Cronograma

- Propiedades de la lista.
- Más funciones
  - Split
  - Join
  - List
- Nuevo tipo de datos: Tuplas
- Manipulación de Tuplas y Listas
- Cuando utilizar las listas o tuplas
- Ejercicios

Listas

- Es un tipo de colección arbitraria de elementos que nos permite agrupar a todos.
- Una lista puede ser definida por los elementos separados por coma
- Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos, y también listas

```
mi_lista = [66, False,"Verdad", [1,2,3,4]]
print (mi_lista)  → [66, False, 'Verdad', [1, 2, 3, 4]]

mi_lista[0:2] = [1,True]
print (mi_lista) → [1, True, 'Verdad', [1, 2, 3, 4]]

print (len(mi_lista))   → 4
mi_lista[0:2] = [2,3,4,5]

print (mi_lista)
print (len(mi_lista))   ├→ 6
```

Propiedades de la Lista

- Son ordenadas
- Pueden contener cualquier tipo de datos dentro.
- Los elementos de una lista se pueden acceder a través de indices.
- Se puede anidar más listas dentro de una lista.
- Son objetos mutables.
- Es dinámica

Comparaciones entre listas

- Podemos comparar dos listas con los operadores matemáticos

  = =, <, <=, >, >=

- La comparación se realiza por orden lexicógrafo
- Primero se compara los dos primeros elementos, si son iguales se comparan con los siguientes. Si son distintos se hace la comparación.
- Si una secuencia es una subsecuencia inicial de la otra, la secuencia más corta es la más pequeña (menor)

Comparaciones entre listas (2)

```
#Creamos dos listas iguales
lista1, lista2 = [1,2,3,4], [1,2,3,4]
print lista1 == lista2
>> True


#Creamos dos listas diferentes
lista1, lista2 = [1,2,3,4], [1,2,3,4,5]
print lista1 == lista2
>> False

#Creamos dos listas iguales pero desordenas
lista1, lista2 = [1,2,3,4], [4,3,2,1]
print lista1 == lista2
>> False

#La ordenamos y la comparamos
print lista1.sort() ==  lista2.sort()
>> True
```

Convertir cadenas a listas

- Podemos pasar de una cadena a lista con la función ***split()***
  - ***str.split(sep ,maxsplit=-1) -> list of strings***
  - ***sep : carácter de separación***
  - ***maxsplit : cantidad máxima de separaciones realizados.***
- Por ejemplo:
  - *lista_favorita = "mi palabra favorita".split(" ")*
  - print (lista_favorita) → *['mi', 'palabra', 'favorita']*
  - lista_numeros = "1:2:3:4:5".split(":",2)
  - print (lista_numeros) → ['1', '2', '3:4:5']

Convertir cadenas a listas

- Pero qué pasa si queremos obtener una lista con cada elemento.
- Tenemos la función ***list()***
  - list() -> genera una lista vacía
  - list(iterable) -> genera una lista con cada elemento.
- Por ejemplo:
  - a = list("mi palabra")
  - print (a) → ['m', 'i', ' ', 'p', 'a', 'l', 'a', 'b', 'r', 'a']

Convertir listas a cadenas

- Si queremos realizar el proceso inverso, convertir una lista a una palabra.
- Tenemos la función ***join()***
  - ***separador.join(iterable) -> string***
  - Retorna una palabra que contiene la concatenación de cada elementos de la lista separados por el elemento separador
- Por ejemplo.
  - palabra = " ".join(['mi','palabra','completa'])
  - print (palabra) → “mi palabra completa”

Tuplas

- La tupla es una secuencia heterogénea de elementos, accesible a través de un índice.
- Las tuplas se diferencian de las listas porque las tuplas no son objetos dinámicos, por lo tanto son **INMUTABLES**, como los números
- Se declara una tupla usando paréntesis y separando los valores por una coma.

```
t = (1,2,3)
print(t) → (1, 2, 3)
```

Tuplas (2)

- **El tipo tuple:** *print (type(t))* →
  - *<type 'tuple'>*
- Las tuplas se acceden mediante un índice.
  - t = ("a","b",2)
  - t[0] → “a”
  - t[-1] → 2
  - t[1:3] → (“b”,2)
  - t += (3,)
- Pero no las podemos modificar! (inmutables)
  - t[0] = 1 TypeError: 'tuple' object does not support item assignment

Tuplas (3)

- A las tuplas también se les puede aplicar la función len() para calcular su longitud
  - len(t) → 3
  - Z = ()
  - len(Z) → 0
- Se pueden concatenar tuplas mediante operador +
- No se pude concatenar cualquier tipo:

```
print(d + e + lista) → TypeError: can only
concatenate tuple (not "list") to tuple
```

- Cuidado: Una vez creada una tupla no se pueden borrar sus elementos

Empaquetado de tuplas

- Si a una variable se le asigna una secuencia de valores separados por comas, el valor de esa variable será la tupla formada por todos los valores asignados
  - a = 1
  - b = 2
  - c = 3
  - d = a,b,c
  - len(d) → 3
  - d → (1,2,3)
  - d += (4,)

Desempaquetado de tuplas

- Lo mismo sucede al revés (Desempaquetado)
  - m1,m2,m3 = (3,2,1)
  - print (m1)
  - print (m2)
  - print (m3)
- A las tuplas se le pueden aplicar las operaciones de rango (con el carácter ':', tal como vimos en listas y strings)

Ejercicios

Ejercicio 1. Escribir una función que reciba una tupla de elementos e indique si se encuentran ordenados de menor a mayor o no (pueden usar las funciones max y min, pero cuidado, no se pueden comparar tipos diferentes).

Ejercicio 2. Dominó.

a) Escribir una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos tuplas, por ejemplo: (3,4) y (5,4).

b) Escribir una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en una cadena, por ejemplo: 3-4 2-5. Nota: utilizar la función split de las cadenas.

Ejercicios (2)

- **Ejercicio 3**

  Escribir una función que reciba una lista de tuplas (apellido, nombre, inicial del segundo_nombre) y devuelva una lista de strings donde cada uno contenga, primero el nombre, luego la inicial con un punto, y luego el apellido.

Ejercicios (3)

- **Ejercicio 4: Batalla Naval**
  - Jugaremos contra la máquina. Haremos una tablero de 5 X 5 . Cada posición del tablero tiene una 'O'
  - La máquina eligirá una fila y columna de forma aleatoria donde ubicará su barco.
  - Debermos adivinar la ubicación en un máximo de 3 intentos
  - En caso de no acertar, en cada jugada debemos marcar la opción elegida con una 'X'

Ejercicios (4)

```
Juguemos as la batalla naval!
O O O O O
O O O O O
O O O O O
O O O O O
O O O O O
Turno  1
Adivina fila: 1
Adivina columna: 2
No tocaste mi barco!
O O O O O
O O X O O
O O O O O
O O O O O
O O O O O
Turno  2
Adivina fila:
```