Tentamen i Programmering DI2006

Datum: 2022-04-20
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!

Svartsalong till Del 1

Svara med ett kryss i *ett* av alternativen (a)–(g) för var och en av frågorna 1–14.
Besvara frågorna 15–24 med text på de angivna raderna.

	Svarsalternativ						
Fråga	(a)	(b)	(c)	(d)	(e)	(f)	(g)
1.							
2.							
3.							
4.							
5.							
6.							
7.							
8.							
9.							
10.							
11.							
12.							
13.							
14.							

15. _____

_____

16. _____

17. _____

18. _____

19. _____ 20. _____

21. _____ 22. _____

23. _____ 24. _____

Namn/anonymitetskod: _____

Del 1

1. Om man vill ha utskriften `"Hej!"` kan man köra kommandot (1p)
   (a) `print(Hej!)`
   (b) `print("Hej!")`
   (c) `print("'Hej!'")`
   (d) `print('"Hej!"')`
   (e) `print("""Hej!""")`
   (f) `cat("Hej!")`
   (g) Inget av de ovanstående alternativen

---

2. Om man vill undvika att en rad exekveras då man kör ett program kan man kommentera bort raden genom att låta den (1p)
   (a) börja med tecknet `;`
   (b) börja med `(*` och sluta med `*)`
   (c) börja med tecknet `#`
   (d) börja med tecknet `%`
   (e) börja med `<--` och sluta med `-->)`
   (f) sluta med `<comment>`
   (g) Inget av de ovanstående alternativen

---

3. Vad svarar Python om man kör koden `33//3**2/(15%9)`? (1p)
   (a) `0.5`
   (b) `1.3`
   (c) `5.0`
   (d) `100.7`
   (e) `6.66666666666666667`
   (f) `0`
   (g) Inget av de ovanstående alternativen

---

4. Antag att `x` är en lista med 7 element. För att ange det tredje, fjärde och femte elementet skrivs (1p)
   (a) `x[2,4]`
   (b) `x[3,4,5]`
   (c) `x[3:5]`
   (d) `x[3:4:5]`
   (e) `x[2:3:4]`
   (f) `x[2:5]`
   (g) Inget av de ovanstående alternativen

---

5. Vad svarar Python om man kör följande kod? (1p)

```
a=3
b=4
if a>b**0:
  if a**b<b**a:
    b=a-1
    a=b+a
    print(a*b)
  elif b+a<a*b:
    b=a+1
    a=b-a
    print(a*b)
  else:
    b=a
    a=b//a
    print(a*b)
else:
  print(a/b)
```

(a) `3`

(b) `4`

(c) `7`

(d) `10`

(e) `-1`

(f) `0.75`

(g) Inget av de ovanstående alternativen

---

6. Vad svarar Python om man kör följande kod? (1p)

```
class A:
  def __init__(self,v):
    self.__b=v+1

a=A(2)
print(a.__b)
```

(a) `0`

(b) `1`

(c) `3`

(d) `AttributeError`

(e) `TypeError`

(f) `SyntaxError`

(g) Inget av de ovanstående alternativen

---

7. Låt `d` vara en ordlista (dictionary) med keys: `'a'`, `'b'` och `'c'` med respektive values: `1`, `2` och `3`. Om man skriver `d.pop(1,2)` svarar Python (1p)

   (a) `0`

   (b) `1`

   (c) `2`

   (d) `{'b':2, 'c':3}`

   (e) `'b'`

   (f) `TypeError`

   (g) `SyntaxError`

   (h) Inget av de ovanstående alternativen

---

8. Vad kallas en översiktlig presentation av ett program där olika programsteg anges med olika boxar av olika geometrisk form med text i och pilar mellan boxarna?(1p)

   (a) Flödesschema

   (b) Pseudokod

   (c) Hierarki

   (d) IDE

   (e) Algoritm

   (f) Detaljplan

   (g) Inget av de ovanstående alternativen

---

9. Vad kallas företeelsen när en variabel tilldelas ett nytt värde genom att man exekverar en funktion trots att funktionen inte returnerar något värde? (1p)

