HÖGSKOLAN I HALMSTAD

Tentamen i programmering DI2006 2020-04-15

Tid: 8:00-11:00

Lärare: Johan Thunberg, 035-167273

Anvisningar

- **Tentan skickas in senast klockan 11:00 på blackboard under fliken ”Extra tenta”. Inskickningsmöjlighete stänger klockan 11:00. Ni kan skicka in hur många gånger som helst och sista försöket är det som räknas. Inga undantag görs för sen inlämning, så skicka in i tid!**
- Svaren sparas i ett dokument (text, word eller pdf).
- Det är tillåtet att använda en Pythoninstallation, men det är inte tillåtet att kopiera lösningar från andra studenter eller internet. Det är heller inte tillåtet att på npgot sätt samarbeta med andra studenter (eller individer). Svaren kommer att undersökas för plagiering.
- Vid varje uppgift anges hur många poäng som maximalt utdelas för uppgiften.
- Denna tentamen består av två delar: **Del 1** och **Del 2**. På Del1 kan man få maximalt 30 poäng. På Del 2 kan man få maximalt 20 poäng.
- För betyg 3 gäller det att få minst 15 poäng på Del 1 (Del 2 behöver alltså inte göras för att få detta betyg).
- För betyg 4 gäller det att
    - få minst 15 poäng på Del1 och minst 8 poäng på Del 2,
    - samt muntligen examineras senare idag på frågorna i Del 2 samt relaterade frågor, varefter en samlad bedömning avgör betyget.
- För betyg 5 gäller det att
    - få minst 15 poäng på Del1 och minst 13 poäng på Del 2,
    - samt muntligen examineras senare idag på frågorna i Del 2 samt relaterade frågor varefter en samlad bedömning avgör betyget.
- Om man inte är intresserad av ett högre betyg så avstår man från att ansluta till den muntliga redovisningen. Anslut i tid om i vill examineras muntligt. Om ni ansluter till mötet sent, finns mindre tid för att göra en bedömning.

Del I (Denna del består av sex uppgifter)

Uppgift 1 (6p)

a) Betrakta följande: `y = x%2 + x%1`. För vilka heltal `x` gäller det att `y = 0`? För vilka heltal `x` gäller det att `y = 1`? För vilka heltal `x` gäller det att `y = 2`?

b) Betrakta följande uttryck: `not (x or False) and (y == "ja")`. Ange vad `x` respektive `y` ska vara för att uttrycket ska bli `True` och förklara varför.

c) Betrakta följande uttryck: `(2 A 3) or (2 B 2)`. **A** och **B** ska bytas ut mot jämförelseoperatorer sådana att uttrycket blir `False`. Uppgiften är att ange alla sådana jämförelseoperatorer istället för **A** och **B** så att uttrycket blir `False`.

Uppgift 2 (6p)

a) Betrakta följande programkod:

```
x = int(input("Ange x:"))
value = 0
for i in range(x):
    value += i
print(value)
```

Skriv ett nytt program som fungerar på samma sätt som detta, men inte använder sig av for-loop(ar). Utskriften på skärmen ska bli densamma för det nya programmet som för det gamla programmet för samma val av input.

b) Betrakta följande program:

```
x = int(input("Ange x:"))
value = 0
i = 0
j = 0
while i < x:
    i += 1
    while j < i:
        j += 1
        value += j
    j = 0
print(value)
```

Skriv ett nytt program som fungerar på samma sätt som detta, men inte använder sig av while-loop(ar). Utskriften på skärmen ska bli densamma för det nya programmet som den för det gamla programmet för samma val av input.

c) Betrakta programmet nedan där vissa delar har dolts. Ange vad som ska stå bakom de dolda delarna A, B, C och D, så att utskriften på skärmen vid körning av programmet blir: `02468101214161820` (Obs! det finns olika lösningar på detta problem, det räcker med att ge en.)

```
string = A
for i in range( B ):
    if  C  == 0:
        string +=  D
print(string)
```

Uppgift 3 (6p)

a) Betrakta funktionerna `bar` och `foo` nedan. Finns det ett värde på `value` som `foo` inte hanterar på ett bra sätt och, mer precist, fastnar i en oändlig loop. Ange i sådana fall detta värde, samt modifiera koden i `foo` så att detta hanteras på ett lämpligt sätt.

```
def bar(value):
    return value**2

def foo(value):
    while True:
        if value > 10:
            break
        else:
            value += bar(value)
    return value
```

b) Skriv en funktion som tar som argument en lista. Funktionen adderar sedan elementen i listan och returnerar denna summa. För att få full poäng på denna uppgift krävs att man hanterar fel som uppstår när elementen i listan är av olika typ (t.ex. när ett element är ett heltal och ett annat är en sträng).

