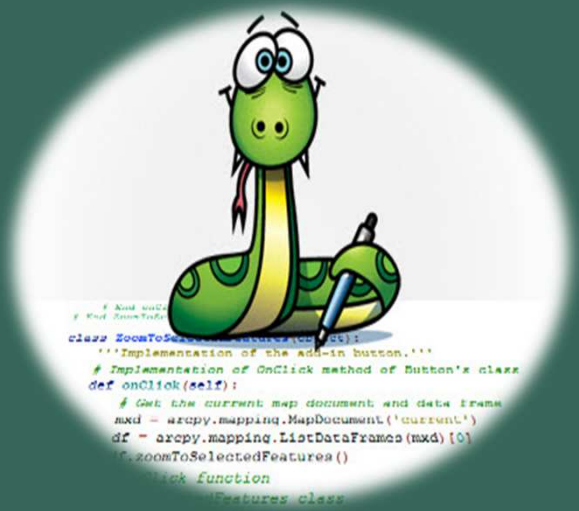
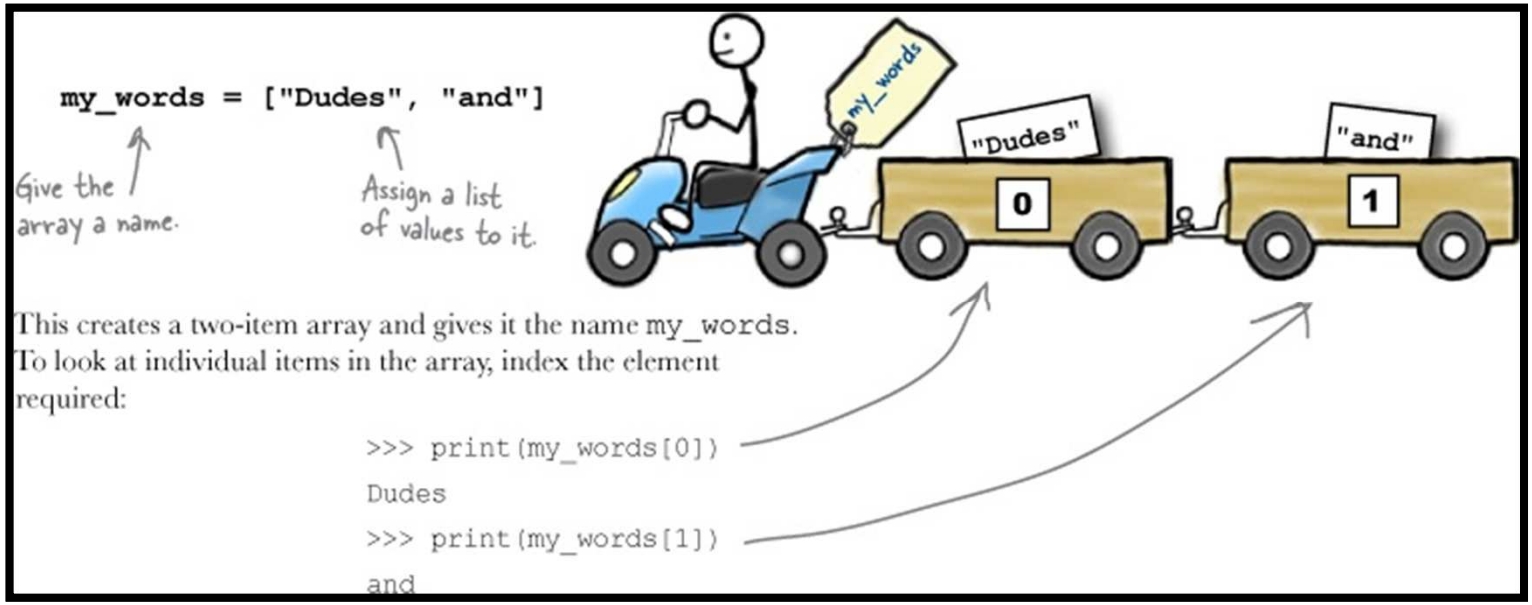
COL·LECCIONS

LLISTES, TUPLES, DICCIONARIS



INTRODUCCIÓ

- Llistes i tuples en Python són objectes que contenen llistes de dades als que s'accedeix mitjançant un índex, de forma similar als arrays o vectors d'altres llenguatges.
- Pertanyen a un tipus de dades que Python anomena seqüències i que inclou també a les cadenes de text.



