

Introducción a la programación con Python

Estructuras y Clonación

Alexis Rodríguez

Marcel Morán C

Esquema

- ¿Que es una estructura?
- Tipos de estructuras
- Terminología y sintaxis de estructuras en Python
- ¿Cómo acceder a los elementos dentro de las estructuras?
- Bucles con estructuras
- Clonacion

¿Cómo puedo guardar un tipo de dato?



¿Cómo puedo guardar un tipo de dato?

- “String”
- 1
- 1.0
- True

¿Cómo puedo guardar un tipo de dato?

- “String”
- 1
- 1.0
- True

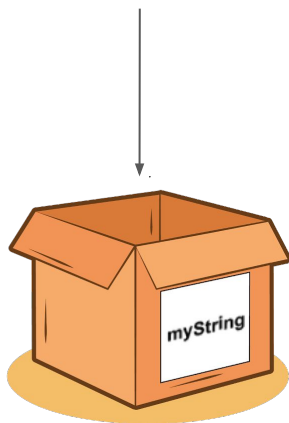
encebollado = “encebollado”



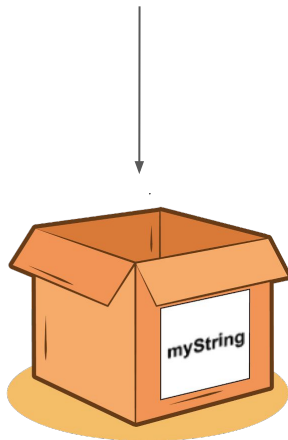
¿Cómo puedo guardar un tipo de dato?

- “String”
- 1
- 1.0
- True

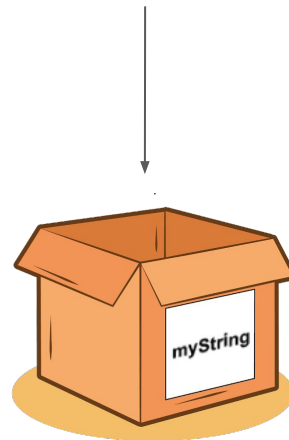
encebollado = “encebollado”



bolon = “bolon”



fritada = “fritada”



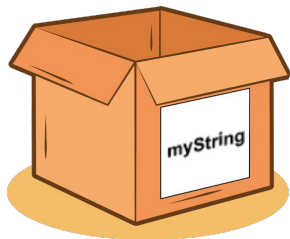
¿Cómo puedo guardar un tipo de dato?

- “String”
- 1
- 1.0
- True

“encebollado”

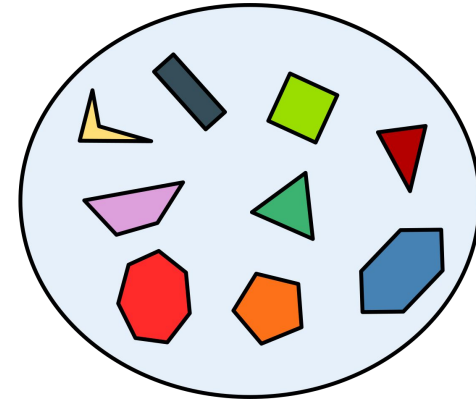
“bolon”

“fritada”



¿Qué es una estructura?

- Una colección de elementos.
- Nos permite agrupar datos
- Existen 4 principales estructuras Listas, Tuplas, Diccionarios y Conjuntos(set)



¿Qué es una lista?

- Estructura de dato que es **transformable**
- Conjunto de elementos **ordenados**
- Permite **duplicados**
- Sintaxis [elemento1, elemento2, elemento3, elemento4, elemento1]

elemento1	elemento2	elemento3	elemento4	elemento1
-----------	-----------	-----------	-----------	-----------

¿Qué es una Tupla?

- Estructura de dato que es **no transformable**
- Usada para fijar datos durante la ejecución de un programa
- Conjunto de elementos **ordenados**
- Permite **duplicados**
- Sintaxis (elemento1, elemento2, elemento3, elemento4, elemento1)

elemento1	elemento2	elemento3	elemento4	elemento1
-----------	-----------	-----------	-----------	-----------

¿Qué es un Set?

- Estructura de dato que es **transformable**
- Conjunto de elementos **no ordenados**
- **No Permite duplicados**
- Sintaxis { elemento1, elemento2, elemento3, elemento4, elemento1 }



elemento1	elemento3	elemento2	elemento4	elemento1
-----------	-----------	-----------	-----------	-----------

¿Qué es un Set?

