Clase 3. Estructuras de Datos Básicas

TRABAJADORES INFORMÁTICOS
organizados en AGC

Ministerio de Trabajo, Empleo y Seguridad Social
Argentina

¿Qué es una

Estructura de Datos?

- **Permite organizar, administrar y almacenar datos**
- **Es una colección de datos**

¿Qué es una

Tupla en Python?

Las tuplas (o ***tuple***) en Python son otro tipo de datos (como int, float, string, etc) que permite representar a una **Estructura de Datos:**

- **Permite almacenar una secuencia ordenada de datos**
- **Sirven para almacenar múltiples datos en una sola variable**

TUPLAS

Creación

Se pueden crear tuplas vacías o con valores

Python

```
>>> tupla_vacia = ()
>>> tupla_vacia
()
```

Python

```
>>> tupla_con_valores = (0,1,2,3)
>>> tupla_con_valores
(0,1,2,3)
>>> tupla_con_valores = ("Curso", "Introductorio", 2021)
>>> tupla_con_valores
('Curso', 'Introductorio', 2021)
```

TUPLAS

Obtener

Python

```
>>> equipos = ("racing", "boca", "independiente", "river", "san lorenzo")
>>> equipos[0]
'racing'
>>> equipos[-1]
'san lorenzo'
>>> equipos[0:3]
('racing', 'boca', 'independiente')
>>> equipos[3:0:-1]
('river', 'independiente', 'boca')

# Es un error consultar por posiciones que no existen!
>>> equipos[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
```

~~Modificar~~

Python 

```
# Es un error modificar una tupla!
>>> equipos = ("racing", "boca", "independiente", "river", "san lorenzo")
>>> equipos[4] = "huracan"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Las tuplas son **inmutables**, es decir, no se pueden modificar.

Cómo recorrerlas

Python

```
>>> equipos = ("racing", "boca", "independiente", "river", "san lorenzo")
>>> for equipo in equipos:
...     print(equipo)
...
racing
boca
independiente
river
san lorenzo
```

Otras operaciones

Python 

```
>>> avellaneda = ("racing", "independiente")
>>> len(avellaneda)
2
>>> capital = ("boca", "river", "san lorenzo")
>>> avellaneda * 2
('racing', 'independiente', 'racing', 'independiente')
>>> equipos = avellaneda + capital
>>> equipos
('racing', 'independiente', 'boca', 'river', 'san lorenzo')
```

Por qué las utilizamos?

Permiten agrupar fácilmente y de forma ordenada múltiples datos (valores) relacionados entre sí y que no van a ser modificados.

Por ejemplo, podríamos agrupar un día como (día, mes, año).

Nos permite implementar funciones que devuelvan múltiples datos.

Ejercicio

10 mins ⏱️

Contador de Tuplas

Implementar una función **contar_tuplas** que reciba una tupla con tuplas en su interior, y devuelva una tupla con la cantidad de túplas que había y la cantidad total de elementos (entre todas las tuplas)

Ejercicio

Contador de Tuplas

Python 

```
def contador_tuplas(tuplas):
    elementos = 0
    for tupla in tuplas:
        elementos += len(tupla)
    return len(tuplas), elementos
```

¿Qué es una

Lista en Python?

Las listas (o ***list***) en Python son otro tipo de datos que permite representar a una **Estructura de Datos:**

- **Permite almacenar una secuencia ordenada de datos**
- **Sirven para almacenar múltiples datos en una sola variable**
- **Es una estructura de datos mutable, es decir, se pueden modificar sus datos**

LISTAS

Creación

Se pueden crear tuplas vacías o con valores

Python

```
>>> lista_vacia = []
>>> lista_vacia
[]
```

Python

```
>>> lista_con_valores = [0,1,2,3]
>>> lista_con_valores
[0,1,2,3]
>>> lista_con_valores = ["Curso", "Introductorio", 2021]
>>> lista_con_valores
['Curso', 'Introductorio', 2021]
```

LISTAS

Obtener

Hasta acá es muy parecido a lo que vimos para las **tuplas**

Python

```
>>> equipos = ["racing", "boca", "independiente", "river", "san lorenzo"]
>>> equipos[0]
'racing'
>>> equipos[-1]
'san lorenzo'
>>> equipos[0:3]
['racing', 'boca', 'independiente']
>>> equipos[3:0:-1]
['river', 'independiente', 'boca']

# Es un error consultar por posiciones que no existen!
>>> equipos[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: lista index out of range
```

Modificar

Python

```
>>> equipos = ["racing", "boca", "independiente", "river", "san lorenzo"]
>>> equipos.append("huracan")
>>> equipos
['racing', 'boca', 'independiente', 'river', 'san lorenzo', 'huracan']
>>> equipos[5] = "velez"
>>> equipos
['racing', 'boca', 'independiente', 'river', 'san lorenzo', velez']
```

A diferencia de las tuplas que son inmutables, las **listas** son estructuras **mutables**

Eliminar por índice

Python

```
>>> equipos = ["racing", "boca", "independiente", "river", "san lorenzo"]
>>> equipos.pop()
'san lorenzo'
>>> equipos
['racing', 'boca', 'independiente', 'river']
>>> equipos.pop(2)
['racing', 'boca', 'river']

# Es un error borrar posiciones que no existen!
>>> equipos.pop(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: pop index out of range
```

Eliminar por valor

Python

