Parte 1. Fundamentos de la programación

Unai Pérez-Goya
Área de Lenguajes y Sistemas Informáticos

Curso 2024/2025

Parte 1.

Fundamentos de la programación

Sesión VIII: Collecciones (`set` eta `dict`)

Tabla de contenidos

Resumen de sesiones anteriores

Contenido de la sesión I

Lista de contenidos

1. Introducción a Python. Historia y características;
2. Lenguajes de programación. Concepto y uso;
3. Lenguajes interpretados *vs.* compilados. Programas ejecutables. Máquina intérprete;
4. Entorno de desarrollo integrado (IDE)

Contenido de la sesión II

- Variables y uso básico en pytho
- Declaración y asignación;
- Tipos
  - Numéricos
  - Texto
  - Booleanos
- Constantes

Contenido de la sesión III

- Estructura de los programas
- Estructuras de control
  1. `if` egitura;
  2. `if-else` egitura;
- Funciones, capsulación y ámbito de la variable
- Ejercicios.

Contenido de la sesión IV

- Interacción con el usuario;
- Estructura `if-elif`.
- Estructura `if-elif-else`.
- Funciones sobre variables de tipo `str`

Contenido de la sesión V

- Datos estructurados
- Uso de `list`
- Recorrer listas con `for [each]`
- Ejercicios

Resumen de la sesión VI

- Programas iterativos
- Uso de `for` con `range`
- Uso de `while`
- Ejercicios

Contenido de la sesión VII

- Repaso
- Recursividad
- Ejercicios recursividad
- Ejercicios de repaso

Resumen

Repaso: Datos estructurados

- Un tipo de dato estructurado es aquel que permite almacenar más de un dato.
- *Existen diferentes interpretaciones para los datos que deben mantener el orden*:
  - Vector como secuencia de números.El orden será la posición de la secuencia.
  - Matriz como tabla de números, donde el orden de los elementos está dado por filas y columnas.
  - ...
- Aprenderemos varios tipos de datos con diferentes tipos de estructuras.
  - La idea común es almacenar/acceder a más de un dato en una misma estructura.

//bground.png

Repaso: Datos estructurados

- Listas de valores (`list`);
- Grupos de valores (`set`);
- Diccionarios (`dict`);
- +
- Matrices;

Conjuntos (`set`)

Conjuntos

Conjuntos matemáticos

- Un conjunto es una colección de datos que no siguen el orden ni se repiten.
  a) ¡No están ordenados!
  b) ¡Los elementos no se pueden repetir!
- No existe el concepto de orden, ni primer ni último elemento.
- Los elementos pertenecen o no pertenecen a un conjunto
- En python, podremos usar la analogia de una bolsa donde introducimos o retiramos elementos

Conjuntos

Crear un conjunto

- Un conjunto (`set`) se puede genera asignando los valores que contiene entre corchete ({,}).
- Es decir:
  - Declarar una variable: `a=5`;
  - Declarar un conjunto vacio: `a=set()`
  - Declarar un conjunto de enteros: `a={5,1,5,67,243,-4}`;
  - Declarar un conjunto de `float`: `a={-5.3,2.5}`;
  - Declarar un conjunto de cadenas de carácteres: `a={'hola','papel'}`;
  - ...
- Al ejecutar `type(a)`, python responderá `<class 'set'>`
  - Para empezar, suficiente.

Conjuntos

Crear un conjunto

- Un conjunto (`set`) se puede genera asignando los valores que contiene entre corchete ({,}).
- Es decir:
  - Declarar una variable: `a=5`;
  - Declarar un conjunto vacio: `a=set()`
  - Declarar un conjunto de enteros: `a={5,1,5,67,243,-4}`;
  - Declarar un conjunto de `float`: `a={-5.3,2.5}`;
  - Declarar un conjunto de cadenas de carácteres: `a={'hola','papel'}`;
  - ...
- Al ejecutar `type(a)`, python responderá `<class 'set'>`
  - Para empezar, suficiente.

Conjuntos

Operaciones entre conjuntos

- Un conjunto se puede mostrar utilizando la función `print`.
- `a={5,1,8,67,243,-4}`,
  - `print(a)` → `{5,1,8,67,243,-4}`;
- `a={1,1,1,3,3,3}`,
  - `print(a)` → `{1,3}`;
- `a=set('pamplona')`,
  - `print(a)` → `{'l','p','m','o','n','a'}`;

Conjuntos

Funciones sobre conjuntos

- Los conjuntos, como las funciones se gestionan con funciones
- Dado el conjunto `a={5,1,8,67,243,-4}`,
  - `print(a)` → `{5,1,8,67,243,-4}`;
- Función `add`: `a.add(4)`,
  - `print(a)` → `{5,1,4,8,67,243,-4}`;
- Función `remove`: `a.remove(8)`,
  - `print(a)` → `{5,1,4,67,243,-4}`;

Conjuntos

Operaciones entre conjuntos

- Para saber si un elemento está dentro de un conjunto utilizaremos el operador `in`.
- Dado el conjunto `a={5,1,67,243,-4}`.
  - `5 in a` → `True`;
  - `6 in a` → `False`;
  - `'hola' in a` → `False`.

Conjunto

Operaciones entre conjuntos

- Se pueden hacer operaciones entre conjuntos.
- El resultado será otro conjunto.
- Union entre conjunto:
  - Los elementos que pertenecen a cualquiera de los dos conjuntos
- Intersección entre conjuntos:
  - Los elementos que aparecen en ambos conjuntos
- Resta entre conjuntos:
  - Los elementos del primero después de eliminar los elementos del segundo

Conjunto

Multzoen gaineko eragiketak

- Se pueden hacer operaciones entre conjuntos.
- Ejemplo: (`a={1,3,5}` eta `b={3,5,7}` multzoekin)
- Union entre conjunto:
  - `c= a|b` → `c={1,3,5,7}`.
- Intersección entre conjuntos:
  - `c= a&b` → `c={3,5}`.
- Resta entre conjuntos:
  - `c= a-b` → `c={1}`.
  - `c= b-a` → `c={7}`.

Ejemplos sobre conjuntos

Ejemplo: crear un conjunto

✪ **Ejemplo 1:** Escribe un programa que cree un conjunto

Ejemplos sobre conjuntos

Ejemplo 1: Crear un conjunto 1

```
1 # Declarar el conjunto vacio
2 conjunto = set()
3 print(conjunto)
4
5 # Declarar el conjunto con valores
6 conjunto = {1,2,7}
7 print(conjunto)
8
```

Código: Crear un conjunto 1.

Ejemplos sobre conjuntos

Ejemplo 1: Crear un conjunto 2

```
1 # Crear un conjunto a partir de una lista
2 una_lista = [1,3,45,3,1,5,7]
3 conjunto = set(una_lista)
4 print(conjunto)
5 # Gerar un conjunto a partir de un str
6 texto = 'Hola Mundo!'
7 conjunto = set(texto)
8 print(conjunto)
```

Código: Crear un conjunto 2.

Ejemplos sobre conjuntos

Ejemplo: operaciones sobre conjuntos

- **Ejemplo 2** Escribe un programa que declare dos conjuntos y muestre su unión, intersección y resta

Conjunto

Ejemplo: operaciones sobre conjuntos

```
1 # declarar variables
2 conjunto_a = {1,2,7}
3 conjunto_b = {7,5,10}
4
5 # operazioak
6 interseccion = conjunto_a & conjunto_b
7 union = conjunto_a| conjunto_b
8 resta = conjunto_a - conjunto_b
```

Código: Ejemplo: operaciones sobre conjuntos.

Ejemplos sobre conjuntos

Ejemplo: añadir valores nuevos y existentes

- **Ejemplo 3** Nola gehitu ditzaket elementuak aurretik sortutako multzo batean? Zer gertatzen da errepikatutako elementuak gehitzen baditut?

Conjunto

Ejemplo: añadir valores nuevos y existentes

```
1  # declarar variables
2  conjunto_a = {1,1,2,7}
3  print(conjunto_a)
4
5  # elementuak gehitu
6  conjunto_a.add(3)
7  print(conjunto_a)
8
9  # elementuak gehitu
10 conjunto_a.add(2)
11 print(conjunto_a)
```

Código: Ejemplo: operaciones sobre conjuntos.

Conjunto

Ejemplo: añadir valores nuevos y existentes

```
1  # declarar variables
2  conjunto_a = {1,1,2,7}
3  print(conjunto_a)
4
5  # elementuak gehitu
6  conjunto_a = conjunto_a | set(6)
7  print(conjunto_a)
8
9  # elementuak gehitu
10 conjunto_a = conjunto_a | set(7)
11 print(conjunto_a)
```

Código: Ejemplo: operaciones sobre conjuntos.

Ejemplos sobre conjuntos

Ejemplo: añadir valores nuevos y existentes

✪ **Ejemplo 4** Escribe un porgrama que inserte nuevos valores a un conjunto. Haz que añada valores existentes al conjunto.

Ejemplos sobre conjuntos

Ejemplo: añadir valores nuevos y existentes

```
1  # declarar variables
2  conjunto_a = {1,1,2,7}
3  print(conjunto_a)
4
5  # elementuak gehitu
6  conjunto_a.add(2.2)
7  print(conjunto_a)
8
9  # elementuak gehitu
10 conjunto_a.add('1')
11 print(conjunto_a)
```

Código: Ejemplo: operaciones sobre conjuntos.