- Estructura de dato que es **transformable**
- Conjunto de elementos **no ordenados**
- **No Permite duplicados**
- Sintaxis { elemento1, elemento2, elemento3, elemento4 }

elemento4	elemento3	elemento2	elemento1
-----------	-----------	-----------	-----------

¿Qué es un Diccionario?

- Estructura de dato que es **transformable**
- Estructura de dato que contiene un pareja datos única
- Pareja hecha de llave(key) y valor(value)
- Facilita la recuperacion de informacion
- Conjunto de elementos **ordenados** desde la versión de python 3.6
- **No Permite duplicados (Pareja)**
- Sintaxis {Key1 : Value1, Key2 : Value2}

Key1	Value1
Key2	Value2

Terminología y sintaxis de estructuras en Python

Listas

`a = [elemento1, elemento2, elemento3, element4, elemento1]`

elemento1	elemento2	elemento3	elemento4	elemento1
-----------	-----------	-----------	-----------	-----------

Tuples

`a = (elemento1, elemento2, elemento3, elemento4, elemento5)`

elemento1	elemento2	elemento3	elemento4	elemento5
-----------	-----------	-----------	-----------	-----------

Conjuntos(Set)

`a = { elemento1, elemento2, elemento3, elemento4 }`

elemento4	elemento3	elemento2	elemento1
-----------	-----------	-----------	-----------

Diccionarios

`a = {Key1 : Value1, Key2 : Value2}`

Key1	Value1
Key2	Value2

¿Cómo acceder a las estructuras?

- Listas
- [index]

a = [elemento1, elemento2, elemento3, elemento4, elemento1]

a[2]

Terminal: elemento3

elemento1	elemento2	elemento3	elemento4	elemento1
0	1	2	3	4

¿Cómo acceder a las estructuras?

- Tuples
- [index]

a = (elemento1, elemento2, elemento3, elemento4, elemento5)

a[2]

Terminal: elemento3

elemento1	elemento2	elemento3	elemento4	elemento5
0	1	2	3	4

¿Cómo acceder a las estructuras?

- Conjunto
- [index]

Conjuntos(Set)

a = { elemento1, elemento2, elemento3, elemento4 }

a[1]

elemento3

elemento4	elemento3	elemento2	elemento1
0	1	2	3

¿Cómo acceder a las estructuras?

- Conjunto
- No llevan un orden
- Elementos pueden ser recuperados con iteraciones sobre los elementos de este
- [index]

Conjuntos(Set)

`a = { elemento1, elemento2, elemento3, elemento4 }`

elemento4	elemento3	elemento2	elemento1
-----------	-----------	-----------	-----------

¿Cómo acceder a las estructuras?

- Diccionarios
- [llave]

```
a = {llave1 : Value1, llave2 : Value2}
```

```
a[llave1]
```

```
Value1
```

llave1	Value1
llave2	Value2

¿Cómo acceder a las estructuras?

- Listas
- Rebanada
- Slicing / rebanar



¿Cómo acceder a las estructuras?

- Listas
- Rebanada
- Slicing / rebanar
- [inicio(inclusivo):fin(exclusivo)]

a = [elemento1, elemento2, elemento3, elemento4, elemento1]

a[0:2]

Terminal: [elemento1, elemento2]

elemento1	elemento2	elemento3	elemento4	elemento1
0	1	2	3	4

¿Cómo acceder a las estructuras?

- Listas
- Rebanada
- Slicing / rebanar
- [inicio(inclusivo):fin(exclusivo)]

`a = [elemento1, elemento2, elemento3, elemento4, elemento1]`

`a[0::]`

Terminal: `[elemento1, elemento2, elemento3, elemento4, elemento1]`

elemento1	elemento2	elemento3	elemento4	elemento1
0	1	2	3	4

¿Cómo acceder a las estructuras?

- Listas
- Rebanada
- Slicing / rebanar
- [inicio(inclusivo):fin(exclusivo)]
- index negativos

`a = [elemento1, elemento2, elemento3, elemento4, elemento5]`

`a[-1]`

Terminal: elemento5

elemento1	elemento2	elemento3	elemento4	elemento5
-5	-4	-3	-2	-1

¿Cómo acceder a las estructuras?

