Tentamen i Programmering DI2006

Datum: 2022-05-25
Tid: 9.00–13.00
Ansvarig lärare: Eric Järpe (tel: 0729-77 36 26, email: eric.jarpe@hh.se)

Anvisningar

- Tillåtna hjälpmedel är
  - formelsamling (som är häftad till tentamenstexten)
  - miniräknare TI-30Xa (Texas Instruments)
  - skrivpapper
  - penna
  - suddigummi
  - linjal
  - frukt, fika
- Till varje uppgift finns angivet hur många poäng som maximalt utdelas för uppgiften.
- Tentamen består av två delar: **Del 1** och **Del 2**.
- Samtliga frågor i Del 1 ska besvaras i den svarstalong som är bifogad med tentamenstexten.
- Frågorna i Del 2 ska besvaras på vanligt separat rutat papper.
- Då programkod anges som svar ska den vara i så körbart skick som möjligt.
- Del 1 består av 24 frågor och här kan man maximalt få 30 poäng.
- Del 2 består av 3 frågor och här kan man maximalt få 20 poäng.
- För betyg 3 krävs minst 15 poäng på Del 1. (Del 2 behöver inte alls göras för att få detta betyg.)
- För betyg 4 krävs minst 15 poäng på Del 1 och 5 poäng på Del 2.
- För betyg 5 krävs minst 15 poäng på Del 1 och 10 poäng på Del 2.

LYCKA TILL!

Del 1

Flervalsfrågor

1. Komplettera påståendet så att det blir korrekt:
   "Python är en slags _____ som använder en _____". (1p)
   (a) högnivåspråk, kompilator
   (b) högnivåspråk, interpretator
   (c) lågnivåspråk, kompilator
   (d) lågnivåspråk, interpretator
   (e) maskinkod, kompilator
   (f) maskinkod, kompilator
   (g) Inget av de ovanstående alternativen

---

2. Komplettera påståendet så att det blir korrekt:
   "En variabel vid programmering finns vanligen i _____ i datorn. (1p)
   (a) minnet
   (b) tangentbordet
   (c) skärmen
   (d) nätaggregatet
   (e) batteriet
   (f) judkortet
   (g) Inget av de ovanstående alternativen

---

3. Vilket av de följande uttrycken är *inte* ett korrekt uttryck i Python? (1p)
   (a) `67-89/34`
   (b) `"Ålder" + "Vikt"`
   (c) `True or False`
   (d) `list(range(4))`
   (e) `43 -> 34`
   (f) `x = 73%37`
   (g) Inget av de ovanstående alternativen

---

4. För att få reda på sifferkoden (ASCII-värdet) för en variabel `a` kan man skriva(1p)
   (a) `int(a)`
   (b) `str(a)`
   (c) `input(a)`
   (d) `chr(a)`
   (e) `ord(a)`
   (f) `num(a)`
   (g) Inget av de ovanstående alternativen

---

5. Vad kallas ett uttryck som antingen får värdet `True` eller värdet `False`? (1p)
   (a) Logiskt
   (b) Booleskt
   (c) Sträng
   (d) Numeriskt
   (e) Vektorvärt
   (f) Hexadecimalt
   (g) Inget av de ovanstående alternativen

---

6. I objektorienterad programmering kan man "gömma" funktioner i en klass så att de inte blir möjliga att anropa då man använder klassen. Denna gömning görs genom att funktionsnamnet börjar med (1p)
   (a) _ (dvs en underscore)
   (b) __ (dvs två underscores)
   (c) ___ (dvs tre underscores)
   (d) . (dvs en punkt)
   (e) .. (dvs två punkter
   (f) ... (dvs tre punkter)
   (g) Inget av de ovanstående alternativen

---

7. Vid objektorienterad programmering ska man i definitionen av en klass alltid ha med ett extra argument till funktionerna. Detta argument anges inte sedan då funktionerna definierade inom klassen används utanför kalssen. Vad kallas detta extra argument? (1p)
   (a) `void`
   (b) `x`
   (c) `null`
   (d) `none`
   (e) `pointer`
   (f) `self`
   (g) Inget av de ovanstående alternativen

---

8. Då man anropar en funktion i Python och vill ändra argumentens ordning i förhållande till hur de förekommer vid definitionen av funktionen kan man använda (1p)
   (a) nyckelordsargument
   (b) positionella argument
   (c) valbara argument
   (d) horisontella argument
   (e) formella arguement
   (f) dolda argument
   (g) Inget av de ovanstående alternativen

---

9. I vilken del av en dator sparas filerna? (1p)
   (a) I RAM-minnet
   (b) I CPU:n
   (c) På moderkortet
   (d) På hårddisken
   (e) I grafikkortet
   (f) I adaptern
   (g) Inget av de ovanstående alternativen

---

10. Vilket kommando i Python öppnar *inte* filen `output.txt` för att skriva till den?(1p)
    (a) `f = open("output.txt","wb")`
    (b) `f = open("output.txt","ab")`
    (c) `f = open("output.txt","r")`
    (d) `f = open(output.txt,"x")`
    (e) `f = open(output.txt,"xt")`
    (f) `f = open(output.txt,"w")`
    (g) Inget av de ovanstående alternativen

---

11. Låt `L` vara en lista i Python med värdet `[2,7,3,4]`. På vilket sätt kan man *inte* ta bort sista elementet i `L`? (1p)
    (a) `del L[-1]`
    (b) `L.remove(4)`
    (c) `L.save(0:3)`
    (d) `L.pop()`
    (e) `L = L[:-1]`
    (f) `L = [L[i] for i in range(4)]`
    (g) Inget av de ovanstående alternativen

---

12. Antag att `s` är en sträng i Python. Vilket av följande kommandon får värdet `True` om alla tecken i `s` är stora bokstäver (versaler) eller alla tecken är siffror? (1p)

    (a) `s.islower() or s.isdigit()`

    (b) `s.isupper() and s.isdigit()`

    (c) `s.islower() or s.isalpha()`

    (d) `s.isupper() and s.isspace()`

    (e) `s.islower() and s.isupper()`

    (f) `s.isupper() or s.isdigit()`

    (g) Inget av de ovanstående alternativen

---

13. Vid objektorienterad programmering i Python är *objekt* en samling (1p)

    (a) variabler och attribut

    (b) metoder och funktioner

    (c) metoder och attribut

    (d) variabler och klasser

    (e) klasser och arv

    (f) överklasser och underklasser

    (g) Inget av de ovanstående alternativen

---

14. Rekursiva funktioner lämpar sig väl för att definiera (1p)

    (a) utskrift av heltalen `1` till `N`

    (b) utskrift av heltalen `N` till `1`

    (c) generering av Fibonaccitalen (dvs den talföljd som börjar med `x[0]=0` och `x[1]=1` och sedan uppflyller villkoret `x[i+2]=x[i+1]+x[i]` för alla index `i=0,1,2,3,...`)

    (d) beräkning av kvadratrötter (dvs tal `x` sådana att `x*x==q` för ett givet tal `q`)

    (e) generering av slumptal då man importerat modulen `random`

    (f) utskrift av konsonanterna i alfabetet

    (g) Inget av de ovanstående alternativen

Skrivfrågor

15. Hur tilldelar man variabeln `A` värdet av en mängd med elementet `1` i Python? (2p)

---

16. Ange på två rader en funktion som tar tre tal som argument och returnerar det minsta av dem. (2p)

---

17. Följande funktion ska ta en sträng, `s`, som argument och returnera en sträng enbart bestående av tecknet `x` och det ska vara lika många som det är i strängen `s`:

```
def f(s):
  return __________
```

Vad måste stå på den tomma raden? (2p)

---

18. Hur kan man, utan att importera några moduler, skriva en funktion som tar en lista av tal från 1 till 7 som argument och skriver ut hur många av varje sort det är. T.ex. ska listan `[3,3,5,2,3,1,7]` resultera i utskriften: (3p)

```
1: 1
2: 1
3: 3
4: 0
5: 1
6: 0
7: 1
```

---

19. Skriv ett program som efterfrågar ett heltal, `n`. Sedan ska funktionen generera `n` st slumpmässiga heltal mellan `1` och `10` och skriva ut hur många av dessa som var jämnt delbara med 3. (3p)

---

20. I en funktion finns risken att ett tal `x` får värdet `0`. I ett avsnitt har man koden:

```
___:
  if 1/x<0.5:
    y=1+x
_____ ZeroDivisionError:
____
```

Komplettera denna kod på de tre tomma raderna så att man inte får något felmeddelande och så att det inte händer något alls om `x` skulle råka ha värdet `0`. (4p)

Del 2

Programmeringsuppgifter

21. *Portvakten*
Vid ett nätverk har man en server som avgör vilka som är behöriga att få tillträde genom en inloggningsprocedur. För detta ändamål har man en variabel `dict` av typen *dictionary* som innhåller alla användarnamn som *keys* och hashvärdet av deras lösenord som motsvarande *values*. Antag att `dict` är global variabel i denna uppgift. Eventuella moduler som behöver användas ska importeras.

(a) Skriv funktionen `hash` som tar en sträng `s` och returnerar en lista, `h` (hashvärdet), av heltal, dvs element av typen *integer*. Dessa heltal ska, tecken för tecken i strängen `s`, beräknas enligt

$$10^5[\log(n(n+3)) \bmod 1]_5$$

där n är tecknets ASCII-värde, log innebär att man ska beräkna logaritmen, mod 1 betyder att man ska behålla decimaldelen av talet (dvs de siffror som kommer efter decimaltecknet) och $[\ldots]_5$ betyder att man ska behålla de 5 första decimalerna i talet mellan hakparenteserna. Detta innebär att $10^5[\ldots]_5$ blir ett heltal. (4p)

(b) Skriv funktionen `get_passwd` som anger att användaren ska skriva sitt användarnamn (dvs `Username:`), läser in svaret till variabeln `user` och sedan anger att användaren ska ange sitt lösenord (dvs `Password:`) och *inte* ger en variabel detta värde utan istället läser ger variabeln `passwd` hashvärdet (beräknat med hjälp av funktionen `hash`) av lösenordet. (3p)

(c) Slutligen ska funktionen `check` ta variablerna `user` och `passwd` som argument och kontrollera om det finns någon *key* i `dict` som har värdet `user`. Om det inte finns ska funktionen ge utskriften

```
Användarnamnet eller lösenordet är fel!
```

Om det finns i `dict` ska motvarande *value* ha värdet av `passwd`. Om det inte matchar ska funktionen ge utskriften

```
Användarnamnet eller lösenordet är fel!
```

Om det matchar ska funktionen svara `Välkommen` *användarnamnet*`!` (2p)

22. *Minigma*

Simon Singh har sagt att *krypteringssystemet Enigma hör till de mest fruktade genom historien.* Det bestod av många kluriga steg som gjorde det mycket svårt att knäcka. Ett enklare system som dock har många likheter med Enigma kan åstadkommas genom att kombinera ett steg av *subsitution* och ett av *rotation.* Du ska i denna uppgift skriva ett program som genomför kryptering enligt just den principen. Programmet ska bestå av tre funktioner: `plain_in`, `encrypt` och `crypt_out`.

(a) Skriv funktionen `plain_in` som ska läsa in en text från filen `klartext.txt` och sedan omvandla den inlästa texten till en lista av *characters* som är rensad från allt utom stora och små bokstäver (utom å,ä,ö), siffror, ordmellanrum och punkter. Alla stora bokstäver omvandlas till motsvarande små. Låt oss ta ett exempel:
Antag nu att klartexten
`The owls are not what they seem`
läses in från filen `klartext.txt`. Ut från `plain_in` ska då listan `["t","h","e"," ","o","w","l","s"," ","a","r","e"," ","n","o","t"," ","w","h","a","t"," ","t","h","e","y"," ","s","","e","e","m"]` komma. (4p)

(b) Sedan följer funktionen `encrypt` som ska implementera krypteringen. Denna ska gå till så att varje bokstav från klartextalfabetet bestående av 38 tecken: `abcdefghijklmnopqrstuvwxyz0123456789 .` (observera att det näst sista är ordmellanrum). Dessa ska systematiskt bytas mot tecknen i ett kryptoalfabet som är samma tecken men i en slumpmässig ordning. Emellertid ska kryptoalfabetet även *roteras* ett steg för varje nytt tecken ur klartexten som ska krypteras. Tillbaks till exemplet:
I funktionen `encrypt` slumpas ett kryptoalfabet fram. Antag att det blir t.ex.
Klartextalfabet: `abcdefghijklmnopqrstuvwxyz0123456789 .`
Kryptoalfabet: `ylg84m.jrs7bkoh5tez2xd9uwp nci13af0qv6`
Då ska `t` (första tecknet ur klartexten) bytas mot `2`. Därefter skiftas kryptotexten ett steg, dvs sista tecknet hamnar först, första tecknet hoppar till andra plats, andra tecknet hoppar till tredje, osv, så att vi får:
Klartextalfabet: `abcdefghijklmnopqrstuvwxyz0123456789 .`
Kryptoalfabet: `6ylg84m.jrs7bkoh5tez2xd9uwp nci13af0qv`
Därmed ska andra tecknet ur klartexten, `h`, bytas mot `.` varefter kryptoalfabetet skiftas ett snäpp framåt igen:
Klartextalfabet: `abcdefghijklmnopqrstuvwxyz0123456789 .`
Kryptoalfabet: `v6ylg84m.jrs7bkoh5tez2xd9uwp nci13af0q`
Tredje tecknet ur klartexten, `e`, blir `g`, osv. (5p)

(c) Den tredje funktionen, `crypto_out`, ska skriva ut hela kryptotextens tecken som en enda sträng (inte som en lista av tecken) till filen `kryptotext.txt`. I exemplet innebär det att texten `2.gf7embcij3pyy4dmn265q9z67ihox` skrivs till filen `crypto_out.txt`. (2p)

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.

Table 13-2 Some of the optional arguments to the `create_line` method

Argument	Description
`arrow=value`	By default, lines do not have arrowheads, but this argument causes the line to have an arrowhead at one or both ends. Specify `arrow=tk.FIRST` to draw an arrowhead at the beginning of the line, `arrow=tk.LAST` to draw an arrowhead at the end of the line, or `arrow=tk.BOTH` to draw arrowheads at both ends of the line.
`dash=value`	This argument causes the line to be a dashed line. The value is a tuple, consisting of integers, that specifies a pattern. The first integer specifies the number of pixels to draw, the second integer specifies the number of pixels to skip, and so forth. For example, the argument `dash=(5, 2)` will draw 5 pixels, skip 2 pixels, and repeat until the end of the line is reached.
`fill=value`	Specifies the color of the line. The argument's value is the name of a color, as a string. There are numerous predefined color names that you can use, and Appendix D shows the complete list. Some of the more common colors are `'red'`, `'green'`, `'blue'`, `'yellow'`, and `'cyan'`. (If you omit the `fill` argument, the default color is black.)
`smooth=value`	By default, the `smooth` argument is set to `False`, which makes the method draw straight lines connecting the specified points. If you specify `smooth=True`, the lines are drawn as curved splines.
`width=value`	Specifies the width of the line, in pixels. For example, the argument `width=5` causes the line to be 5 pixels wide. By default, lines are 1 pixel wide.

Table 13-4 Some of the optional arguments to the `create_oval` method

Argument	Description
`dash=value`	This argument causes the outline of the oval to be a dashed line. The value is a tuple, consisting of integers, that specifies a pattern. The first integer specifies the number of pixels to draw, the second integer specifies the number of pixels to skip, and so forth. For example, the argument `dash=(5, 2)` will draw 5 pixels, skip 2 pixels, and repeat until the end of the line is reached.
`fill=value`	Specifies a color to fill the oval with which. The argument's value is the name of a color, as a string. There are numerous predefined color names that you can use, and Appendix D shows the complete list. Some of the more common colors are `'red'`, `'green'`, `'blue'`, `'yellow'`, and `'cyan'`. (If you omit the `fill` argument, the oval will not be filled.)
`outline=value`	Specifies the color of the oval's outline. The argument's value is the name of a color, as a string. There are numerous predefined color names that you can use, and Appendix D shows the complete list. Some of the more common colors are `'red'`, `'green'`, `'blue'`, `'yellow'`, and `'cyan'`. (If you omit the `outline` argument, the default color is black.)
`width=value`	Specifies the width of the oval's outline, in pixels. For example, the argument `width=5` causes the line to be 5 pixels wide. By default, lines are 1 pixel wide.