Cronograma

- Algunas conceptos sobre listas y tuplas
- Diccionarios
  - Características generales
  - Recorriendo diccionarios
  - Buscar elementos
  - Agregar elementos
  - Borrar elementos
  - Funciones para diccionarios
  - Ejercicios

Sobre listas y tuplas

- ¿Cuál es la diferencia entre una lista y una tupla?
  - Ambos pueden tener elementos de cualquier tipo
  - Ambos mantienen el orden de los elementos
- Diferencias técnicas: mutabilidad vs inmutabilidad
  - Listas → mutables (pueden ser cambiadas)
    - Cambio de tamaño automático
  - Tuplas → inmutables (no se pueden cambiar)
    - No se permite cambiar el tamaño

Sobre listas y tuplas (2)

- Esta diferencia se manifiesta de muchas formas, por ejemplo, las listas tienen un método *append()* para añadir elementos, mientras que las tuplas no.
- Las tuplas no tienen necesidad del método *append() (ni de otros)* porque no se pueden modificar.

Sobre listas y tuplas (3)

- ¿Cuándo usar una u otra colección?
- En general, **las listas** se utilizan cuando se tiene una secuencia de longitud desconocida
- **Las tuplas** se utilizan cuando se conoce el número de elementos de antemano. La posición del elemento es semánticamente significativa.

Sobre listas y tuplas (4)

- Supongamos que tenemos una función que dentro de un directorio nos devuelve todos los archivos .py (*.py). El resultado debería entregarse en una lista, porque no sabemos cuantos archivos podemos encontrar. Es algo cambiante.
- Supongamos que necesitamos almacenar 5 valores para representar la ubicación de estaciones de monitoreo: *id*, *ciudad*, *departamento*, *lat y lon*. Una tupla es lo más adecuado para esto en lugar de una lista:
  - *sta_lucia = (44, "Sta. Lucia", "CAN", 40, 105)*

Diccionarios (I)

- Colecciones que relacionan **clave:valor**
- Por ejemplo:
  - *dicc = {'Sueños de libertad': 'Frank Darabont','12 Monos':'Terry Gilliam','El coleccionista de huesos':'Phillip Noyce'}*
- El primer valor es la clave, y el segundo es el valor.
- Como clave podemos utilizar solamente **tipos inmutables**: números, cadenas, booleanos, tuplas

Diccionarios (2)

- En otros lenguajes, a los diccionarios se los llama arreglos asociativos, matrices asociativas, o también tablas de hash
- Es posible definir un diccionario directamente con los miembros que va a contener.
- También se puede inicializar el diccionario vacío y luego agregar los valores de a uno o de a muchos.

Diccionarios (3)

- *Ejemplos*
  - Se encierra el listado de valores entre llaves { }
  - Las parejas de clave y valor se separan con comas
  - La clave y el valor se separan con ’:’
    - punto = {’a’: 1, ’b’: 2, ’c’: 3}
    - clase = {}
    - clase["lunes"] = ['8:00', '10:00']
    - clase["martes"] = ('8:00', '10:00')
    - clase[“miercoles”] = “No hay”

Diccionarios (4)

- A diferencia de las listas que se accede por índice los diccionarios se accede por la clave utilizando *[ ]*
  - *print (dicc['Sueños de libertad']) → 'Frank Darabont'*
  - *print (clase["martes"]) →* ('8:00', '10:00')
- Se pueden obtener el valor según su clave pero NO se pueden obtener claves usando los valores

Diccionarios (5)

- Si tratamos de buscar una clave que no esta en el diccionario nos mostrará un mensaje de error.
  - *print (clase["jueves"])*
  - Traceback (most recent call last):

    File "<stdin>", line 1, in <module> KeyError: 'jueves'
- La función ***get()*** devuelve el valor None si la clave no está en el diccionario o un valor por omisión (opcional).
  - *print (clase.get("jueves")) → None*
  - *print (clase.get("jueves", [])) → []*