CARACTERÍSTIQUES

- No hi ha límit a la quantitat d'elements que poden contenir.
- Poden contenir qualsevol tipus de dada, incloent altres seqüències. (Una matriu en Python és una llista de llistes).
- No es necessari saber la grandària(quantitat d'elements que tindrà la seqüència en el moment de crea-la).
- Suporten les funcions natives:
  - **len(seqüència)**
  - **max(seqüència)**
  - **min(seqüència)**
- Tenen dos mètodes comuns:
  - **Sequencia.index('x')**: Retorna l'índex de la primera ocurrència de 'x' en la seqüència.
  - **Secuencia.count('x')**: Retorna el número de vegades que apareix 'x' en la seqüència.

INTRODUCCIÓ

- Successió d'elements que poden ser accedits mitjançant un índex.
- La llista és un tipus de col·lecció. Seria equivalent a lo que en altres llenguatges es coneix com arrays, o vectors.
- Són variables que emmagatzemen arrays, internament cada posició pot ser un tipus diferent de dada.
- Les llistes poden contenir qualsevol tipus de dades: números, cadenes, booleans, … i també llistes.
- **CREAR UNA LLISTA:**
  - S'escriu el nom de la variable = s'indica entre claudàtors, i separat entre comes, els valors que volem incloure a la llista:
    - Exemple: **vector = [22, True, 'Hola Mon', [1, 2]]**
- **ACCEDIR ALS ELEMENTS:**
  - Per accedir e un element, tenim que indicar l'índex de l'element entre claudàtors. **L'índex del primer element de la llista és el 0**:
  - Exemple: **valor = vector [0] # valor val 22**

MODIFICAR UN ELEMENT DE LA LLISTA

- Per modificar un element de la llista, hem d'indicar el nom de la variable especificant entre claudàtors la posició a modificar:
  - Exemple: **vector = [22, True, 'Hola Mon', [1, 2]]**

    vector [0] = 33 # ara la llista serà vector = [33, True, 'Hola Mon', [1, 2]]

- En Python l'operador ***[ ]*** es pot utilitzar amb números negatius. Si s'utilitza un número negatiu com a índex, significa que l'índex comença a contar des del final, cap a la esquerra; es a dir, amb [-1] accediríem a l'últim element de la llista, amb [-2] al penúltim, amb [-3], a l'antepenúltim, i així successivament.
  - Exemple: **vector = [22, True, 'Hola Mon', [1, 2]]**

    vector [-2] = 4 # ara la llista serà vector = [22, True, 4, [1, 2]]

Join (unió)

- Transforma una seqüència en una cadena. Coloca un separador.
- Exemple:

```
print('-'.join(['1', '2', '3', '4', '5', '6']))
print('.'.join(['1', '2', '3', '4', '5', '6']) )
```

```
1-2-3-4-5-6
1.2.3.4.5.6
```

SLICING (LLESQUES)

- Permet obtenir una part de la seqüència original utilitzant **sequencia[x:y:z]**
- Retorna una seqüència amb les característiques següents:
  - Del mateix tipus que la original (una llesca d'un llista és una llista, una llesca d'una tupla és una tupla i una llesca d'una cadena és una cadena).
  - Conté els elements des de seqüència[x] fins a sequencia[y-1]
  - Saltant-se z elements cada vegada

SLICING: PARTICIONAT

- ***slicing*** o particionat: Consisteix en permetre seleccionar porcions de la llista. Si en lloc d'un número escrivim dos números inici i fi separats per dos punts (inici : fi) Python interpretarà que volem una llista que vagi des de la posició inici a la posició fi, sense incloure aquest últim.
  - Exemple: **vector = [99, True, 'Hola mon', [1.36 , 2.89]]**

valor1 = vector [0 : 2] # valor1 val [99, True]

- Si escrivim tres números **[inici : fi : salt]** en lloc de dos, el tercer s'utilitza per determinar cada quantes posicions afegir un element a la llista.
- Exemple: **vector = [99, True, 'Hola Mon', [1.36 , 2.89]]**

valor1 = vector [0 : 4 : 2] # valor1 val [99, 'Hola Mon']

Els números negatius també es poden utilitzar en un *slicing*, amb el mateix comportament.

SLICING: PARTICIONAT

- No es necessari indicar el principi i el final del slicing, sinó que, si aquests s'ometen, s'utilitzaran per defecte les posicions d'inici i fi de la llista, respectivament:
  - Exemple: **vector = [84, True, 'Hola Mon']**

    valor = vector [1 :] **# valor val [True, 'Hola Mon']**

    valor = vector [: 2] **# valor val [84, True]**

    valor = vector [:] **# valor val [84, True, 'Hola Mon']**

    valor = vector [: : 2] **# valor val [84, 'Hola Mon']**

- També podem utilitzar aquest mecanisme per a modificar la llista:
  - Exemple: **vector = [84, True, 'Hola Mon', [1, 2]]**

    vector [0 : 2] = [0 , 1] **# valor vector [0, 1, 'Hola Mon', [1, 2]]**

- **Podem modificar inclús la mida de la llista:**

  vector [0 : 2] = [False] **# valor vector [False, 'Hola Mon', [1, 2]]**

REEMPLAÇAR

- Es poden reemplaçar parts d'una llista amb altra, o amb parts de'altra, utilitzant la notació de llesques(slices):
- **mi_lista[0:2] = [3, 4]** # reemplaça els dos primers elements

```
secuencia=[1,2,3,4,5]
print(secuencia)
secuencia[0:2]=[33,44]

print(secuencia)
```

```
C:\WINDOWS\system32\cmd.exe
[1, 2, 3, 4, 5]
[33, 44, 3, 4, 5]
```

ACCEDIR ALS ELEMENTS D'UNA LLISTA INCLOSA EN ALTRA LLISTA

- Si volem accedir a un element d'una llista inclosa a dintre d'altra llista, posarem dos claudàtors: el primer per indicar a quina posició de la llista exterior volem accedir, i el segon per seleccionar l'element de la llista interior:
  - Exemple: **vector = ['una llista', [1 , 2]]**

    valor = vector [1] [1] # valor val 2

FUNCIONS DE LES LLISTES: *len( )*

- Per a conèixer la longitud d'una llista es pot aplicar la funció ***len( )***.
- Exemples:

13_exemple_listes_Exemple_Longitud.py

```
1
2  numeros = [1258,-695,0,2,36]
3  print (' Quantitat d\'elements d ela llista: ',len(numeros))
4
5
6
7
```

```
C:\WINDOWS\system32\cmd.exe
Quantitat d'elements d ela llista:  5
```

FUNCIONS DE LES LLISTES: *append(num) , insert(posicio,num)*

- **append**: Serveix per agregar un nou valor al final de la llista. La funció no controla si el valor ja existeix o no en la llista. La seva forma és:

  append (num_a_inserir)

13_exemple_llistes_Exemple_Agregar.py

```
1  # Exemple agregar element al final d'una llista
2
3  numeros = [1258,-695,0,2,36]
4  numeros.append(-88)
5
6  for num in numeros:
7      print(num)
8
9
10
```

C:\WINDOWS\system32\cmd.exe

```
1258
-695
0
2
36
-88
```

- **insert**: Insereix un nou valor a la posició de l'index que se l'indica com a argument i desplaça una posició la resta d'elements de la llista. La seva forma és:

  insert(posició, num_a_inserir)

13_exemple_llistes_Exemple_Insert.py

```
1  # Exemple insertar element en una posició concreta d'una llista
2
3  numeros = [1258,-695,0,2,36]
4  numeros.insert(2,77)
5
6  for num in numeros:
7      print(num)
8
9
10
```

C:\WINDOWS\system32\cmd.exe

```
1258
-695
77
0
2
36
```

FUNCIONS DE LES LLISTES: *remove(num)*

- **remove**: Elimina un valor de la llista. La seva forma es: **remove(num_a_eliminar)**

13_exemple_llistes_Exemple_Remove.py

```
1  # Exemple eliminar element d'una llista
2
3  numeros = [1258,-695,0,2,36]
4  numeros.remove(0)
5
6  for num in numeros:
7      print(num)
8
```

C:\WINDOWS\system32\cmd.exe

```
1258
-695
2
36
```

- Si hi ha valors repetits, només elimina la primera aparició del número a eliminar:

13_exemple_llistes_Exemple_RemoveRepetits.py

```
1  # Exemple eliminar element repetit d'una llista
2
3  numeros = [0,0,1258,-695,0,2,36,0]
4  numeros.remove(0)
5
6  for num in numeros:
7      print(num)
8
9
10
11
```

C:\WINDOWS\system32\cmd.exe

```
0
1258
-695
0
2
36
0
```

Si el valor a eliminar no existeix, dona error.