   (a) indirekt evaluering

   (b) implicit tilldelning

   (c) imperativ programmering

   (d) exklamation

   (e) deamon

   (f) sidoeffekt

   (g) Inget av de ovanstående alternativen

---

10. Om `a` har värdet `(1,2)` och `b` värdet `(5,6)` så får `a` värdet `(1,2,5,6)` om man skriver (1p)

    (a) `a = a.append(b)`

    (b) `a = a+b`

    (c) `a = (a,b)`

    (d) `a = a and b`

    (e) `a = a*b`

    (f) `a = ''.join([str(a),str(b)])`

    (g) Inget av de ovanstående alternativen

---

11. Hur skrivs siffran 37 då den lagras som en byte? (1p)

(a) `trettiosju`

(b) `00111010`

(c) `01001011`

(d) `3*10+7*1`

(e) `00100101`

(f) `H68900AD`

(g) Inget av de ovanstående alternativen

---

12. Vad svarar Python om man kör följande kod? (1p)

```
try:
  [0,1][2]
except:
  print("Error")
```

(a) `0`

(b) `1`

(c) `SyntaxError`

(d) `IndexError`

(e) `TypeError`

(f) `Error`

(g) Inget av de ovanstående alternativen

---

13. Vad kan man skriva för att få reda på alla globalt definierade variabler? (1p)

(a) `global_variabels()`

(b) `globVars()`

(c) `global('all')`

(d) `globals()`

(e) `Variabels('Global')`

(f) `Global(True)`

(g) Inget av de ovanstående alternativen

---

14. Lista är ett exempel på en datatyp som är (1p)

(a) mutable

(b) immutable

(c) hårdvara

(d) mjukvara

(e) objektorienterad

(f) låg nivå

(g) Inget av de ovanstående alternativen

---

15. Komplettera vad som ska stå på rad 1 och rad 2 i koden nedan så att användaren instrueras: `Ange gurka eller äpple:`. Sedan ska svaret bli `Grönsak` om användaren valt gurka och `Frukt` om valet var äpple. Om användaren angivit något annat än `gurka` eller `äpple` ska instruktionen ovan upprepas ända tills användaren svarat ett av de två alternativen `gurka` eller `äpple`. (1p)

```
val=""
        rad 1
          rad 2
if val=="gurka":
  print("Grönsak")
if val=="äpple":
  print("Frukt")
```

---

16. Låt den globala variabeln `abra='abrakadabra'` vara given. Betrakta koden till följande funktioner presenterade på numrerade rader:

```
1 def a(x):
2   if x<3:
3     print('g')
4   else:
5     print(abra[x])
6
7 def b(y):
8   if y>7:
9     a(y-6)
10  elif y%2==0:
11    print('f')
12  else:
13    print(abra[y+2])
14
```

I vilken ordning exekveras raderna då man skriver `b(10)`? (2p)

---

17. Ett primtal är ett heltal som inte är jämnt delbart med något annat heltal än 1 och talet självt. Raderna `n = int(input("Ange det tal som ska kontrolleras:"))` och `prim = True` är givna. Skriv nu på en rad kod som gör att variabeln `prim` blir `True` om `n` är ett primtal och `prim` blir `False` om `n` inte är det. (2p)

---

18. Antag att klassen `Tarning` definieras med koden

```
import random as r
class Tarning:
  def __init__(self):
    self.side_up = 1
  def toss(self):
      tom rad
  def get_side_up(self):
    return self.side_up
```

och att man låter variabeln `c` vara av denna klass genom att skriva `c = Tarning()`. Vad ska man skriva på den tomma raden ovan för att metoden `toss` ska ge `c.side_up` ett värde av `1`, `2`, `3`, `4`, `5` och `6`? (2p)

---

19. Låt `A = set('abc')`, `B = set('cde')`, `C = set('acef')`. Ange (på en rad) hur man kan skriva för att beräkna $\texttt{A} \cap (\texttt{B} \cup \texttt{C})$ (1p)

---

20. Vilken värde får `a` om man skriver
`a = [2**i if i**2%3==1 else 1 for i in range(5,10)]`? (2p)

---

21. Antag att man vill lägga till text som produceras i Python till redan befintlig text i filen `text.txt`. Hur måste man då *först* skriva, för att texten `Hej` sedan ska kunna läggas till den övriga med kommandot `f.write("Hej")`? (1p)

---

22. Vad heter det bibliotek som kallas *Python's standard GUI library* och som kan importeras till Python för att konstruera en GUI och producera grafisk output? (1p)

---

23. Vad svarar Python om man kör följande kod? (1p)

```
class A:
  def __init__(self,v=1):
    self.v = v
  def set(self,v):
    self.v = v-1
    return v

a = A(-1)
print(a.set(a.v+1))
```

---

24. Vad svarar Python om man kör koden nedan? (2p)

```
def collatz(x):
  if x==2 or x==1:
    return []
  elif x%2==0:
    x = x//2
  else:
    x = (3*x+1)//2
  return([x]+collatz(x))

collatz(13)
```

Del 2

25. *Sjuan*

Antag att ordlistan `saol.txt` (en fil med alla svenska ord, ett ord per rad) och 7 olika bokstäver A–Ö är givna. Uppgiften här handlar om att bilda ett 7-bokstavigt och flera 7-bokstaviga ord av dessa bokstäver.

