



# 111- [PF] - Lab - Lista, tupla, diccionario

## Ejercicio 1: Presentar el tipo de dato de lista

### Acceso al IDE de AWS Cloud9

1. Para activar el entorno de su laboratorio, desplácese hasta la parte superior de estas instrucciones y seleccione **Start Lab** (Iniciar laboratorio).

Se abrirá el panel **Start Lab** (Iniciar laboratorio), donde se muestra el estado del laboratorio.

2. Espere hasta que aparezca el mensaje *Lab status: ready* (Estado del laboratorio: listo) y, a continuación, para cerrar el panel **Start Lab** (Iniciar laboratorio), haga clic en la **X**.

3. En la parte superior de estas instrucciones, elija **AWS**.

AWS Management Console se abrirá en una pestaña nueva del navegador. El sistema iniciará la sesión de forma automática.

Nota: Si no se abre una pestaña nueva del navegador, generalmente aparece un anuncio o un icono en la parte superior de este, el cual indica que el navegador no permite

que se abran ventanas emergentes en el sitio. Haga clic en el anuncio o en el icono, y elija Allow pop ups (Permitir ventanas emergentes).

4. En AWS Management Console, elija **Services** (Servicios) > **Cloud9**. En el panel **Your environments** (Sus entornos), busque la tarjeta **reStart-python-cloud9** y elija **Open IDE** (Abrir IDE).

Se abre el entorno de AWS Cloud9.

Nota: Si se abre una ventana emergente con el mensaje `.c9/project.settings have been changed on disk` (Se ha modificado la configuración de `.c9/project.` en el disco), elija Discard (Descartar) para ignorarlo. Del mismo modo, si una ventana de diálogo le pide que Show third-party content (Muestre contenido de terceros), elija No para rechazar la indicación.

---

## Creación del archivo de ejercicio de Python

1. En la barra de menú, elija **File > New From Template > Python File** (Archivo > Nuevo a partir de plantilla > Archivo en Python).  
Esta acción crea un archivo sin título.
2. Elimine el código de muestra que obtiene del archivo de plantilla.
3. Elija **File > Save As...** (Archivo > Guardar como...), proporcione un nombre adecuado para el archivo de ejercicio (por ejemplo, *collections.py*) y guárdelo en el directorio **/home/ec2-user/environment**.

---

## Acceso a la sesión del terminal

1. En su IDE de AWS Cloud9, elija el icono + y seleccione **New Terminal** (Nuevo terminal).

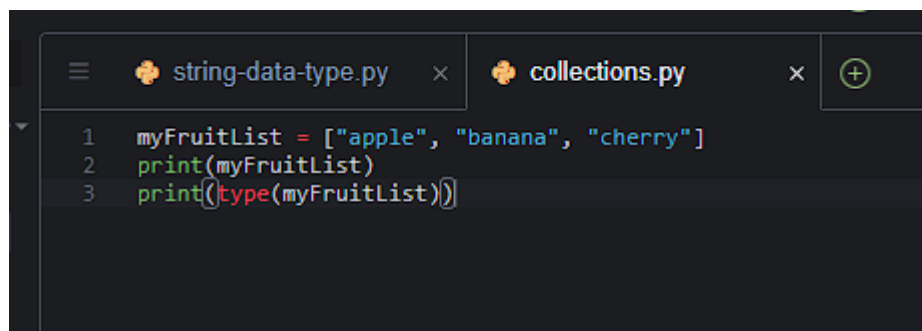
Se abre una sesión de terminal.

2. Para ver el directorio en el que está trabajando actualmente, escriba `pwd`. Este comando lleva a `/home/ec2-user/environment`.
3. En este directorio, también debería ser capaz de localizar el archivo que creó en la sección anterior.

## Definición de una lista

En esta actividad, editará un script en Python para almacenar una colección de nombres de frutas o una lista.

1. En el panel de navegación del IDE, elija el archivo `.py` que creó en la sección *Creación del archivo de ejercicios de Python* anterior.
2. En el archivo, escriba el siguiente código:

A screenshot of a code editor interface. At the top, there are two tabs: 'string-data-type.py' and 'collections.py'. The 'collections.py' tab is active. The code in the editor is as follows:

```
1 myFruitList = ["apple", "banana", "cherry"]
2 print(myFruitList)
3 print(type(myFruitList))
```

```
myFruitList = ["apple", "banana", "cherry"]
print(myFruitList)
print(type(myFruitList))
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

## Acceso a una lista por posición

Puede acceder al contenido de una lista por su posición. En esta actividad, mostrará cada elemento de nuestra lista por su posición:

1. En los lenguajes de programación, el posicionamiento en una lista comienza en el cero (0). Los corchetes indican a Python qué posición en la lista desea. Para acceder a la cadena `apple`, escriba el siguiente código:

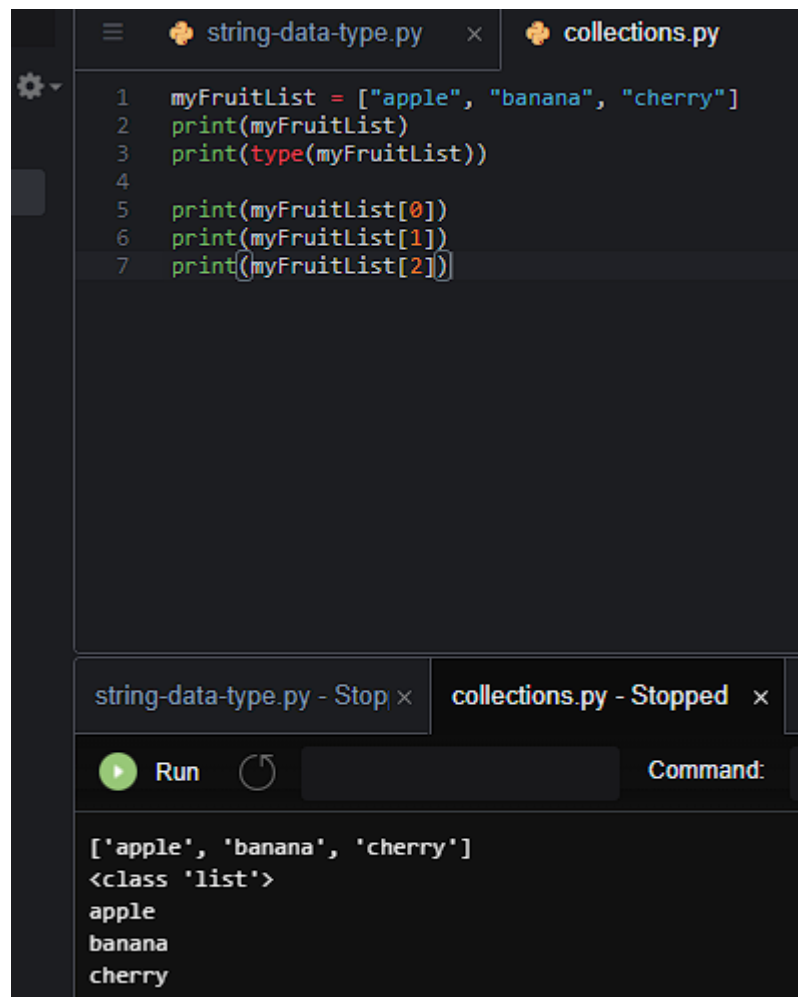
```
print(myFruitList[0])
```

1. Para acceder a la cadena `banana`, escriba lo siguiente:

```
print(myFruitList[1])
```

1. Para acceder a la cadena `cherry`, escriba el siguiente código:

```
print(myFruitList[2])
```



```
string-data-type.py x collections.py
1 myFruitList = ["apple", "banana", "cherry"]
2 print(myFruitList)
3 print(type(myFruitList))
4
5 print(myFruitList[0])
6 print(myFruitList[1])
7 print(myFruitList[2])

string-data-type.py - Stop x collections.py - Stopped x
Run Command:
['apple', 'banana', 'cherry']
<class 'list'>
apple
banana
cherry
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

## Modificación de los valores de una lista

Los valores de una lista se pueden cambiar. En esta actividad, cambiará `cherry` por `orange`.

