrar – koordinaterna `x` och `y`, samt ett tecken `c` som ska placeras på rätt plats. Koordinaterna ska vara i intervallet $0 \leq x, y < 1$. Du måste översätta detta till lämpliga index (motivera vid behov).

```
class Square:
    def __init__(self, side):
        """Initialize a list of lists with spaces in all position, representing
        an empty side * side square."""
        self.map = [[' '] * side for _ in range(side)]

    def plot(self, x, y, c):
        """Plot character c at the appropriate integer location within
        the current map, for x and y coordinates in the range
        0 <= x, y < 1."""
        pass # To be implemented by you

    def __str__(self):
        """Present the map as a single string."""
        return '\n'.join([''.join(row) for row in self.map])


s = Square(5)
# We can plot something manually
s.map[3][3] = 'x'
# We should be able to do so using the plot method as well
s.plot(0.45, 0.73, 'y')
print(s)
```

B3. Koden nedan fungerar, i meningen att den gör någonting som stämmer med dess syfte. Den är däremot olämplig eftersom sättet att använda Python och namngivningen gör det svårt att följa vad som händer.

Föreslå hur man skulle kunna skriva den på ett sätt som bättre använder möjligheterna i Python och gör det lättare att följa vad som sker. Motivera vilka konstruktioner du använder, eventuellt genom att jämföra med andra möjliga alternativ.

```
wordlist = []
with open('words.txt', 'r') as a:
    b = a.readlines()
    i = 0
    while i < len(b):
        k = ''
        j = 0
        while j < len(b[i]):
            if b[i][j].isspace():
                if len(k) > 0:
                    wordlist.append(k)
                k = ''
            else:
                k += b[i][j]
            j += 1
        if len(k) > 0:
            wordlist.append(k)
        i += 1
print(wordlist)
```

B4. I A-delen skrev du funktioner för att identifiera ords redigeringsavstånd och hitta sådana som hade redigeringsavstånd 1 mellan varandra.

Vi kan också vilja hitta en redigeringsväg mellan två ord, det vill säga en följd av giltiga ord som går mellan en startpunkt och en slutpunkt. Den kortaste redigeringsvägen kan vara längre än redigeringsavtåndet. Tänk exempelvis på orden snok och krok. De har redigeringsavstånd 2, men eftersom varken srok eller knok är ett ord blir den kortaste redigeringsvägen längre än två. Beroende på vad man har för ordlista skulle en möjlig väg kunna vara snok, slok, klok, krok, med tre steg utöver startordet.

Det går att hitta en sådan redigeringsväg genom att dels lagra ett lexikon med alla ord där man har funnit en känd redigeringsväg från startordet, dels en kö med vilka ord som ska testas. Genom att använda lexikonet undviker man att undersöka samma ord flera gånger. I startläget ska lexikonet bara innehålla startordet, med en tom lista, d.v.s. `{'snok':[]}` i vårt exempel. Kön ska bara innehålla startordet, alltså `['snok']`.

Tills man hittat en väg till målordet gäller det då att steg för steg ta det första elementet ur kön och hitta alla giltiga ord som har redigeringsavstånd 1 till det ordet. De ord som *inte* redan finns i lexikonet läggs till där, med den nya redigeringsvägen, och sist i kön. Så fort man hittat en väg till målordet är den en giltig lösning på problemet. Om man kommer till slutet av kön utan att ha hittat en väg finns ingen väg.

Skriv funktionen `editPath(start, end, wordlist)` som gör detta.

Exempel: `editPath('snok', 'krok',`
`['snok', 'stag', 'krus', 'snus', 'spis', 'gris', 'klok', 'slog', 'slag', 'pass', 'pris',`
`'klös', 'krok', 'klot', 'slok'])` ska returnera `['slok', 'klok', 'krok']`.

Svar till B1:

Rättning:

Svar till B2:

Rättning:

Svar till B3:

Rättning:

Svar till B4:

Rättning:

Extra svar sida 1:

Extra svar sida 2:

Extra svar sida 3:

Extra svar sida 4:

Extra svar sida 5:

Extra svar sida 6: