Tentamen Programmeringsteknik I 2020-03-20

Skrivtid: 08:00-13:00. 20 minuters extra marginal för elektronisk inlämning.

Tänk på följande

- Om du var anmäld till salstentan anger du din anmälningskod på första svarssidan (ovanför uppgift A1).
- Om du laddar upp i Studentportalen anger du din anonymkod i fältet under. (Du kan alltså ange båda.)
- Ladda i första hand upp din besvarade tenta i Studentportalen. Maila i andra hand ifylld PDF till `carl.nettelblad@it.uu.se`. 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” skall alltså *inte* tolkas så att lösningen inte får struktureras med hjälp av fler funktioner.
- Alla uppgifter gäller programmeringsspråket Python och programkod skall skrivas i Python. Koden skall vara läslig dvs den skall vara vettigt strukturerad och indenterad. Namn på variabler, funktioner, metoder, klasser etc skall 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. 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!

Var god vänd!

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 bör 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.

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: | |

Anonymkod: | |

Del A (obligatorisk för alla)

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

a) Vilken funktion i modulen `math` räknar ut den euklidiska längden på en vektor som lagras som en lista?
Se `https://docs.python.org/3/library/math.html`

- ☐ 1) `ceil`
- ☐ 2) `comb`
- ☐ 3) `dist`
- ☐ 4) `erf`
- ☐ 5) `hypot`
- ☐ 6) `lgamma`

b) Vad är sant om en tupel?

- ☐ 1) Alla element måste ha samma typ
- ☐ 2) En tupels ingående värden/referenser kan bara anges när den skapas
- ☐ 3) En tupel har alltid mer än ett element
- ☐ 4) Om man gör `print` på en tupel får man alltid samma resultat eftersom den är oföränderlig
- ☐ 5) Varje sträng är också en tupel

c) Vad är `random` i följande kod?

```
import random

x = random.randint(0, 100)
```

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

d) Vad är `Worker` i följande kod?

```
x = Worker
y = x(999)
```

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

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) Om följande rad finns i ett program och inte får ändras, vad är viktigast att se till också finns i programmet?

```
file = open('ourfile.txt', 'r')
```

- ☐ 1) `file.close()`
- ☐ 2) `text = file.readlines()`
- ☐ 3) `print(file)`
- ☐ 4) `for c in file.read():`
- ☐ 5) `with file.readlines() as lines:`

g) Vad är skillnaden på en funktion och en metod?

- ☐ 1) Man kan själv definiera egna funktioner
- ☐ 2) Man kan själv definiera egna metoder
- ☐ 3) Alla metoder är magiska
- ☐ 4) En funktion anropas på ett objekt
- ☐ 5) En metod anropas på ett objekt
- ☐ 6) Bara funktioner kan modifiera objekt
- ☐ 7) Bara metoder kan modifiera 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][1]
print(a.pop)
```

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

i) Vilken typ har följande uttryck?

```
[1, 2, 3.5, 4][2]
```

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

Kommentar till mina svar på A1:

Rättning:

A2. Funktionen `cube(lst)` ska ta emot en lista som parameter och returnera en ny lista med varje tal taget i kubik (upphöjt till tre). Skriv den nödvändiga koden för funktionen, samt ett exempel som använder funktionen för att beräkna och skriva ut kuberna för talen 5, 8 och 9.

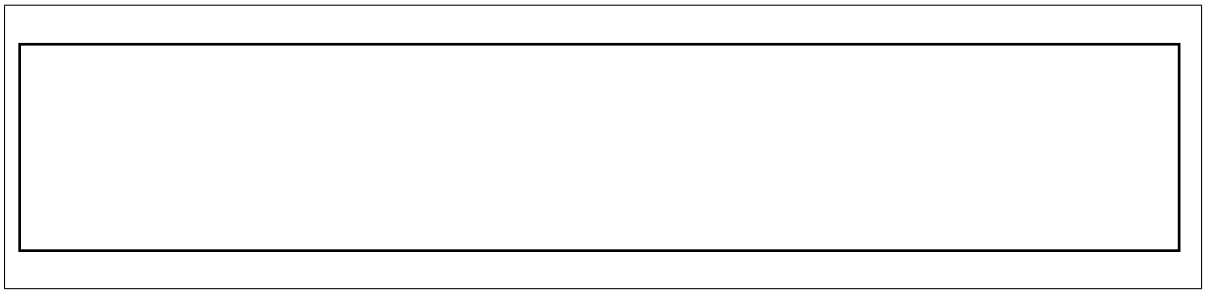
Rättning:

A3. Funktionen `capitalize` ska ta emot ett ord som parameter i form av en sträng och returnera en version där det första tecknet skrivs med versal. Övriga tecken ska vara oförändrade. Exempelvis ska `'hej'` bli `'Hej'`. Skriv klart funktionen. Tänk på att `str` har en metod `upper` som returnerar en ny sträng där alla tecken är versaliserade.