```
>>> equipos = ["racing", "boca", "independiente", "river", "san lorenzo"]
>>> equipos.remove("boca")
>>> equipos
['racing', 'independiente', 'river', 'san lorenzo']

# Es un error borrar valores que no existen!
>>> equipos.remove("atlanta")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

Cómo recorrerlas

Python 

```
>>> equipos = ["racing", "boca", "independiente", "river", "san lorenzo"]
>>> for equipo in equipos:
...     print(equipo)
...
racing
boca
independiente
river
san lorenzo
```

Otras operaciones

Python 

```
>>> avellaneda = ["racing", "independiente"]
>>> len(avellaneda)
2
>>> capital = ["boca", "river", "san lorenzo"]
>>> avellaneda * 2
['racing', 'independiente', 'racing', 'independiente']
>>> equipos = avellaneda + capital
>>> equipos
['racing', 'independiente', 'boca', 'river', 'san lorenzo']
```

Por qué las utilizamos?

Permiten almacenar de forma ordenada múltiples datos (valores) y se puede ir actualizando.

Ejercicio

10 mins ⏱️

FizzBuzz Variante:

En esta ocasión, queremos implementar una función **fizzbuzz** que en lugar de imprimir los valores correspondientes a fizzbuzz, devuelva una **lista con los mismos.**

Ejemplo:

```
>>> fizzbuzz(5)
[“FizzBuzz”, 1, 2, “Fizz”, 4, “Buzz”]
```

Ejercicio

Fizzbuzz Variante

Python 

```
def fizzbuzz(n):
    fizzbuzzes = []
    for i in range(n+1):
        fizzbuzzes.append(obtener_fizzbuzz(i))
    return fizzbuzzes
```

Podría resolverse este ejercicio fácilmente devolviendo una tupla en lugar de lista?

¿Qué es una

Secuencia en Python?

En Python podemos agrupar a una serie de tipos de datos con propiedades comunes bajo el nombre de Secuencias. Estos incluyen a las **tuplas**, **listas** y **strings**. Estas comparten las siguientes propiedades:

- **Se pueden usar los métodos + y * como los vistos anteriormente**
- **Se pueden usar los operadores [i] para acceder a una posición y [i:f:s] para obtener un “slice” de la secuencia**
- **Se pueden iterar usando un ciclo for como los vistos anteriormente para tuplas**

Ejercicio

10 mins ⏱️

Concatenar:

Implementar una función **concatenar** que reciba una tupla con múltiples strings y un string como separador, y que devuelva un string que tenga todos los strings concatenados con el separador.

Ejemplo:

```
>>> concatenar(("hola", "como", "estan"), "-")
"hola-como-estan"
```

Ejercicio

Concatenar

Python 

```
def concatenar(palabras, separador):
    resultado = ""
    completado = 0
    for palabra in palabras:
        resultado += palabra
        completado += 1
        if (completado != len(palabras)):
            resultado += separador
    return resultado
```

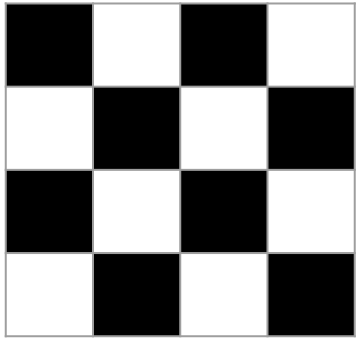
Serviría si en vez de recibir una tupla, recibiera una **lista**?
Qué pasaría si en lugar de una tupla o una lista, recibiera un **string**?

¿Cómo representar

Matrices?

Las matrices son estructuras de 2 dimensiones, por ejemplo, una matriz podría servir para representar un tablero de ajedrez.

- **En Python no hay una estructura nativa que represente matrices**
- **Podemos representar las matrices como una lista de listas**



MATRICES

Ejemplos

Python

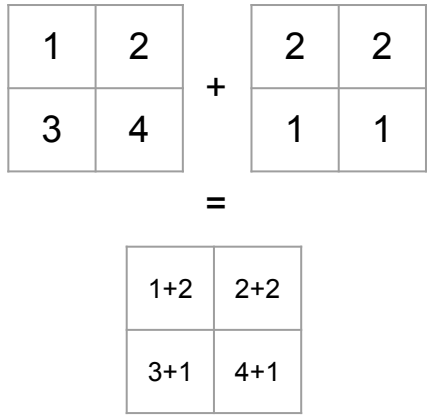
```
>>> matriz = [[1,2,3],[4,5,6],[7,8,9]]
>>> matriz[0][1]
2
>>> for fila in matriz:
...     print(fila)
...
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

Ejercicio

15 mins ⏱️

Suma de Matrices:

Implementar una función **sumar_matrices** que reciba 2 matrices de igual tamaño representadas como lista de listas, y que devuelva una matriz (también como lista de listas) que tenga posición a posición la suma de las recibidas.



Ejemplo:

```
>>> sumar_matrices([[1,2],[3,4]], [[2,2],[1,1]])
[[3,4],[4,5]]
```

Ejercicio

Suma Matrices

Python 

```
def sumar_matrices(m1, m2):
    filas = len(m1)
    columnas = len(m1[0])
    resultado = []
    for i in range(filas):
        resultado.append([])
        for j in range(columnas):
            resultado[i].append(m1[i][j] + m2[i][j])
    return resultado
```