(a) Skriv programmet `finn_ett_ord` som letar efter det första 7-bokstaviga ordet i alfabetisk ordning ur `saol.txt`. Om det inte finns något ord ska programmet ge meddelandet "Ord saknas!" men om det finns ett sådant ord ska detta returneras av funktionen. (2p)

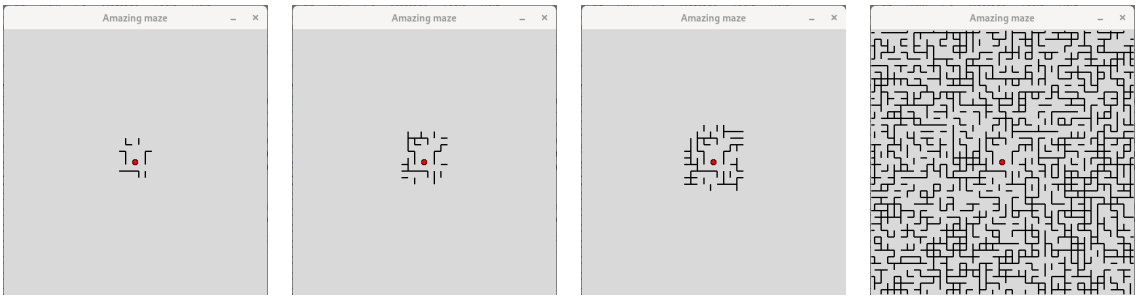
(b) Skriv ett program, `antal_ord`, som letar i `saol.txt` efter alla ord som kan bildas av de 7 givna bokstäverna. I denna uppgift (till skillnad från föregående uppgift) får det även bildas ord med färre än 7 bokstäver. T ex av bokstäverna

A, C, D, H, K, L, O

kan ordet CHOKLAD bildas men även orden OCH, KLO, KO och DOCKA. Denna funktion ska inte returnera något ord utan bara det totala *antalet olika* ord som kan bildas av de 7 bokstäverna. (3p)

26. *Amazing maze*

Målet i denna uppgift är att generera en slumpmässig labyrint. Labyrinten ska utgöras av horisontella och lodräta streck av längd 10 pixlar och hela labyrinten ska blir 400×400 pixlar. Labyrinten ska dessutom byggas "inifrån och ut" så att startläget är i mitten där starten indikeras som en liten röd cirkel med diameter 8 och ett varv av slumpmässiga streck ska omgärda den. Nästa steg ska lägga ytterligare ett varv till det första. Nästa steg ytterligare ett varv osv (se bilderna nedan) ända



Längst till vänster är labyrinten efter första varvet (dvs startläget). Andra bilden visar labyrinten efter andra varvet, tredje är labyrinten efter tredje varvet och fjärde bilden visar den fullbordade labyrinten.

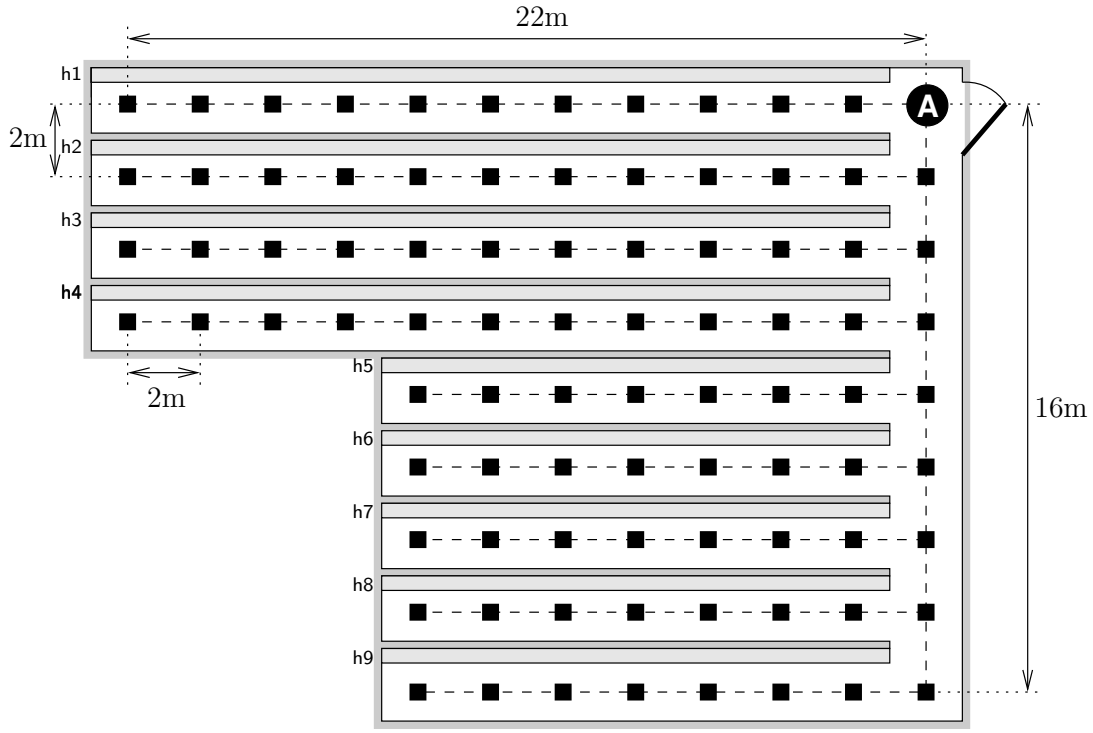
tills labyrinten når begränsningen för fönstret. Varje steg ska tas genom att man klickar på ENTER. Programmet ska kunna avbrytas genom att man klickar på mellanslagstangenten. Det ingår inte i uppgiften att garantera att det ska finnas en väg ut till någon kant från den röda punkten men labyrinten ska som sagt vara slumpmässigt konstruerad och ha ungefär lika många svarta streck som saknade svarta streck. Börja med att dela in hela ytan i ett rutnät med 10 pixlars avstånd mellan rutornas hörn.