```
def capitalize(word):
```

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.

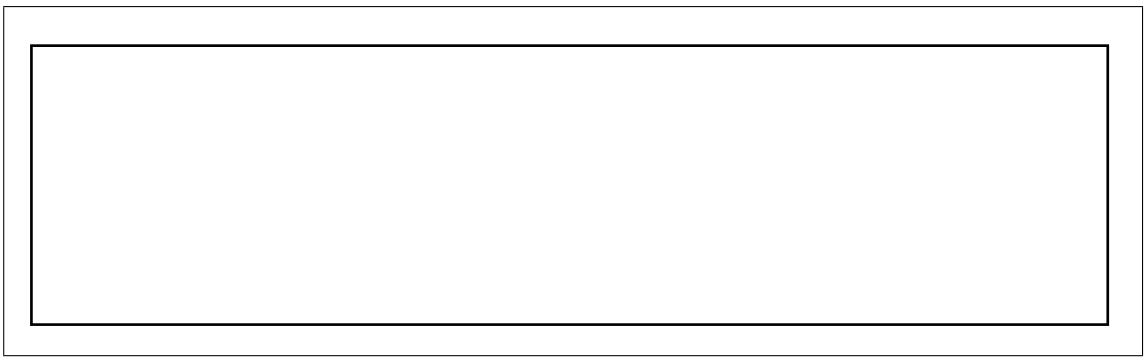


```
    count = 0
    for c in line:
        if c == 'Q':
            count += 1
    return count
```

Rättning:

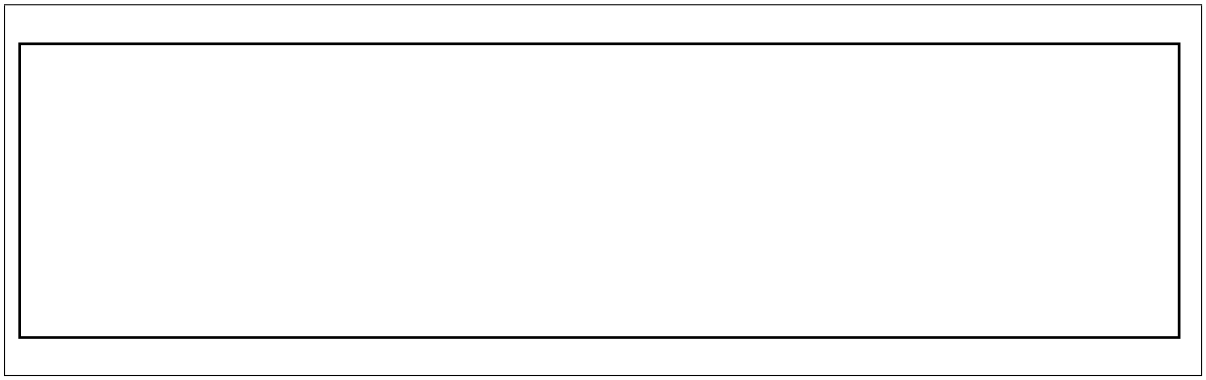
A5. Funktionen `longestWord` ska returnera det längsta ordet från en sträng. Tänk på att metoden `split` kan användas för att dela upp en sträng i ord. Skriv innehållet till funktionen.

```
def longestWord(data):
    """Returns the longest word found in input string data. Words are
     defined as being delimited by whitespace."""
```



Rättning:

A6. Definiera en ny klass `Point` som beskriver en koordinat i ett tvådimensionellt rum. Dess initieringsmetod ska ta in två parametrar `x` och `y` och internt lagra koordinaterna som en tupel i ett attribut (en instansvariabel) `pos`. Det ska inte finnas några andra attribut.



Rättning:

A7. Skriv en ny metod till `Point`, `rot90`. `rot90` ska vrida den befintliga punkten i steg om 90 grader moturs. `p.rot90()` ska då vrida 90 grader moturs, `p.rot90(2)` ska vrida 180 grader moturs och både `p.rot90(3)` och `p.rot90(-1)` ska vrida 270 grader moturs (vilket även motsvarar 90 grader medurs).

Ledning: Rotation 90 grader moturs ($\pi/2$ radianer) motsvarar multiplikation med i för komplexa tal. Annorlunda uttryckt kan de nya roterade koordinaterna (x_r, y_r) skrivas som:

$x_r = -y$
$y_r = x$

Rättning:

A8. Python använder magiska metoder med specifika namn för specifik funktionalitet. Skriv klart metoden `__eq__` för `Point`. `__eq__` används av Python när jämförelser utförs med `==`. Metoden ska returnera `True` om två punktobjekt har samma koordinater och `False` annars.

