KIBERNUM
IT Academy

Módulo 2 – Fundamentos de Programación Python

Estructuras de datos en Python

Objetivos

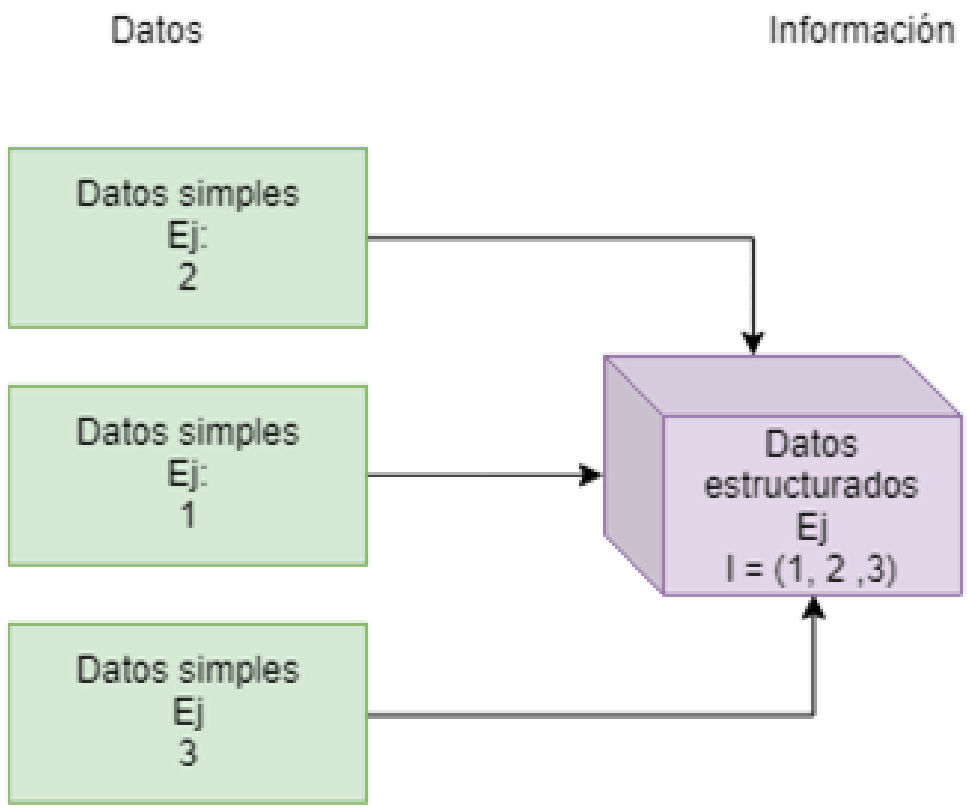
- Reconocer estructuras de datos en Python.