Diccionarios - Características

- No puede haber claves duplicadas. La asignación de un nuevo valor a una clave preexistente borrará el viejo valor.
- No existe un orden en el diccionario.
- Los diccionarios no sirven solamente para almacenar cadenas. Los valores de un diccionario pueden ser de cualquier tipo.
- Aunque las claves están restringidas a que sean de un tipo inmutable, también se pueden mezclar distintos tipos de claves en un diccionario

Recorriendo un diccionario

- Es posible recorrer sus claves y usar esas claves para acceder a los valores.

```
for dia in clase:
    print (dia, ":", clase[dia])
```

- Salida:
  - martes : ('8:00', '10:00')
  - miercoles : No hay
  - lunes : ['8:00', '10:00']

Recorriendo un diccionario (2)

- También es posible obtener los valores como tuplas donde el primer elemento es la clave y el segundo el valor

```
for dia, codigos in clase.items():
    print (dia, ":", codigos)
```

- Salida
  - martes : ('8:00', '10:00')
  - miercoles : No hay
  - lunes : ['8:00', '10:00']

Encontrando un elemento en el diccionario

- Para verificar si una clave se encuentra en el diccionario, es posible utilizar la palabra reservada ***in***.

```
dic = {'x':12, 'y':13}
if 'y' in dic:
 print(d['y'])
```

Agregar elementos

- Podemos utilizar la función ***update*** para agregar un elemento al diccionario.
  - *d = {0:10, 1:20}*
  - *print(d) →{0: 10, 1: 20}*
  - *d.update({2:30})*
  - *print(d) → {0: 10, 1: 20, 2: 30}*
  - *d.update({3:40,4:50,6:70})*
  - *print(d) → {0: 10, 1: 20, 2: 30,3:40,4:50,6:70}*

Borrar elementos

- Podemos utilizar la función ***del*** para borrar un elemento con cierta clave en el diccionario.
  - *d = {'x':12,'y':7}*
  - *del (d['x'])*
  - *print (d) →{'y': 7}*

- También podemos utilizar la función pop(key)

Mas funciones para diccionarios

- *dict.get(clave,[default])*
  - devuelve el valor de la clave o lo que definamos por defecto si la clave no se encuentra en el diccionario
- *diccionario.keys()*
  - Devuelve una lista desordenada, con todas las claves

Mas funciones sobre los diccionarios (2)

- *diccionario.values()*
  - Devuelve una lista desordenada, con todos los valores
- *diccionario.items()*
  - Devuelve una lista desordenada con tuplas de dos elementos, en las que el primer elemento es la clave y el segundo el valor.
- *diccionario.clear()*
  - Elimina todos los elementos del diccionario

Ejercicios

- Ejercicio 1
  - Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundos.
  - Por ejemplo:

    lista = [('Hola', 'don Pepito'), ('Hola', 'don Jose'), ('Buenos', 'días')]

    print (tuplas_a_diccionario(lista))

    Deberá mostrar: { 'Hola': ['don Pepito', 'don Jose'], 'Buenos': ['días'] }

Ejercicios (2)

- Ejercicio 2
  - Escribir una función para imprimir un diccionario donde las claves sean los números entre 1 y n, n es un número natural recibido como parámetro (ambos incluidos) y los valores son los cuadrados de la clave
  - Ejemplo:

    {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

Ejercicios (3)

- Ejercicio 3

  a) Escribir una función que reciba una cadena y devuelva un diccionario con la cantidad de apariciones de cada palabra en la cadena.

  - Por ejemplo, si recibe "Qué lindo día que hace hoy"debe devolver: ’que’: 2, ’lindo’: 1, ’día’: 1, ’hace’: 1, ’hoy’: 1

  b) Escribir una función que cuente la cantidad de apariciones de cada caracter en una cadena de texto, y los devuelva en un diccionario.