Uppgift 4 (4p)

Betrakta följande program som innehåller funktionerna `foo` och `bar`.

```
def foo(arg1, arg2):
    f = open(arg1,"w")
    for i in range(arg2):
        for j in range(i):
            f.write("1")
        f.write("\n")
    f.close()

def bar(arg1, arg2):
    foo(arg1, arg2)
    count = 0
    f = open(arg1,"r")
    for line in f:
        count += len(line) - 1 #?
    f.close()
    return count

print(bar("test", 4))
```

a) Vad gör funktionerna `foo` och `bar`? Beskriv övergripande och kortfattat.
b) Vad händer om vi tar bort ”`- 1`” på raden som avslutas med ”`#?`”
c) Vad skrivs ut på skärmen när programmet körs?

Uppgift 5 (4p)

Betrakta följande program:

```
def foo(string1, string2):
    var1 = set(string1)
    var2 = set(string2)
    var3 = var1 & var2
    var4 = {}
    for i in var3:
        var4[i] = 0
        for j in string1:
            if j == i:
                var4[i] += 1
        for j in string2:
            if j == i:
                var4[i] += 1
    return var4

print(foo("hej", "hej hej hejsan"))
```

a) Ange (inte rad för rad utan övergripande) vad funktionen `foo` gör.
b) Vad skrivs ut på skärmen när programmet körs?

Uppgift 6 (4p)

Betrakta exemplet nedan där en klass återfinns som representerar lådor (boxar). Vissa delar har dolts. Ersätt de dolda delarna A, B och C med kod så att när programmet körs så skrivs följande ut på skärmen: Boxen har volymen: 60

```
class Box:
    def __init__(self, height,
    width, depth):
        [A]

    def compute_volume(self):
        return [B]

    def __str__(self):
        return "Boxen har volymen:" [C]

box = Box(3,4,5)
print(box)
```

Del 2 (denna del består av en uppgift med tre deluppgifter)

Uppgift 7.

I denna uppgift ska du simulera en spindel som klättrar upp på en stege. Så fort spindeln kommer till ett nytt steg på stegen kan den ramla ner till marken igen med en sannolikhet p mellan 0 och 1, eller fortsätta uppåt till nästa steg med sannolikheten 1-p. Stegen innehåller N steg. Om spindeln når det översta steget på stegen så ramlar den ner automatiskt.

a) **(3p)** Skriv en funktion `spindel(p, N)` som returnerar ett trappsteg (representerat som ett heltal) som en simulerad (slumpmässig) spindel ramlar ner ifrån, där `p` och `N` är förklarade ovan. Marken är steg 0, sen följer steg 1, steg 2, …, steg `N`.

b) **(5p)** Skriv en funktion `simulation(p, N, antal_spindlar)` som simulerar antal_spindlar många spindlar som klättrar uppför stegar (och ramlar ner). Funktionen använder sig av funktionen `spindel` och returnerar följande tre saker:
   a. det högsta steg som någon spindel kom till,
   b. medelvärdet av de steg från vilka spindlarna ramlade ner.
   c. Det totala antalet steg som spindlarna klättrade tillsammans.

c) **(12p)** Nu ska du skriva en klass `Spider_climb` som används för mer avancerade simuleringar än de ovan. För klassen gäller följande:
   a. Man ska kunna lägga till och ta bort spindlar.
   b. Varje spindel har en ett eget antal steg på sin stege, dvs. ett eget N som kan skilja sig från andra spindlars N.
   c. För varje spindel är det olika sannolikheter att ramla ner vid de olika stegen. Dessa sannolikheter kan skilja sig mellan spindlarna.
   d. Varje spindel tillbringar en viss tid på varje trappsteg innan den ramlar ner eller fortsätter. Dessa tider skiljer sig mellan steg och spindlar.
   e. Det ska finnas en metod som returnerar spindlarna.
   f. Det ska finnas en metod som simulerar att spindlarna klättrar upp och ramlar ner för stegarna. För varje spindel ska anges hur många gånger den klättrar upp (och ramlar ner). Resultaten ska sparas som instansvariabler (se g. nedan vilka dessa kan vara).
   g. Det ska finnas en metod som visar resultaten av den senaste simuleringen. Förutom a. b. och c. i uppgift b) ovan, ska även det steg som någon spindel spenderade mest tid på visas, samt den totala tiden som spindlarna spenderade på stegarna.
   h. Det ska finnas en metod som tar bort resultaten från den senaste simuleringen.
   i. Det ska finnas en metod som omvandlar objekt av klassen till en sträng där relevant data om objektet återfinns.

Apendix till tentamina i Programmering DI2006
Python: sammanfattning

Kod	**Förklaring**

`name = "Kalle"`
`age = 10`
`pi = 3.14`
`savings= BankAccount()`
	En variabel av **typen string**
	En variabel av **typen int**
	En variabel av **typen float**
	En variabel av typen BankAccount

`print(y)` `print("Hej")`	**Skriver ut** värdet av variabeln y. Skriver ut "Hej"
`x = input("Namn: ")`	**Skriver ut**: "Namn: " och **pausar tills användaren skriver in** något och trycker på Enter. **Lagrar sedan** texten användaren skrev in, i variabeln x.