1. En Python, el posicionamiento en la lista comienza en cero (0), por lo que tiene que utilizar el número 2 para acceder a la tercera posición. Escriba el siguiente código:

```
myFruitList[2] = "orange"
```

1. Muestre la lista actualizada:

```
print(myFruitList)
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']
```

The screenshot shows a Python IDE with two tabs: 'string-data-type.py' and 'collections.py'. The 'string-data-type.py' tab is active, displaying the following code:

```
1 myFruitList = ["apple", "banana", "cherry"]
2 print(myFruitList)
3 print(type(myFruitList))
4
5 print(myFruitList[0])
6 print(myFruitList[1])
7 print(myFruitList[2])
8 myFruitList[2] = "orange"
9 print(myFruitList)
10
```

Below the code editor, there are tabs for 'string-data-type.py - Stop' and 'collections.py - Stopped'. A 'Run' button is visible. The output console shows the following results:

```
['apple', 'banana', 'cherry']
<class 'list'>
apple
banana
cherry
['apple', 'banana', 'orange']
```

## Ejercicio 2: Presentar el tipo de dato de tupla

### Definición de una tupla

Una tupla es similar a una lista, pero no se puede cambiar. Un tipo de dato que no se puede cambiar después de su creación se conoce como *immutable*. Para definir una tupla, se utilizan paréntesis en lugar de corchetes (`[]`).

1. Cree una tupla escribiendo el siguiente código:

```
myFinalAnswerTuple = ("apple", "banana", "pineapple")
print(myFinalAnswerTuple)
print(type(myFinalAnswerTuple))
```

```
10
11 myFinalAnswerTuple = ("apple", "banana", "pineapple")
12 print(myFinalAnswerTuple)
13 print(type(myFinalAnswerTuple))
14 print(myFinalAnswerTuple[0])
```

string-data-type.py - Stop x collections.py - Stopped x (+)

Run Command: collections.py

```
apple
banana
cherry
['apple', 'banana', 'orange']
('apple', 'banana', 'pineapple')
<class 'tuple'>
apple
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

## Acceso a una tupla por posición

Al igual que con una lista, también se puede acceder a los elementos de una tupla por su posición:

1. Para acceder a la cadena `apple`, escriba el siguiente código:

```
print(myFinalAnswerTuple[0])
```

1. Para acceder a la cadena `banana`, escriba el siguiente código:

```
print(myFinalAnswerTuple[1])
```

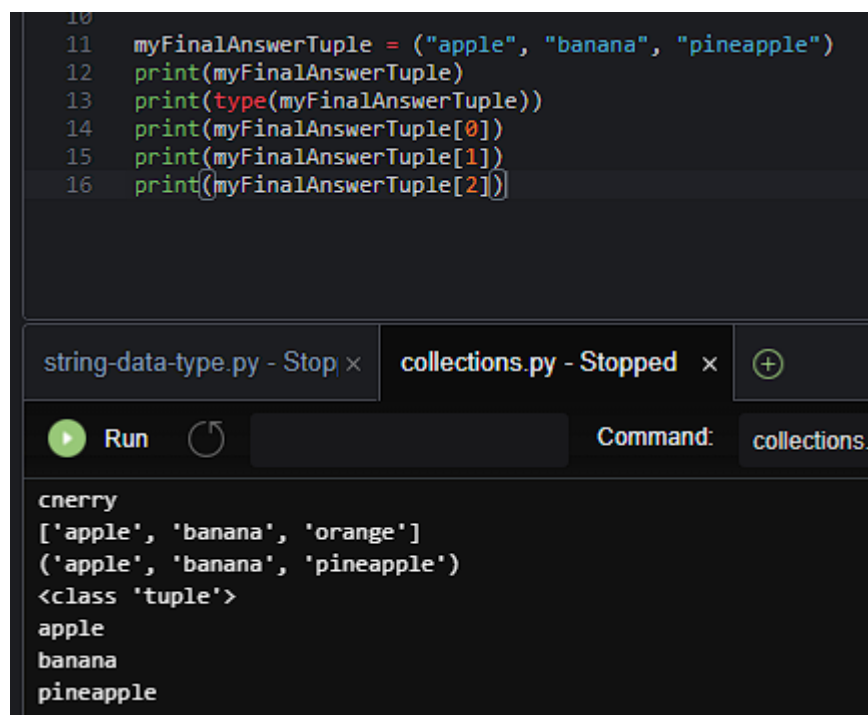
1. Para acceder a la cadena `pineapple`, escriba el siguiente código:

```
print(myFinalAnswerTuple[2])
```

1. Guarde y ejecute el archivo.

2. Cerca de la parte superior de la ventana del IDE, elija el botón **Run** (Play) (Ejecutar [Reproducir]).
3. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']  
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple
```



The screenshot shows a Python IDE with a dark theme. The editor window displays the following code:

```
10  
11 myFinalAnswerTuple = ("apple", "banana", "pineapple")  
12 print(myFinalAnswerTuple)  
13 print(type(myFinalAnswerTuple))  
14 print(myFinalAnswerTuple[0])  
15 print(myFinalAnswerTuple[1])  
16 print(myFinalAnswerTuple[2])
```

Below the editor, there is a toolbar with a green play button labeled "Run" and a circular arrow icon. To the right of the toolbar, there are two tabs: "string-data-type.py - Stop" and "collections.py - Stopped". Below the tabs, there is a "Command:" field with the text "collections.". At the bottom of the IDE, the output console shows the following text:

```
cherry  
['apple', 'banana', 'orange']  
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple
```

## Ejercicio 3: Presentar el tipo de dato de diccionario

### Definición de un diccionario

Un diccionario es una lista cuyas posiciones tienen nombres asignados (claves). Imagine que su lista muestra la fruta favorita de distintas personas.

1. Regrese al script en Python y escriba el siguiente código:



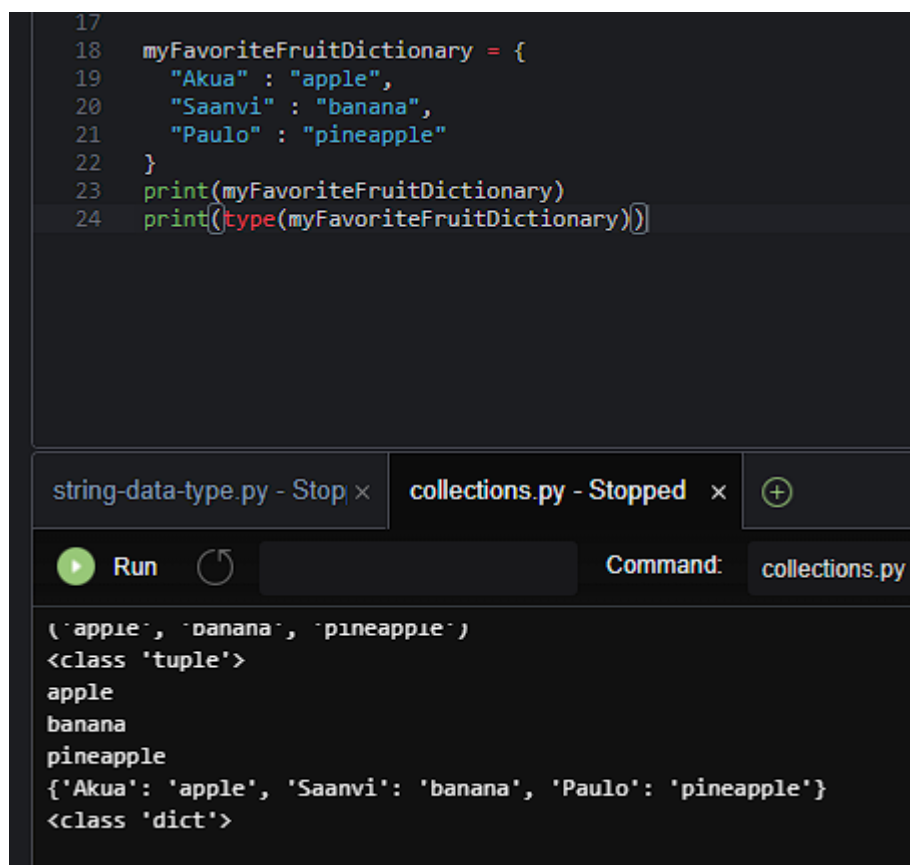
```
myFavoriteFruitDictionary = {  
    "Akua" : "apple",  
    "Saanvi" : "banana",  
    "Paulo" : "pineapple"  
}
```

1. Utilice la función `print()` para escribir el diccionario en el shell:

```
print(myFavoriteFruitDictionary)
```

1. Utilice la función `type()` para escribir el tipo de dato en el shell:

```
print(type(myFavoriteFruitDictionary))
```



The screenshot shows a Python IDE with a dark theme. The editor window displays the following code:

```
17  
18 myFavoriteFruitDictionary = {  
19     "Akua" : "apple",  
20     "Saanvi" : "banana",  
21     "Paulo" : "pineapple"  
22 }  
23 print(myFavoriteFruitDictionary)  
24 print(type(myFavoriteFruitDictionary))
```

Below the editor, there are tabs for "string-data-type.py - Stop" and "collections.py - Stopped". A "Run" button is visible. The command line shows "Command: collections.py". The output window displays the following results:

```
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple  
{'Akua': 'apple', 'Saanvi': 'banana', 'Paulo': 'pineapple'}  
<class 'dict'>
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

## Acceso al diccionario por nombre

En esta actividad, en lugar de utilizar números, recurrirá al nombre de las personas para acceder a su fruta favorita.

1. Para acceder a la fruta favorita de Akua, escriba el siguiente código:

```
print(myFavoriteFruitDictionary["Akua"])
```

1. Para acceder a la fruta favorita de Saanvi, escriba el siguiente código:

```
print(myFavoriteFruitDictionary["Saanvi"])
```

1. Para acceder a la fruta favorita de Paulo, escriba el siguiente código:

```
print(myFavoriteFruitDictionary["Paulo"])
```

1. Guarde y ejecute el archivo.
2. Confirme que el script se ejecuta de forma correcta y que la salida se muestra según lo previsto.

```
['apple', 'banana', 'cherry']  
<class 'list'>  
apple  
banana  
cherry  
['apple', 'banana', 'orange']  
('apple', 'banana', 'pineapple')  
<class 'tuple'>  
apple  
banana  
pineapple  
{'Akua': 'apple', 'Saanvi': 'banana', 'Paulo': 'pineapple'}  
<class 'dict'>  
apple  
banana  
pineapple
```

```
18 myFavoriteFruitDictionary = {
19     "Akua" : "apple",
20     "Saanvi" : "banana",
21     "Paulo" : "pineapple"
22 }
23 print(myFavoriteFruitDictionary)
24 print(type(myFavoriteFruitDictionary))
25
26 print(myFavoriteFruitDictionary["Akua"])
27 print(myFavoriteFruitDictionary["Saanvi"])
28 print(myFavoriteFruitDictionary["Paulo"])
```

string-data-type.py - Stop x collections.py - Stopped x +

Run Command: collections.py

```
banana
pineapple
{'Akua': 'apple', 'Saanvi': 'banana', 'Paulo': 'pineapple'}
<class 'dict'>
apple
banana
pineapple
```

¡Felicitaciones! Ha trabajado con los tipos de datos de lista, tupla y diccionario en Python.

## Finalizar laboratorio

¡Felicitaciones! Ha llegado al final del laboratorio.

1. Elige **End Lab** (Finalizar laboratorio) en la parte superior de esta página y, a continuación, selecciona Yes (Sí) para confirmar que deseas finalizar el laboratorio.

Un panel muestra el mensaje *DELETE has been initiated... You may close this message box now* (Se ha iniciado la ELIMINACIÓN... Ya puedes cerrar este mensaje).

1. Aparece brevemente el mensaje *Ended AWS Lab Successfully* (El laboratorio de AWS finalizó correctamente), que indica que el laboratorio ha finalizado.