EEBE

Informàtica

Llistes i tuples

Robert Joan Arinyo
Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

Llistes, per a què?

Necessitat de representar conjunts d'informacions variables, no necessàriamient homogènies, amb operacions àgils com:

- Modificació del valor d'un element.
- Inclusió d'un nou element.
- Eliminació d'un element.
- Concatenació de dues llistes.
- Extracció d'una subllista.

Tipus de dades

Recordem la classificació dels tipus de dades que considerem són:

1. Simples o escalars
   - enter: `int`
   - real: `float`
   - caracter: `str`
   - booleà: `bool`
2. Compostos
   - cadenes de caràcters: `str`
   - **llistes**: `list`
   - **tuples**: `tuple`
   - diccionaris: `dict`

Llistes

Llistes

- Una **llista** és un tipus compost de dades que representa una seqüència d'elements d'informació tals que
  1. Cada element pot ser d'un tipus diferent.
  2. Els elements s'individualitzen mitjançant un índex.
  3. La informació dels elements és modificable.
  4. Hom pot incloure nous elements a la llista.
  5. Hom pot eliminar elements de la llista.
- Exemples i notació
  - [10, 20, 30]
  - ['a', 'b', c']
  - [30.5, True, 'abc', -6, ['x', 'y']]
  - []

Operacions I ()

- Creació per assignació: (En tornarem a parlar)
  - Llista arbitrària: `llista = [1, 2, 'a', -12.5]`
  - La llista buida: `llista = []`
- Llargada d'una llista: `len(llista)`
- In dividualització d'una component: `llista[index]`.

  De manera anàloga al que passava a les cadenes, els índexs d'una llista pertanyen a l'interval [0..len(llista-1].

Operacions II ()

- Assignació a una component: `llista[i] = 'abc'`

  Amb `0 <= i <= len(llista) - 1`.

- **Subllista**: Conjunt d'elements consecutius d'una llista.

  Assignació d'una subllista:

  `llista[i:j] = ['a', 12, ···, 'z']`

  Amb `j - i = len['a', 12, ···, 'z']`.

Operacions III ()

- Extensió d'una llista:
  1. Per concatenació directa:
     ```
     llista = llistaA + llistaB
     ```
  2. Per adjunció d'un nou element al final:
     ```
     llista.append(valor_del_nou_element)*
     ```
  3. Per extensió d'una llista:
     ```
     llista.extend(una_llista)*
     ```

     Nota què això és equivalent a
     ```
     llista = llista + una_llista
     ```

* Funció que modifica directamet la llista sense assignació explícita.

Operacions IV ()

- Operador de test d'inclusió o pertinença d'un element a una llista
  - **Semàntica**: Decideix si un valor determinat, `valor`, és o no és un element d'una llusta donada, `llista`.
  - **Sintaxi**: `valor in llista`
- Exemple

```
llista = ["un", "dos", ["tres", 4]]
if "2" in llista :
   print("Pertany!")
else :
   print("No pertany!")
```

Operacions V ()

Còpia de llistes

- **Àlies**: una mateixa llista (informació) amb dos noms
  `llistaA = llistaB`

- Còpia real: dues llistes diferents amb el mateix contingut
  1. `llistaA = llistaB[:]`
  2. `llistaA = llistaB.copy()`

Operacions VI ()

- Esborrat de l'element amb índex `i`:
  `del(llista[i])`*
- Esborrat dels elements des d'`iInf` fins a `Isup - 1`:
  `del(llista[iInf:iSup])`
- Esborrat de tots els elements d'una llista:
  `llista.clear()`*

  Nota que això és equivalent a l'assignació
  `llista = []`

* Funció que modifica directamet la llista sense assignació explícita.

Operacions VII ()

- Ordenació de llistes **homegènies**
  - **Semàntica**: Ordenació creixent o decreixent d'una llista donada, `llista`, segons els valors de les seves components.
  - **Sintaxi**: `llista.sort(reverse = boolea)`
    Quan el `boolea` pren valor `False` o no s'utilitza, l'ordenació es fa segons valors creixents. Quan pren valor `True`, l'ordenació es fa segons valors decreixents.

- Exemple

llista	boolea	llista ordenada
['Subaru', 'BMW', 'Ferrari']	no hi és	['BMW', 'Ferrari', 'Subaru']
	False	['BMW', 'Ferrari', 'Subaru']
	True	['Subaru', 'Ferrari', 'BMW']

Interconversió entre

llistes i cadenes

Llistes i cadenes I (VI)

Exercici

Dissenya una funció tal que, donada un cadena, calculi i retorni una llista en la que cada component consecutiva contingui el caràcter corresponent de la cadena donada.

```
def transformaCadenaEnLlista(cadena):
  """
    >>> cadena = "abc, ABC"
    >>> transformaCadenaEnLlista(cadena)
    ['a', 'b', 'c', ',', ' ', 'A', 'B', 'C']
  """
```

Llistes i cadenes II (VI)

- Transformació d'una cadena en una llista
  - **Semàntica**: Transforma els caràcters d'una cadena donada, `cadena`, en una llista, `llista`, tal que cada component de `llista` representa **un caràcter** de la cadena donada, `cadena`.
  - **Sintaxi**: `llista = list(cadena)`
- Exemple

```
cadena = "L'Oncle Sam"
llista = list(cadena)
```

Aleshores la `llista` és:

```
['L', "'", 'O', 'n', 'c', 'l', 'e', ' ', 'S', 'a',
'm']
```

Llistes i cadenes III (VI)

- Transformació d'una cadena en una llista de cadenes separades segons un delimitador.
  - **Semàntica**: Transforma els caràcters d'una cadena donada, `cadena`, en una llista, `llista`, tal que cada component de `llista` representa **una subcadena** de la cadena donada, `cadena`, delimitada per una altra **subcadena** determinada, `subcadena`.
  - **Sintaxi**: `llista = cadena.split(subcadena)`
- Exemple

cadena	separador	llista
"Strawberry fields forever"	" "	["Strawberry", "fields", "forever"]
	"e"	["Strawb", "rry fi", "lds for", "v", "r"]

Llistes i cadenes IV (VI)

Exercici

Dissenya una funció tal que, donada una llista on cada component emmagatzema una cadena, calculi i retorni una cadena resultant de la concatenació de les cadenes de la llista.

```
def transformaLlistaEnCadena(cadena):
  """
    >>> llista = ['Ara', 'va' 'de', 'bo']
    >>> transformaLlistaEnCadena(llista)
    'Aravadebo'
  """
```

Llistes i cadenes V (VI)

- Transformació d'una **llista de cadenes** en una única cadena
  - **Semàntica**: Transforma les cadenes que contenen els elements d'una llista, `llista`, en una única cadena, `cadena`, conformada per la concatenació dels valors de la `llista`. Com a element d'unió s'empra una cadena donada, `cadena_unio`.
  - **Sintaxi**:
    ```
    cadena = cadena_unio.join(llista)
    ```
- Exemple

llista	cadena d'unió	cadena resultant
['Les', 'bruixes', 'es', 'pentinen']	" "	'Lesbruixesespentinen'
	"+"	'Les+bruixes+es+pentinen'

Llistes i cadenes VI (VI)

Exercici

Programa la teva versió de la funció `joinMeva()` definida pel `doctest` següent

```
def joinMeva(llista, cadena_unio):
  """
   >>> llista = ['Les', 'bruixes', 'es', 'pentinen']
   >>> cadena_unio = '+ ** +'
   >>> joinMeva(llista, cadena_unio)
   "Les+ ** +bruixes+ ** +es+ ** +pentinen"
   >>> llista = []
   >>> cadena_unio = '-??-'
   >>> joinMeva(llista, cadena_unio)
   ""
  """
```

EXERCICIS

Dissenya una funció tal que donats els coeficients de l'equació implícita d'una recta $ax + by + c = 0$, retorni els coeficients equivalents d'una recta perpendicular, $-bx + ay + c = 0$. Cal que la funció passi el `doctest`

```
def equacio(a, b, c) :
  """
    >>> equacio(5, 4, 3)
    [-4, 5]
    >>> equacio(0, 4, 3)
    [-4, 0]
    >>> equacio(5, 0, 3)
    [0, 5]
  """
```

How to Think like ... Exercice 9.22-18

Write a function `addLists(l1, l2)` that takes two lists of numbers of the same length, and returns a new list containig the sums of the corresponding elements of each.

```
def addList(a, b) :
  """
    >>> addList([1, 2, 3], [4, 5, 6])
    [5, 7, 9]
    >>> la = [10, 11, 12]
    >>> lb = [12, 11, 10]
    >>> addList(la, lb)
    [22, 22, 22]
    >>> addList([-1, 1], [1, -1])
    [0, 0]
  """
```

Donades dues llistes de noms `vectorA` i `vectorB`, totes dues amb *n* components i que emmagatzem vectors d'$\mathbb{R}^n$, dissenya la funció que calcula i retorna el producte escalar definida pel `doctest` següent:

```
def producteEscalar(vectorA, vectorB) :
  """
    >>> vectorA = [1, 2, 3]
    >>> vectorB = [-1, 2, -3]
    >>> producteEscalar(vectorA, vectorB)
    -6
    >>> producteEscalar([3, 2], [-2, 3])
    0
  """
```

Donada una llista que emmagatzema valors de tipus homogènis en totes les seves components, escriu una funció `Python` de nom `esCreixent(llista)` tal que, decideixi si està ordenada segons valors creixents o no. Cas que la resposta sigui negativa, la funció retornarà l'índex de la component de la llista que trenca l'ordenació.

```
def esCreixent(llista) :
    """
    >>> esCreixent(['a', 'b', 'c', 'd', 'e'])
    (True, -1)
    >>> esCreixent(['a', 'b', 'c', 'A', 'd'])
    (False, 3)
    >>> esCreixent(['A', 'a', 'c', 'd', 'e'])
    (True, -1)
    >>> esCreixent([1, 2, 7, - 1, 21, 12])
    (False, 3)
    """
```