Contenido

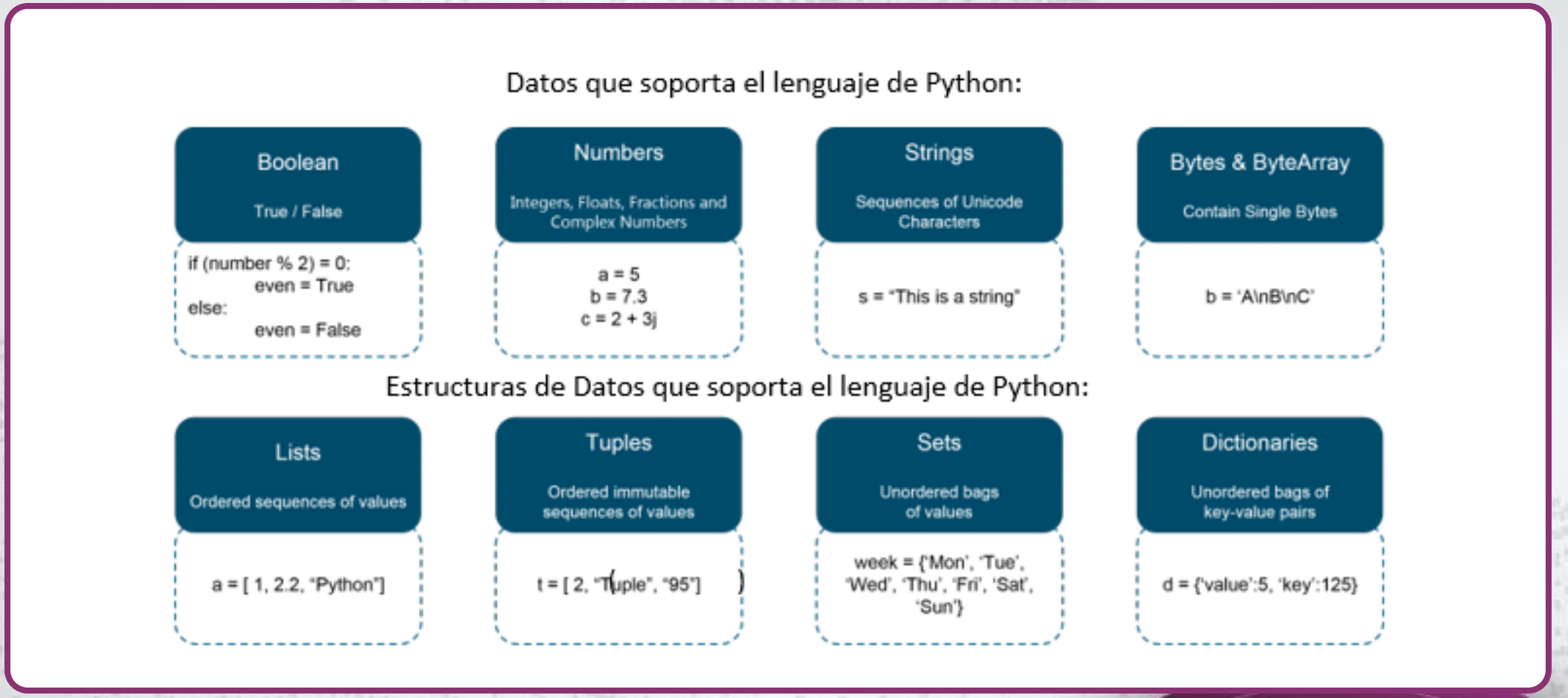
1. Listas.
2. Diccionarios.
3. Tuplas.
4. Sets.
5. Strings.

KIBERNUM

Estructuras de Datos en Python



Datos en Python



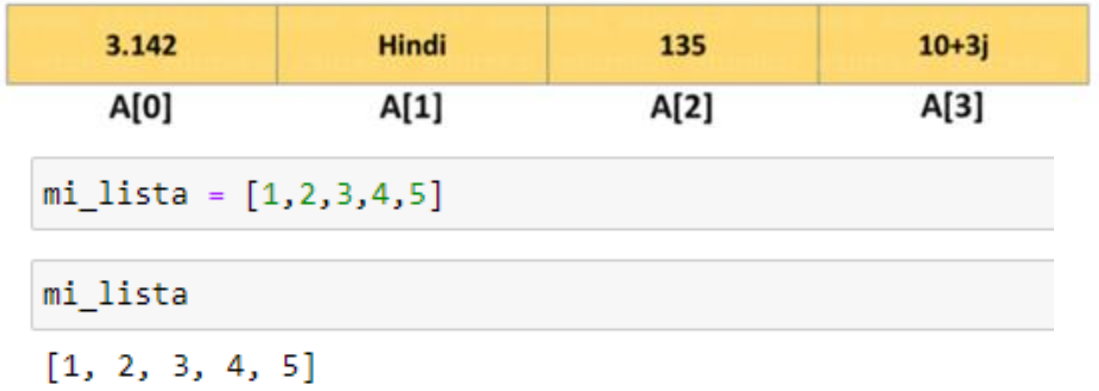
Listas en Python

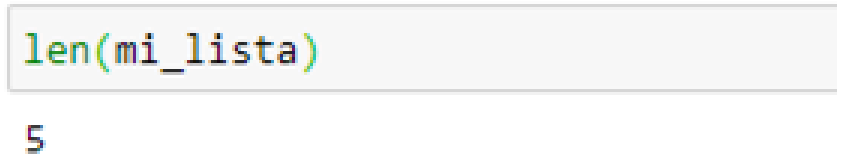
Se implementa una lista para almacenar la secuencia de varios tipos de datos. Es el tipo de datos más común y confiable.

Sintaxis:
nombre_list = [arg1, arg2,...]

Existen varias funciones para trabajar con listas. Por ejemplo : len() para obtener el número total de elementos en una lista.

Sintaxis:
len(nombre_list)





Creando una Lista

A continuación, algunos ejemplos de creación de listas.

Listas de cadenas de texto:

```
input_string = input ("Ingrese nombres separados por coma:")
family_list = input_string.split(",")
print("Mostrando todos los nombres:", family_list)
```

```
Ingrese nombres separados por coma: Silvia, Maria, Luis
Mostrando todos los nombres: ['Silvia', ' Maria', ' Luis']
```

Listas de números:

```
input_string = input ("Ingrese números separados por espacio:")
number_list = input_string.split()
print("todos los elementos ingresados:", number_list)
```

```
Ingrese números separados por espacio: 3 4 6
todos los elementos ingresados: ['3', '4', '6']
```

Creando una Lista

También podemos agregar elementos a una lista existente:

Sea la siguiente lista:

```
mi_lista = [1,2,3,4,5]
```

```
mi_lista
```

```
[1, 2, 3, 4, 5]
```

Se anexa un nuevo valor a la lista:

```
# agregar valores a una lista
mi_lista.append(6)
```

```
mi_lista
```

```
[1, 2, 3, 4, 5, 6]
```

Listas en Python

Podemos modificar los valores dentro de una lista.

Asimismo, crear una lista donde algunos de sus elementos contienen otras listas.

Podemos reasignar valores a una lista:

```
In [35]: my_list[0] = 'NEW'

In [98]: my_list
Out[98]: ['NEW', 'b', 'c', 'd']
```

Podemos crear listas dentro de otras listas

```
In [99]: nest = [1,2,3,[4,5,['target']]]

In [100]: nest[3]
Out[100]: [4, 5, ['target']]
```

Referenciando de esta forma los elementos

```
In [101]: nest[3][2]
Out[101]: ['target']

In [102]: nest[3][2][0]
Out[102]: 'target'
```

Tuplas en Python

Las tuplas son variables donde se almacenan diferentes tipos de datos estructurados que deben estar ordenados desde el comienzo.
Las tuplas son inmutables, no se pueden modificar durante la ejecución del programa. Por ende, al crearla debes colocar, sí o sí, los elementos que se van a almacenar.

Sintaxis:
nombre_tupla = (arg1, arg2,…)

```
# son estructuras inmutables, se crean con tamaño fijo
t = (78.0987273,-37.09876233)
```

```
t[0]
```

```
78.0987273
```

```
t[0] = 0.0
```

```
---------------------------------------------------------------------------
--------
TypeError                                 Traceback (most recent ca
ll last)
<ipython-input-99-b414e7829fbd> in <module>
----> 1 t[0] = 0.0

TypeError: 'tuple' object does not support item assignment
```

KIBERNUM

Cortando Listas y Tuplas

- Podemos referenciar una porción (slice) de la lista o tupla:

- También, a la hora de mostrar una Lista o Tupla en Python podemos indicarle que nos la muestre completa excepto los últimos elementos de ella.
  Indicando, como final, un número negativo.

```
In [30]: my_list
Out[30]: ['a', 'b', 'c', 'd']

In [33]: my_list[1:]
Out[33]: ['b', 'c', 'd']

In [34]: my_list[:1]
Out[34]: ['a']
```

¿Qué pasa si...?

```
s="qwertyuiop"

s[0:3]
s[-2:-5]
s[-5:-2]
```

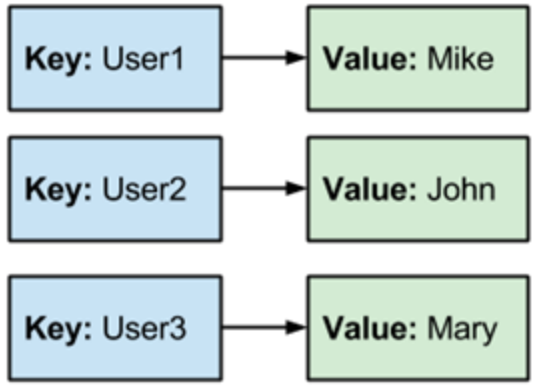
KIBERNUM

Diccionarios en Python

- El tipo de dato diccionario (tipo dict) en Python es un tipo de dato compuesto. Cada clave del diccionario está separada de su valor por dos puntos “:”, sus elementos separados por comas y todo va delimitado con llaves {}.
- Las claves son únicas en los diccionarios, pero el valor no tiene por qué serlo.

```
[28] dicc1={100:"Python",200:7,300:"Hola"}
     dicc1
```

```
[30] dicc1={100:"Python",200:7,300:"Hola"}
     dicc1[200]
     type(dicc1[300])
```



Diccionarios en Python

```
# población de los países
poblacion = [18, 40, 14, 200]
```

```
poblacion
```

```
[18, 40, 14, 200]
```

```
paises = { 'CL': 18, 'AR': 40, 'PE': 14, 'BR':200 }
```

```
paises
```

```
{'CL': 18, 'AR': 40, 'PE': 14, 'BR': 200}
```

```
paises['PE']
```

```
14
```

En el siguiente ejemplo, se desea almacenar la población de Chile, Argentina, Perú y Brasil. Una estructura de tipo “lista” podría ser utilizado, pero nos obligaría a saber qué país utiliza, qué posición del listado. Los diccionarios, por otra parte, serían de mayor utilidad en este caso pues son colecciones de elementos asociativos. Podemos asignar valores y obtenerlos por llave o nombre.

Diccionarios en Python

- Los diccionarios, pueden a su vez almacenar cualquier tipo de elemento, sean incluso listas u otros diccionarios. Una buena elección en el diseño de la estructura nos permite resolver el caso de uso de mejor forma.

```
agenda = { 'Juan Perez':'+569922332', 'Jose Soza':'+5699943322' }
```

```
agenda['Juan Perez']
```

```
'+569922332'
```

```
c1 = {'Fono1': '5553322', 'Fono2':'2846126', 'Mail': 'j@yahoo.com'}
c2 = {'Fono1': '5334432', 'Fono2':'3223412', 'Mail': 'p@yahoo.com'}
c3 = {'Fono1': '2338932', 'Fono2':'4267841', 'Mail': 'u@yahoo.com'}

agenda = {'Juan': c1, 'Pepe': c2, 'Luis': c3}
```

```
agenda['Pepe']['Fono2']
```

```
'3223412'
```

¿Qué pasa si?

```
d={‘k1’:{‘kint’:[1,2,3]}}
¿cómo obtengo 1?
```

KIBERNUM

Sets en Python

Los Sets son colecciones de elementos únicos, y cuando se puede, ordenados.

```
{1,2,3}
```

```
{1, 2, 3}
```

```
{1,2,3,1,2,1,2,3,3,3,3,2,2,2,1,1,2}
```

```
{1, 2, 3}
```

Listas de cadenas de texto:

```
A = {1,2,3}
```

```
A.add(4)
```

```
print(A)
```

```
{1, 2, 3, 4}
```

Sets en Python

- Con esta estructura (sets) se puede realizar operaciones de conjuntos.

```
A = {1,2,3}
B = {3,4,5,6}
```

```
A.union(B)
```

```
{1, 2, 3, 4, 5, 6}
```

```
A.intersection(B)
```

```
{3}
```

```
A.difference(B)
```

```
{1, 2}
```

```
A - B
```

```
{1, 2}
```

Strings en Python

- Las cadenas de caracteres (o strings) son un tipo de dato compuesto por secuencias de caracteres que representan texto.
- Estas cadenas de texto son de tipo **str** y se delimitan mediante el uso de comillas simples o dobles.
- Cada uno de los caracteres de una cadena (incluidos los espacios) tienen asignado un índice.

```
[14] cadena="Python"
     print(cadena)
```

Strings en Python

Las cadenas de caracteres, al igual que las listas, están indexadas.

Esto significa que se puede acceder a un elemento de la cadena ya sea mediante el índice o el índice reverso.

Caracteres :	P	y	t	h	o	n
Índice :	0	1	2	3	4	5
Índice inverso :	-6	-5	-4	-3	-2	-1

```
[13] cadena="Python"
     print(cadena[0])
     print(cadena[-1])
```

Dudas y consultas

KIBERNUM

Fin presentación

KIBERNUM