`x = 15`
`if x < 10:`
`  s = "x < 10"`
`  print(s)`
`elif x < 20:`
`  s = "x < 20"`
`  print(s)`
`elif x == 20:`
`  print("x = 20")`
`else:`
`  print("else:")`
`  print("x > 20")`
	Villkorssatser.
	
	if testar om ett villkor är uppfyllt.
	
	elif hänger ihop med if och testas bara om testet i if misslyckas.
	
	else hänger också ihop med if och testas bara om det i if och eventuella elif misslyckas.

`age == 100`	Exempel på villkor man kan testa. **Kom ihåg: == INTE =** när man vill testa ett villkor!
`x < 100 and y < 100`	Exempel på villkor man kan testa. Testar om både variabeln x **och** variabeln y är mindre är 100.

for tal **in** range(100): print(tal)	**for loop** För varje tal från 0 till och med 99: Skriv ut talet.
for num **in** [1, 6, 2, 5]: print(num)	För varje nummer i listan [1, 6, 2, 5]: Skriv ut numret.

var = 10
while var < 100:
  print(var)
  var = var + 1
	while loop
	Medans villkoret
	var < 100 är uppfyllt, gör följande om och om igen:
	- Skriv ut värdet av var
	- Öka var med ett .

def funktionsnamn(a, b):
  r = a*b + 10
  r = r/10
  r = r – 10
  return r
	Funktionsdefinition. Denna kod körs först då funktionen anropas. a och b kallas för **parametrar** och är variabler utan värde som kan användas inuti funktionen.

funktionsnamn(1, 2) x = funktionsnamn(2, 5)	**Anrop** av funktionen *funktionsnamn* kör koden i funktionsdefinitionen. **Argumenten** 1 och 2 är värden som lagras i parametrarna i funktionsdefinitionen

fil=open("filnamn", "r")
 	
for ord in fil:
  //gör något
 	
text=fil.read()
line=fil.readline()
	Öppna en fil och läser den ord för ord. w- skriv , a-append
	
	läser en fil till variabel text
	
	läser en rad från fil till variabel line

lista=[] for var in lista	Skapa tom lista Iterera genom lista

Skapa klasser och objekt

```
class BankAccount:

     def __init__(self, iname, ibalance):
          self.__name=iname
          self.__balance=ibalance

     def deposit(self, amount):
          balance=balance-amount
```

```
savigs= BankAcount(”kalle”, 100)    // skapa objekt
```

Användbara funktioner:

```
int(a)  // koverterar a från String till int
str(a)  // konverterar a från int till String
ord(a)  // konverterar a från char till int
chr(a)  // konverterar a från int till char

random.randint(a,b) // slumpar tal mellan a och b

text.split(delimitator)  Dela en sträng i en lista av ord.
Argumenten delimitator används som separator. Metoden returnerar
en lista.
```

Andra String metoder:

Table 8-1 Some string testing methods

Method	Description
`isalnum()`	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
`isalpha()`	Returns true if the string contains only alphabetic letters and is at least one character in length. Returns false otherwise.
`isdigit()`	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
`islower()`	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
`isspace()`	Returns true if the string contains only whitespace characters and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`).
`isupper()`	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.

Table 8-3 Search and replace methods

Method	Description
`endswith(substring)`	The *`substring`* argument is a string. The method returns true if the string ends with *`substring`*.
`find(substring)`	The *`substring`* argument is a string. The method returns the lowest index in the string where *`substring`* is found. If *`substring`* is not found, the method returns –1.
`replace(old, new)`	The *`old`* and *`new`* arguments are both strings. The method returns a copy of the string with all instances of *`old`* replaced by *`new`*.
`startswith(substring)`	The *`substring`* argument is a string. The method returns true if the string starts with *`substring`*.

List metoder

Table 7-1 A few of the list methods

Method	Description
`append(item)`	Adds *item* to the end of the list.
`index(item)`	Returns the index of the first element whose value is equal to item. A `ValueError` exception is raised if item is not found in the list.
`insert(index, item)`	Inserts *item* into the list at the specified *index*. When an item is inserted into a list, the list is expanded in size to accommodate the new item. The item that was previously at the specified index, and all the items after it, are shifted by one position toward the end of the list. No exceptions will occur if you specify an invalid index. If you specify an index beyond the end of the list, the item will be added to the end of the list. If you use a negative index that specifies an invalid position, the item will be inserted at the beginning of the list.
`sort()`	Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
`remove(item)`	Removes the first occurrence of *item* from the list. A `ValueError` exception is raised if item is not found in the list.
`reverse()`	Reverses the order of the items in the list.

Dictionary metoder:

Table 9-1 Some of the dictionary methods

Method	Description
`clear`	Clears the contents of a dictionary.
`get`	Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
`items`	Returns all the keys in a dictionary and their associated values as a sequence of tuples.
`keys`	Returns all the keys in a dictionary as a sequence of tuples.
`pop`	Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
`popitem`	Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.
`values`	Returns all the values in the dictionary as a sequence of tuples.