Una llista emmagatzema a cada component un valor real. Escriu una funció `Python` de nom `primerNegatiu(llista)` tal que, donada una llista com la descrita determini segons índexs creixents, l'índex del primer valor negatiu de la llista. Cas que la llista no contingui cap valor negatiu, el resultat serà -1.

```
def primerNegatiu(llista) :
  """
    >>> primerNegatiu([3, 4, 5, -21, -12, 34])
    3
    >>> primerNegatiu([-33, 4, 5, -21, 12, 34])
    0
    >>> primerNegatiu([3, 4, 5, 21, 12, -34])
    5
    >>> primerNegatiu([3, 4, 5, 21, 12, 34])
    -1
    >>> primerNegatiu([])
    -1
  """
```

Una llista emmagatzema a cada component una cadena. A cada cadena com a màxim hi ha una subcadcena `"isme"`. Escriu una funció `Python` de nom `canviaCadena(llista)` tal que, donada una llista com la descrita, canvïi les ocurrències de la subcadena `"isme"` per la subcadena `"mesi"`. Usa el següent `doctest`

```
def canviaCadena(llista) :
  """
    >>> llista = ['atavisme', 'ismelar', 'traismeta']
    >>> canviaCadena(llista)
    ['atavmesi', 'mesilar', 'tramesita']
    >>> canviaCadena([])
    []
  """
```

solució sense usar la funció `find()` de `Python`

```
def cercaIsme(cadenaA, cadenaB, cadenaC):

  lenA = len(cadenaA)
  lenB = len(cadenaB)
  if lenA < lenB:
    return False, cadenaA
  else:
    for j in range(lenA - lenB + 1):
      esIsme = cadenaA[j : j+lenB] == cadenaB
      if esIsme :
        cadenaNova = cadenaA[:j] + cadenaC + cadenaA[j+lenB:]
        return True, cadenaNova
    return False, cadenaA

def canviaCadena(llista):
  """
  >>> canviaCadena(['atavisme', 'ismelar', 'traismeta'])
  ['atavmesi', 'mesilar', 'tramesita']
  >>> canviaCadena([])
  []
  """

  for i in  range(len(llista)):
    calCanviar, cadena =  cercaIsme(llista[i], "isme", "mesi")
    if calCanviar:
      llista[i] = cadena
  return llista

if __name__ == '__main__' :
  import doctest
  print(doctest.testmod())
```

solució usant la funció `find()` de `Python`

```
def actualitzaCadena(cadenaA, i, cadenaB, cadenaC):

    lenB       = len(cadenaB)
    cadenaNova = cadenaA[:i] + cadenaC + cadenaA[i+lenB:]
    return cadenaNova

def canviaCadena(llista):
  """
  >>> canviaCadena(['atavisme', 'ismelar', 'traismeta'])
  ['atavmesi', 'mesilar', 'tramesita']
  >>> canviaCadena([]) #doctest:
  []
  """
  for i in  range(len(llista)):
    j = llista[i].find("isme")
    if j > -1 :
      llista[i] = actualitzaCadena(llista[i], j, "isme", "mesi")
  return llista

if __name__ == '__main__' :
  import doctest
  print(doctest.testmod())
```

Tuples

Tuples I (IV)

- Un **tuple** és un tipus compost de dades que representa una seqüència d'elements d'informació tals que
  1. El tuple és immutable.
  2. Els elements components s'individualitzen mitjançant un índex.
  3. Els tipus de les informacions emmagatzemades en les components no són necessàriament homogènis.
- Exemples i notació
  - (10, 20, 30)
  - ('a', 'b', c')
  - (30.5, True, 'abc', -6, ['x', 'y'])
  - () tuple buit
  - (27,)

Tuples II (IV)

- **mutabilitat**: Valors i estructura poden variar.
- **immutabilitat**: Ni valors ni estructura poden variar.

- **Llistes**: Són mutables.
- **Tuples**: Són immutables.

Tuples III (IV)

Com a conseqüència de la immutabilitat de tuples, el canvi del valor d'una component no es pot fer de manera immediata. Considera el tuple

```
t = (1, 2, 3, 4, 5, 6)
```

Si hom vol substituir el valor `4` pel valor `-10` cal fer:

```
t = t[:3] + (-10,) + t[4:]
```

o bé

```
t_list      =  list(t)
t_list[3]   =  -10
t           =  tuple(t_list)
```

Tuples IV (IV)

DEURES

Havent vist a classe les llistes, estudia pel teu compte les operacions sobre tuples en, per exemple, qualsevol de les fonts següents:

- Learn Python: `https://www.tutorialspoint.com/python/python_tuples.htm`
- W3Schools: `https://www.w3schools.com/python/python_tuples.asp`
- Community: `https://www.digitalocean.com/community/tutorials/understanding-tuples-in-python-3`
- How to Think Like a Computer Scientist, capítol 9: `http://openbookproject.net/thinkcs/python/english3e/`

I consulta amb el professor els dubtes que tinguis.

Això és tot per avui