- Listas
- Rebanada
- Slicing / rebanar
- [inicio(inclusivo):fin(exclusivo)]
- index negativos

`a = [elemento1, elemento2, elemento3, elemento4, elemento5]`

`a[-3:-1]`

Terminal: `[elemento3, elemento4]`

elemento1	elemento2	elemento3	elemento4	elemento5
-5	-4	-3	-2	-1

Añadir y remover elementos

- Listas - añadir

```
varios_numeros = []  
  
varios_numeros.append(5)  
varios_numeros.append(4)  
varios_numeros.append(3)  
print(varios_numeros)  
>>> [5, 4, 3]
```

5

5	4
---	---

5	4	3
---	---	---

- Diccionarios - añadir

```
capitales = {}
```

```
capitales['Ecuador'] = 'Quito'  
capitales['Colombia'] = 'Bogota'  
capitales['Peru'] = 'Lima'  
print(capitales)  
>>> {'Ecuador': 'Quito', 'Colombia': 'Bogota', 'Peru': 'Lima'}
```

Añadir y remover elementos

- Listas - remover

```
varios_numeros = [2, 4, 8, 10]
```

```
del varios_numeros[2]
```

```
print(varios_numeros)  
>>> [2, 4, 10]
```

- Diccionarios - remover

```
países_capitales = {'Ecuador': 'Quito', 'Peru': 'Lima'}
```

```
del capitales['Peru']
```

```
print(países_capitales )  
>>> {'Ecuador': 'Quito'}
```

Bucles con estructuras

Listas

```
varios_numeros = [2, 4, 8, 10]
```

```
for indice in range(len(varios_numeros)):  
    print('Numero:', varios_numeros[indice])
```

```
Numero: 2  
Numero: 4  
Numero: 8  
Numero: 10
```

```
for numero in varios_numeros:  
    print('Numero:', numero)
```

```
Numero: 2  
Numero: 4  
Numero: 8  
Numero: 10
```

Bucles con estructuras

Diccionario - Bucle por keys

```
países_capitales = {'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota'}
```

```
for país in países_capitales:  
    print('País:', país)
```

```
País: Ecuador  
País: Peru  
País: Colombia
```

```
for país in países_capitales.keys():  
    print('País:', país)
```

```
País: Ecuador  
País: Peru  
País: Colombia
```

Bucles con estructuras

Diccionario - Bucle por valores

```
capitales = {'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota'}
```

```
for capital in pais.es.capitales.values():  
    print('Capital:', capital)
```

Capital: Quito

Capital: Lima

Capital: Bogota

Bucles con estructuras

Diccionario - Bucle por keys y valores

```
capitales = {'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota'}
```

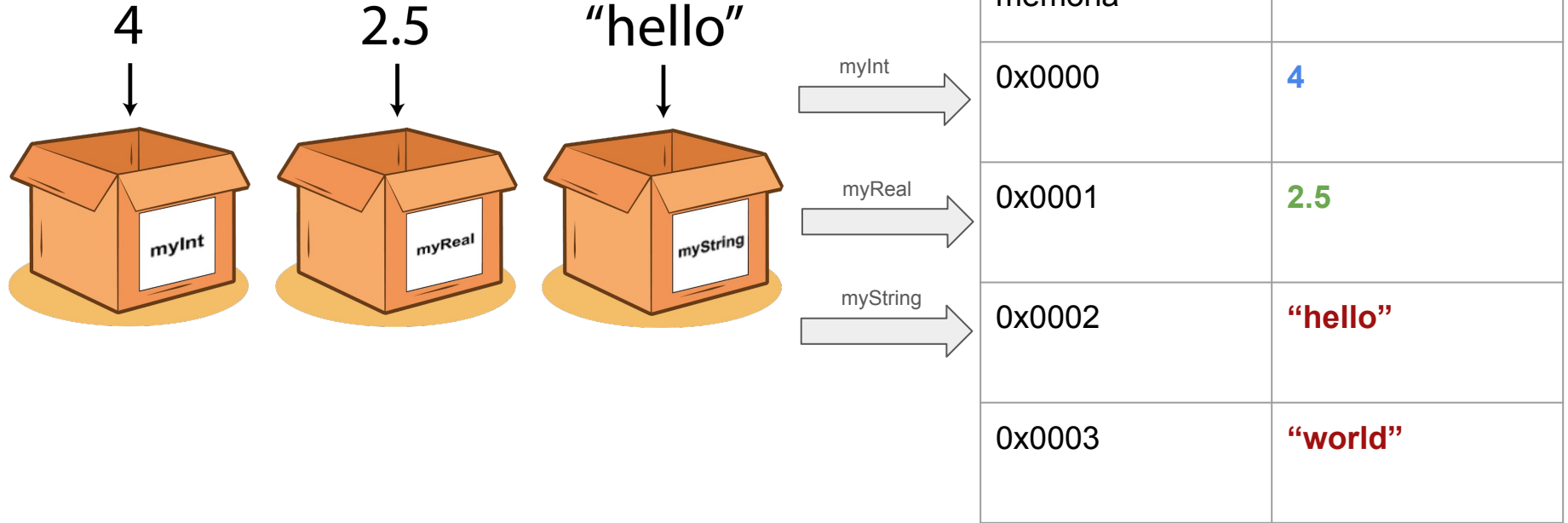
```
for pais, capital in pais_capitales.items():  
    print('Tupla pais y capital:', pais, capital)
```

```
Tupla pais y capital: ('Ecuador', 'Quito')  
Tupla pais y capital: ('Peru', 'Lima')  
Tupla pais y capital: ('Colombia', 'Bogota')
```

```
for pais, capital in pais_capitales.items():  
    print('Pais:', pais, '- Capital:', capital)
```

```
Pais: Ecuador - Capital: Quito  
Pais: Peru - Capital: Lima  
Pais: Colombia - Capital: Bogota
```