13_exemple_llistes_Exemple_RemoveNoExisteix.py

```
1  # Exemple eliminar element repetit d'una llista
2
3  numeros = [1258,-695,2,36]
4  numeros.remove(0)
5
6  for num in numeros:
7      print(num)
8
```

C:\WINDOWS\system32\cmd.exe

```
Traceback (most recent call last):
  File "M:\Curs2016-2017\codi\13_exemple_llistes_Exemple_RemoveNoExisteix.py", line 4, in <module>
    numeros.remove(0)
ValueError: list.remove(x): x not in list
```

FUNCIONS DE LES LLISTES: *index( ), in*

- **index**: Ens indica la posició d'un valor a dintre de la llista. La seva forma és: **index( )**

13_exemple_listes_Exemple_Index.py

```
1  # Exemple posició d'un element en la llista
2
3  numeros = [1258,-695,2,36]
4  indice= numeros.index(36)
5
6  print(' L\'element es troba a la posició:',indice)
7
```

```
C:\WINDOWS\system32\cmd.exe
L'element es troba a la posició: 3
```

  - Si el valor no es troba a la llista es produirà un error:

13_exemple_listes_Exemple_Index_noEncontrado.py

```
1  # Exemple posició d'un element que no es troba en la llista
2
3  numeros = [1258,-695,-2,36]
4  pos=numeros.index(-369)
5
6  print('L\'element està a la posició:',pos)
7
```

```
C:\WINDOWS\system32\cmd.exe
Traceback (most recent call last):
  File "M:\Curs2016-2017\codi\13_exemple_listes_Exemple_Index_noEncontrado.py", line 4, in <module>
    pos=numeros.index(-369)
ValueError: -369 is not in list
```

  - Si hi ha valors repetits, l'índex que retorna és el de la primera aparició.

- **in**: Per preguntar si un valor determinat és un element d'una llista s'utilitza l'operació **in**.

13_exemple_listes_Exemple_In.py

```
1  # Exemple buscar un element en la llista
2
3  numeros = [1258,-695,-2,36]
4
5  if (2 in numeros):
6      print('L\'element està en la llista.')
7  else:
8      print('L\'element no està  a la llista.')
```

```
Seleccionar C:\WINDOWS\system32\cmd.exe
L'element no està  a la llista.
```

FUNCIONS DE LES LLISTES: *sort( ), reverse()*

- **sort()**: Reordena la llista
- **reverse()**: Inverteix els elements

```
secuencia=[500, -2,1,2,3,4,5]
secuencia.sort()
print(secuencia)
secuencia.reverse()

print(secuencia)
```

```
C:\WINDOWS\system32\cmd.exe
[-2, 1, 2, 3, 4, 5, 500]
[500, 5, 4, 3, 2, 1, -2]
```

CONCATENCIÓ(SUMA)

- La suma de dos seqüències a y b genera una altra seqüència que conté els elements d'ambdues i en la que els elements de b apareixen després des de a. Les seqüències han de ser del mateix tipus. No es pot sumar cadenes i tuples, o tuples i llistes.

```
secuencia=[1,2,3,2,5,2,2,8,9]
secuencia2=[20,21]
print(secuencia)
print(secuencia+secuencia2)
```

```
C:\WINDOWS\system32\cmd.exe
[1, 2, 3, 2, 5, 2, 2, 8, 9]
[1, 2, 3, 2, 5, 2, 2, 8, 9, 20, 21]
```

MULTIPLICACIÓ

- El resultat de la multiplicació de seqüències per un número **n**, és sumar la seqüència a sí mateixa **n** vegades.

```
print('a'*5)
secuencia=[1,2,3,4,5]
print(secuencia*3)
```

```
C:\WINDOWS\system32\cmd.exe
aaaaa
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```