```
def __eq__(self, other):
    """Compares self to other, returning True if two Point instances
    represent the same coordinates, False otherwise."""
```

Rättning:

A9. Antag nu att all kod du har skrivit i `Point` finns i filen `smallvector.py`. Skriv den nödvändiga koden för att använda `Point` från en annan Pythonfil, skapa två vektorer med koordinaterna $x_1 = 3, y_1 = 5$; $x_2 = -5, y_2 = 3$, rotera den andra 270 grader moturs, jämföra dem med == och skriva ut resultatet.

Rättning:

A10. Nedanstående program innehåller ett antal mindre skrivfel. Skriv den korrekta koden med korta kommentarer om vad som var fel. Funktionen `type` returnerar typen för det värde som anges som parameter.

```
lex = {}
lex{'a'} = 'hej'
lex['b'] = [5 9]
for k in lex.keys:
    print('Information for key {k}')
    print f'Val: {lex[k]}, Type: {type(lex[k])}'
```

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. Skriv innehållet till nedanstående funktion. Den ska bygga upp ett lexikon med nycklar och värden. För varje sådan post ska den läsa in två rader från tangentbordet. Den första raden ska utgöra nyckeln som ska lagras i en form med enbart versaler. Användaren skriver in värdet på nästa rad. Detta ska lagras som ett heltal. Om man anger en tomrad som nyckel ska inläsningen avslutas och lexikonet returneras.

```
def read_uppercase_keys_and_ints():
    """Reads two lines per entry, one with key, one with value, stores all
    entries in a dictionary, which is returned. Keys are stored as UPPERCASE only,
    values as ints. Input is stopped by entering an empty line as key."""
```

B2. Antag att du har en lista `plist` där elementen är punkter i form av objekt av klassen `Point`, som du skrev i föregående tentamensdel. Vi vill nu sortera punkterna i *vinkelordning* (relativt origo $(0, 0)$, med vinkel 0 grader/radianer pekande rakt åt höger $(1, 0)$ och positiva vinklar gående moturs). Resultatet av sorteringen, fortfarande objekt av klassen `Point`, ska lagras i `plist2`. Den ursprungliga listan ska inte förändras. Skriv kod som utför detta. Du kan utnyttja funktionen `atan2` i modulen `math`, som kan användas för att ta reda på vinklar enligt denna definition. Den har följande dokumentation:

math.atan2(y, x) Return atan(y / x), in radians. The result is between -pi and pi. The vector in the plane from the origin to point (x, y) makes this angle with the positive X axis. The point of atan2() is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For example, atan(1) and atan2(1, 1) are both pi/4, but atan2(-1, -1) is -3*pi/4.

B3. Återstående uppgifter handlar om två klasser, `FallingBall` och `RollingBall`. Dessa ska utföra en mycket enkel (direkt dålig) simulering med tidsstegning, som styrs av koden nedan. Din kod ska även fungera om konstanterna i exemplet nedan byts ut eller om man skapar flera objekt av samma klass.

```
balls = [FallingBall(10, 0, 0.5), RollingBall(10, 10, 0.2)]

timestep = 0.01
for i in range(0, 1000):
    time = i * timestep
    if i % 10 == 0 or i < 5:
        print(f'Time: {time:5.2f}')
        print(f'{balls}')

    for b in balls:
        b.step(timestep)
```

Se nästa sida!

Inledningen för utmatningen från programmet ska se ut som nedan. Värdena från din lösning ska stämma exakt.