Ejemplos sobre conjuntos

Ejemplo: añadir valores nuevos y existentes

```
1  # declarar variables
2  conjunto_a = {1,1,2,7}
3  print(conjunto_a)
4
5  # elementuak kendu
6  conjunto_a.discard(1)
7  print(conjunto_a)
8
9  # elementuak kendu
10 conjunto_a = conjunto_a - {7}
11 print(conjunto_a)
```

Código: Ejemplo: operaciones sobre conjuntos.

Estructura `for` sobre conjuntos

- Se puede utilizar la estructura `for` con los conjuntos:

☞
```
for x in conjunto:
    instrucciones
```

- ... donde `conjuto` puede ser cualquier conjunto
- ... donde `x` nombre de variable definida por el programador
  - En cada iteración tomará el valor de un elemento del conjunto
  - Sin un orden concreto... primera iteración un elemento, segunda iteración otro elemento,...

Ejemplos sobre conjuntos

Ejercicio 1: hitzak kontatu

- **Ejercicio 1** Esaldi batean dauden hitz desberdin kopurua kontatzen dituen funtzioa sortu. Esaldiak puntuazio markak baditu zer gertatzen da?

Diccionarios (`dict`)

Concepto de diccionarios

- Un **diccionario**, o lista indexada, es una lista desorrdenada de elementos emparejados <clave:valor>.
- Es decir, el *valor* guardará la información del elemento, al que se le unirá una (*clave*) por la que se le conocerá dentro de la estructura.
- Las claves no se pueden repetir, y funcionan como los conjuntos.
  - Se puede decir que siguen el concepto de los diccionarios, cada palabra (*clave*) va unida a su descripción (el *valor*).

!! Las claves no se pueden repetir pero puede existir dos claves con el mismo valor.

Diccionarios

- Los diccionarios sirven para guardar información y recuperarla en cualquier momento.
- La búsqueda de información funciona automáticamente. Imagina que queremos guardar los datos de las personas.
  - Para que las personas sean únicas deberemos definir un identificador
  - Los números del DNI pueden ser el identificador o la clave.
  - Es decir, cada par del diccionario será <DNI, NOmbre>;
  - Así se puede obtener buscando el nombre de una persona, su número de DNI.
- Es muy utilizado por comodidad, sin embargo, no es nada eficiente en términos de computación.

Diccionarios

Crear un diccionario

- Se identifican con las mismas llaves que los conjuntos pero al asignarlos con valores se crean parejas.
- Para unir una clave y su valor se debe utilizar '`:`'.
- Por ejemplo: `personas={'12345678A':'Unai', '12345679A':'Alberto','12345679B':'Alfredo'}`.
- Se pueden combinar cualquier clave y valor .
  - Esto es un poco raro, en la mayoria de lenguajes de programación esto no es posible.

Diccionarios

Crear un diccionario

- Ejemplos:
  - `a={'juan':23,'pedro':33,5:232};`
  - `a={'juan':23,'pedro':23,5:23};`
  - `a={'juan':-5.6,'pedro':23,'alonso':'nada'};`

!! Si las claves se repiten, solo se guardará el último valor asignado, el antiguo será eliminado.

Diccionarios

Utilizando diccionarios

- Se pueden mostrar con `print`.
- `a={'juan':5,'pedro':44,'alonso':-4}` bada...

a) `print(a)`
   - → `{'juan':5,'pedro':44,'alonso':-4}`;

b) `print(a.keys())`
   - → `dict_keys(['juan','pedro','alonso'])`;

c) `print(list(a.keys()))`
   - → `['juan','pedro','alonso']`;

d) `print(list(a.values()))`
   - → `[5,44,-4]`;

Diccionarios

Utilizando diccionarios

- Para seleccionar uno por uno los elementos utilizaremos corchetes (`[,]`),
  - Como si fueran listas, pero utilizando la clave
- En lugar de utilizar la posición del elemento, utilizaremosclave.
- Dado el diccionario `a={'juan':5,'pedro':44,'alonso':-4}`.
  - `a[1]` → `Error!`;
  - `a['pedro']` → `44`;
  - `a['pedro']*2` → `88`;
  - `a['PEDRO']` → `44`;
  - `a['marcos']` → `Error!`.

Diccionarios

Utilizando diccionarios

- Para evitar errores se puede comprobar si existe una clave.
- Esto se puede hacer con el operador `in`
- Dado el diccionario `a={'juan':5,'pedro':44,'alonso':-4}`.
  - `'pedro' in a` → `True`;
  - `'PEDRO' in a` → `False`;
  - `'marcos' in a` → `False`;
  - `44 in a` → `False`.

Ejemplos sobre conjuntos

Ejemplo 1: declarar un diccionario

- **Ejemplo 1** Escribe un programa que declare un diccionario y muestre sus valores

Diccionarios Pythonen

Ejemplo 1: declarar un diccionario

```
1  # declarar variables
2  persona_1 = {'izena':'Juan', 'abizena':'perez'}
3  persona_2 =  {'izena':'Pedro', 'abizena':'sanchez'}
4
5  #  mostrar en pantalla
6  print(persona_1)
7  print(persona_1['izena'])
8  print(persona_2['izena'])
9  print(persona_1['abizena'])
10 print(persona_2['abizena'])
```

Código: declarar un diccionario.

Diccionarios Pythonen

Ejemplo 2: declarar un diccionario y modificarlo

✪ **Ejemplo 2** Crea un programa que declare un diccionario y lo modifique de alguna manera.

Diccionarios Pythonen

Ejemplo 2: declarar un diccionario y modificarlo

```
1  # declarar variables
2  grupo = {'a':'Juan', 'b':'Pedro'}
3
4  print(grupo['a'] )
5  print(grupo['b'] )
6
7  grupo['a'] = 'Martin'
8  print(grupo['a'] )
9
10 grupo['a'] = 27
11 print(grupo['a'] )
12 print(grupo)
```

Código: declarar un diccionario y modificarlo.

Diccionarios Pythonen

Ejemplo 3: declarar un diccionario y modificarlo

- **Ejemplo 3** Diccionarios bat esleitu eta elementuak ezabatzen dituen programa sortu.

Diccionarios Pythonen

Ejemplo 3: declarar un diccionario y modificarlo

```
1 # declarar variables
2 diccionario = {0:'cero',1:'uno',2:'dos',3:'tres'}
3
4 print(d)
5 print(d[1])
6
7 del d[1]
8 print(d)
9 print(d[1])
```

Código: declarar un diccionario y modificarlo.

Diccionarios Pythonen

Ejemplo 4: declarar un diccionario y modificarlo

- **Ejemplo 4:** crear un programa que declare un diccionario y elimine elementos del mismo. Esta vez las claves tendrán valores diferentes.

Diccionarios Pythonen

Ejemplo 4: declarar un diccionario y modificarlo

```
1  # declarar variables
2  diccionario = {0:'zero',1:'bat',2:'bi',3:'hiru'}
3
4  print(d)
5  print(d[1])
6
7  del d[1]
8  print(d)
9  if 1 in d.keys():
10   print(d[1])
11 else:
12   print('No existe el valor')
```

Código: declarar un diccionario y modificarlo.

Diccionarios Pythonen

Ejemplo 5: declarar un diccionario y añadir elementos

- **Ejemplo 5** Crear un programa que declare un diccionario, elimine elementos y añada otros nuevos. Además, en esta ocasión los valores del diccionario tendrán diferentes tipos.

Diccionarios Pythonen

Ejemplo 5: declarar un diccionario y modificarlo

```
1  # declarar variables
2  diccionario = {0.1:'zero',1:'bat',2:'bi','Unai':'hiru'}
3
4  print(diccionario)
5  print(diccionario[1])
6
7  del diccionario[1]
8  print(diccionario)
9  if 1 in diccionario.keys():
10   print(diccionario[1])
11 else:
12   diccionario[1] = 'nuevo valor'
13
14 print(diccionario)
```

Código: declarar un diccionario y modificarlo.

Ejercicios

Ejercicio 1

- **Ejercicio 1.** Escribe una función que solicite al usuario un conjunto de 5 números enteros. Los números deberán pedirse al usuario y no podrán repetirse. Crea dos funciones diferentes: en una se utilizará la función `len` , y en la otra no se utilizará la función `len` .

Ejercicio 2

- **Ejercicio 2.** Escribe un programa que solicite dos conjuntos de 5 números enteros. Antes de que el programa finalice, mostrará la intersección, la suma y la diferencia entre los dos conjuntos.

Ejercicio 3

- ✪ **Ejercicio 3.** Escribe un *programa* que cree un diccionario de personas. Los datos de una persona consisten en su nombre y su número de DNI, ambos deberán solicitarse al usuario. Así, el *programa* deberá pedir el nombre de la persona y su número de DNI cada vez que se quiera añadir un nuevo elemento al diccionario. El proceso finalizará cuando el usuario introduzca el código `0`.
- ☆ **Bonus:** Asegúrate de que el proceso anterior no permita introducir valores negativos ni valores repetidos (puedes usar `keys()`).

Ejercicio 4

- **Ejercicio 4.** Completa el ejercicio anterior para poder ver los datos de las personas en el diccionario. Para ello, el programa deberá tener 2 opciones: añadir un nuevo usuario y buscar a una persona. Al utilizar la opción de buscar, mostrará todos los DNI y dará la opción de pedir el número de DNI de una persona y mostrar su nombre.