(a) Första uppgiften är att skriva en funktion, `ett_steg`, som för en given rityta och en given ruta i den ritytan med sannolikhet 50% ritar ett horisontellt svart streck och med 50% sannolikhet ritar ett lodrätt sträck, båda två intill den givna rutan. (4p)

(b) Andra uppgiften är att skriva huvuddelen av programmet, funktionen `main`, som använder funktionen `ett_steg` för att rita alla streck som ska ingå i första varvet runt den röda startcirkeln, tala om att användaren måste trycka ENTER för att få nästa varv osv. (4p)

27. *Programmerbar truck*

Antag att du jobbar på ett lager och ska konstruera en vägväljare för en programmerbar gaffeltruck som är verksam på lagret. En skiss över lagret, dess hyllor och avstånden mellan dem ser du i figuren nedan. Trucken startar och slutar alltid sina



turer i positionen markerad Ⓐ. De streckade linjerna markerar var trucken kan åka och de fyllda kvadraterna markerar var den kan stanna. Då något ska flyttas från en plats, `p1`, till en annan, `p2`, anges positionerna som tupler `(h,d,n,t)` där `h` är namnet på hyllan (`'h1'`, `'h2'`, ...), `d` är antalet meter in från den 16 m långa huvudgången, `n` är numret på det hyllplan som gaffeln ska ställas in på och `t` är tjänsten (`'r'` för lämna, `'g'` för hämta och `'x'` för att varken lämna eller hämta) som ska utföras i denna position. Trucken tar följande kommandon:

`TCW`	Sväng 90° medsols
`GFXY`	Åk framåt `XY` meter där `X` är ett tal 0–9 och `Y` är ett jämnt tal 0, 2, 4, 6, 8.
`FXG`	Höj/sänk lyftgaffeln till hyllplan `X` (där `X` är en siffra 0–9) och *hämta* objektet där
`FXL`	Höj/sänk lyftgaffeln till hyllplan `X` (där `X` är en siffra 0–9) och *lämna* objektet där

Exempel Två positioner skulle kunna vara `('h8',10,9,'g')` och `('h3',2,1,'r')`. En order till trucken kan utgöras av ett sådant par av positioner och innebär att ett objekt ska hämtas på hyllplan 9, 10 meter in i gången vid hylla `h8` och flyttas till hyllplan 1, 2 meter in i gången vid hylla `h3`. Rätt instruktion till trucken skulle i detta fall bli: `'F9G,TCW,GF10,TCW,TCW,TCW,GF10,TCW,TCW,TCW,GF02,TCW,F1L'`.

(a) Skriv en klass `One` som givet två platser, `B` och `C` i formen av platsangivelser enligt ovan, hanterar problem baserat på dessa två platser. En metod inom denna klass, `Trip`, ska beräkna hur trucken ska färdas från `B` till `C` utan att göra något annat. En annan, `Get` ska göra samma sak som `Trip` men börja med att något hämtas i position `B`. En tredje, `Leave` ska göra samma sak som `Trip` men börja med att något lämnas i position `B`. (5p)

(b) Skriv nu `Many`, underklass till `One`, med metoden `move` som givet många positioner svarar med en instruktion som utgörs av att börja i Ⓐ, passera alla positionerna och sluta i Ⓐ med hjälp av metoderna i klassen `One`. (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.