Clonación



Clonación

```
primera variable = 10
otra variable = primera variable
otra variable = otra variable + 1
print('primera_variable:', primera_variable)
```

```
primera_variable: 10
```

```
primera variable = [1, 2, 3]
otra variable = primera_variable
otra variable.append(4)
print('primera_variable:', primera_variable)
```

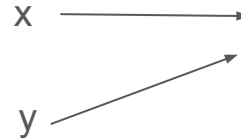
```
primera_variable: [1, 2, 3, 4]
```


Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

x = 10

y = x



Ubicación en memoria	Valores
0x0000	10
0x0001	
0x0002	

Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

x = 10

y = x

y = y + 1

x →

y →

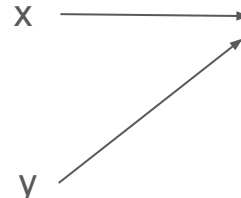
Ubicación en memoria	Valores
0x0000	10
0x0001	11
0x0002	

Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

x = [1, 2, 3]

y = x



Ubicación en memoria	Valores
0x0000	[1, 2, 3]
0x0001	
0x0002	

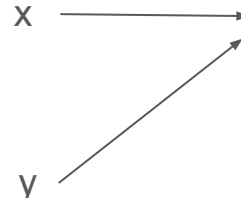
Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

```
x = [1, 2, 3]
```

```
y = x
```

```
y.append(4)
```



Ubicación en memoria	Valores
0x0000	[1, 2, 3, 4]
0x0001	
0x0002	

Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

```
x = [1, 2, 3]
```

```
y = x.copy()
```

x →

y →

Ubicación en memoria	Valores
0x0000	[1, 2, 3]
0x0001	[1, 2, 3]
0x0002	

Clonación

- Estructura de tipo de datos pueden ser clonados
- Variables se asignan al mismo de estructura de dato (alias)
- Dependiendo del tipo de dato las modificaciones se aplican a la misma estructura o a una copia

```
x = [1, 2, 3]
```

```
y = x.copy()
```

```
y.append(4)
```

x →

y →

Ubicación en memoria	Valores
0x0000	[1, 2, 3]
0x0001	[1, 2, 3, 4]
0x0002	

Conclusión

- Multi datos pueden ser guardados como una colección con la ayuda de estructuras
- Existen 4 principales estructuras de datos en python
- Estructuras syntax [], (), { }, {key:Value}
- Indexeo de estructuras
- Cómo utilizar bucles para recuperar datos de las estructuras
- Existen diferencias en la asignación de tipos de datos y tipos de estructuras de datos
- Como clonar correctamente la estructura para evitar problemas de asignación

Retroalimentación

- Para retroalimentación dirigirse al siguiente enlace <https://forms.gle/zUL4jQGkGMTXgdAQ9> .
- Déjanos saber qué podemos hacer para mejorar el curso