```
Time:  0.00
[FallingBall(10.0000, 0.0000), BouncingBall(10.0000, 10.0000)]
Time:  0.01
[FallingBall(9.9990, -0.0981), BouncingBall(10.0998, 9.9804)]
Time:  0.02
[FallingBall(9.9971, -0.1962), BouncingBall(10.1994, 9.9608)]
Time:  0.03
[FallingBall(9.9941, -0.2941), BouncingBall(10.2988, 9.9411)]
Time:  0.04
[FallingBall(9.9902, -0.3917), BouncingBall(10.3980, 9.9215)]
Time:  0.10
[FallingBall(9.9464, -0.9675), BouncingBall(10.9892, 9.8038)]
```

I vår enkla simulering har fallande bollar bara y-koordinater och rullande bollar (på en plan yta) bara x-koordinater. De utsätts för gravitation från jorden. I initieringsmetoden får båda en position (i y-led respektive x-led, positiva y uppåt, positiva x höger), en hastighet (i y-led respektive x-led) och en luftmotståndskoefficient respektive en friktionskoefficient.

Skriv början på definitionen av de båda klasserna med initieringsmetoder som lagrar nödvändig information i olika attribut.

B4. Skriv metoderna `__repr__` och `__str__` för båda klasserna. De ska returnera strängar som beskriver objektets tillstånd, med utseende motsvarande `FallingBall(y, v)` och `RollingBall(x, v)` där `x`, `y` och `v` är värden för det aktuella objektet. `__repr__` och `__str__` för respektive klass ska alltså returnera samma sak, men se till att inte duplicera kod i onödan.

a) Skriv de båda metoderna för `FallingBall`.

b) Skriv de båda metoderna för `RollingBall`.

B5, B6. I följande två uppgifter ska vi implementera metoden `step` som givet ett tidssteg Δt (i sekunder) uppdaterar objektets hastighet och position för båda klasserna. Allmänt ska metoderna beräkna accelerationen a och uppdatera v enligt $v_1 = v_0 + a\Delta t$. Positionen ska sedan uppdateras enligt $p_1 = p_0 + v_1\Delta t$.

Det krävs dock vissa extra försiktighetssteg för att vår simulering inte ska bli orimlig med denna mycket enkla tidsstegning. När den fallande bollen når marken ska den, till synes momentant, studsa uppåt i stället, med hela farten bevarad (en så kallad elastisk stöt). Fart nedåt byter tecken till fart uppåt.

Den rullande bollen bromsas av friktion. Om man är oförsiktig kan det få hastigheten att byta tecken upprepade gånger. Se till att din boll i stället stannar helt och hållet om friktionen är tillräckligt stor för att föra hastigheten till 0 inom ett tidssteg.

Tänk på att även om du inte kan räkna igenom en lång tidsstegning för hand kan exemplet på utmatning ovan användas för att kontrollera delar av korrekt beteende, som positivt/negativt tecken på olika värden, ungefär i vilken storleksordning de ska vara, om de växer snabbare eller långsammare över tid o.s.v.

Se nästa sida!

B5. Implementera `FallingBall.step`.

Det finns hela tiden en nedåtriktad (negativa y) tyngdacceleration 9.81m/s^2. Det betyder att farten nedåt under en sekund ökar med 9.81m/s. På exempelvis 0.1s ökar den med 0.981m/s.

Dessutom finns ett luftmotstånd. Accelerationen från luftmotståndet i vår modell är beroende av dels *kvadraten* på den nuvarande hastigheten och dels en luftmoståndskoefficient, som vi angav i initeringsmetoden. Luftmoståndet är förstås alltid riktat åt motsatt håll relativt bollens nuvarande hastighet. När bollen är på väg uppåt bromsas den nedåt och vice versa.

Se till att din `step`-metod får bollen att studsa med bevarad fart på ett rimligt sätt varje gång den träffar $y = 0$. Motivera hur du gör detta.

Ledning: Förutom studsen kan accelerationen som verkar på bollen alltså beskrivas som $a = -9.81 - fv^2 \operatorname{sgn}(v)$, där f är luftmotståndskoefficienten, funktionen $\operatorname{sgn}(v)$ är -1 om $v < 0$ och 1 annars. Funtionen sgn finns inte inbyggd i Python.

B6. Implementera `RollingBall.step`.

Bollen rullar med en initial hastighet åt något håll.

Den bromsas kontinuerligt av friktion.

Friktionsaccelerationen är proportionell mot normalaccelerationen och en friktionskoefficient. På ett plant underlag, som i vår uppgift, är normalaccelerationen likvärdig med tyngdaccelerationen, alltså 9.81m/s. Friktionsaccelerationen är givetvis motriktad den nuvarande hastigheten.

Se till att din `step`-metod får bollen att stanna till hastighet exakt 0 när friktionen har bromsat den tillräckligt – det är orimligt att friktionen får bollens rörelse att byta riktning. Motivera hur du gör detta.

Ledning: Accelerationen som verkar på bollen kan alltså beskrivas som $a = -9.81f \operatorname{sgn}(v)$, där f är friktionskoefficienten.

Svar till B1:

Rättning:

Svar till B2:

Rättning:

Svar till B3:

Rättning:

Svar till B4:

Rättning:

Svar till B5:

Rättning:

Svar till B6:

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: