Práctica 01 - Introducción

Pedro Fernando Flores Palmeros
Programación avanzada

Febrero 2020

Nota: Ingrese las líneas de código que se van mostrando a lo largo del texto para que vaya observando los cambios que se van generando. Al final debe de crear un archivo `.py` que incluya todos los comandos.

1 Lista

Una *lista* es una collección de elementos con un orden particular. PUedes hacer listas que incluyan letras el slfabeto, dígitos, nombres de familiares, etc. La lista puede contener cualquier dato y no necesariamente debe de exister una relación entre los lementos de la lista.

En `python`, se utilizan los `[]` para indicar una lista, los elementos individuales de la lista están seprados por coma. A continuación se muestra un ejemplo de una lista que contiene algunos tipos de bicicletas

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```

Ejercicio: Introduzca las líneas de código anteriores y escriba cuál fue el resultado de la ejecución tal cual aparece en la terminal

1.1 Ingresando a los lementos de la lista

Las listas son colecciones ordenadas, entonces se puede acceder a cada elemento de la lista indicando la posición o el `index` del elemento deseado. Para acceer a algún elemento de la lista, se debe de escribir el nombre de la lista seguido del índice entre corchetes.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[1])
```

Ejercicio: Introduzca las líneas de código anteriores y escriba cuál fue el resultado de la ejecución tal cual aparece en la terminal

La lista de bicicletas está conformada de puras cadenas de caracteres (A partir de ahora y a lo largo del curso se utlizará la palabra *string* de manera indistinta), entonces se pueden utlizar los métodos de string por ejemplo

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[0].title())
```

Es necesario que observe que el índice de las listas en `python` comienza en 0 y no en 1

1.2 Tarea

Realizar los siguientes programas para repasar los conceptos anteriormente expuestos

- Nombres: Realiza una lista con algunos nombres de tus amigos en una lista llamada `names`. Imprime en la pantalla el nombre de cada persona ingresando elemento por elemento.
- Mensaje: En la lista creada anteriormente, además de imprimir el nombre de cada persona imprime un mensaje personalizado para cada persona.
- Tu propia lista: Piensa en una lista deseos, la lista debe de tener por lo menos 15 elementos. Imprime algunos de los deseos. Ejemplo *Me gustaría tener una moto Honda".*

2 Modificando, Agregando y Removiendo elementos de la lista

2.1 Modificando los elementos de una lista

Para modificar los lementos de una lista simplemente basta con ingresar al índice del elemento y asignarle otro valor.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles[0] = 'ducati'
print(motorcycles)
```

2.2 Agregando elementos a la lista

La forma más simple de agregar elementos a la lista es través del método —append—. Cuando se utiliza éste método el elemento agregado será el último de la lista.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.append('ducati')
print(motorcycles)
```

2.3 Insertando elementos a una lista

Si no se quisiera agregar un elemento en una ubicación especificada de la lista, el método `append` no sería útil, para esta operación se necesita el método `insert`, éste método va acompañado de la posición en la que se desea ingresar y el elemento que se debe de ingresar.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.insert(0, 'ducati')
print(motorcycles)
```

2.4 Removing Elements from a List

Existen dos métodos para quitar elementos de una lista, el primero es `del` y el segundo es `pop`

2.4.1 del

Si se sabe la posición elemento que se quiere eliminar se pude utilizar el siguiente código

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[0]
print(motorcycles)
```

3 Funciones

Una función es una subrutina dentro del programa, que se decide aislar para que después pueda ser llamada en cualquier parte del programa, en el siguiente ejemplo se ve una forma para poder declarar una función

```
def greet_user():
    """Dipaly a simple greeting"""
    print("Hello!")


greet_user()
```

3.1 Passing Information to a Function

Una función puede recibir información para que pueda ser utilizada dentro de sí misma

```
def greet_user(username):
    """Display a simple greeting."""
    print("Hello, " + username.title() + "!")
```

```
greet_user('jessie')
```

3.2 Paso de argumentos

Una función puede tener múltiples argumentos, hay diferentes maneras de enviar argumentos a una función en general destacan dos tipos *Positional Arguments* y *keyword arguments*

3.2.1 Positional Arguments

En este tipo de argumentos es importante la forma (orden) en que se envían los argumentos. Considere el siguiente ejemplo

```
def describe_pet(animal_type, pet_name):
   """Display information about a pet."""
   print("\nI have a " + animal_type + ".")
   print("My " + animal_type + "'s name is " + pet_name.title() + ".")
describe_pet('hamster', 'harry')
describe_pet('dog', 'willie')
```

en comparación con el siguiente código,

```
def describe_pet(animal_type, pet_name):
   """Display information about a pet."""
   print("\nI have a " + animal_type + ".")
   print("My " + animal_type + "'s name is " + pet_name.title() + ".")
describe_pet('harry', 'hamster')
```

3.2.2 Keword Arguments

Para evitar los problemas del programa anterior

```
def describe_pet(animal_type, pet_name):
   """Display information about a pet."""
   print("\nI have a " + animal_type + ".")
   print("My " + animal_type + "'s name is " + pet_name.title() + ".")
describe_pet(animal_type='hamster', pet_name='harry')
```

3.3 Default Values

En ocasiones se puede definir uno o varios parámetros de manera predeterminada

```
def describe_pet(pet_name, animal_type='dog'):
   """Display information about a pet."""
   print("\nI have a " + animal_type + ".")
   print("My " + animal_type + "'s name is " + pet_name.title() + ".")
```

```
describe_pet(pet_name='willie')
```

3.4 Tarea

- **T-Shirt**: Elabora un función llamada `make_shirt()` que acepte el tamaño y el texto que se imprimirá en la playera. La función deberá de imprimir el tamaño y el texto que se han enviado a la función.
- **Playeras Grandes**: Modifique la función `make_shirt()` de tal manera que el tamaño por default sea `grande` y el texto predefinido sea `I <3 Python`. Genere una playera grande y mediana con el mensaje predeterminado, y genere una playera pequeña con un mensaje diferente.
- **Ciudades**: Escriba una fución llamada `describe_city()` que acepte como argumentos la ciudad y el país. La función debe imprimir un enunciado sencillo como: *París está en Francia*. Define el parámetro para el país con un valor predeterminado. Llame la función tres veces y que por lo menos una no sea el país predeterminado.

4 Clases

De forma muy general la clase está conformada de atributos y métodos, los atributos son variables o estructuras de datos, mientras que los métodos generalmente están definidos a través de funciones.

Las clases deben de tener una función principal que se llama constructor, en dicho constructor se inicializan y se definen todos los atributos, los métodos son funciones comunes y corrientes con la única particularidad de que deben de tener un argumento `self` que indica la pertenencia a la clase, de hecho es un puntero que apunta hacia la misma clase (en c++ sería equivalente al this-¿)

```
class Dog():
   """Clase que ayuda a describir un perro"""
   def __init__(self, name, age):
      """Inicializa los atributos name y age"""
      self.name = name
      self.age = age

   def sit(self):
      """Simula el comportamiento de un perro al sentarse"""
      print(self.name.title() + " is now sitting")

   def roll_over(self):
      """Simula el comportamiento de un perro al girar"""
      print(self.name.title() + " ha dado una vuelta")
```

4.1 Definiendo un objeto de una clase

Al hecho de generar un objeto de una clase se le conoce como instanciar, o instancia, para poder generar el objeto, primero se debe definir la clas una vez que la clase se ha definido se puede instanciar el objeto.

```
my_dog = Dog('willie',6)
print("My dog's name is " + my_dog.name.title() + ".")
print("My dog is " + str(my_dog.age) + " years old.")
```

Para acceder a los atributos se debe de utilizar el operador (.) acompañado del nombre del atributo y del nombre del objeto.

```
my_dog.name
```

Lo mismo sucede con los métodos

```
my_dog.sit()
my_dog.roll_over()
```

4.2 Tarea

- **Restaurant:** Declare una clase llamada `Restaurant`. El constructor debe de tener dos atributos, `restaurant_name` y `cuisine_type`. Desarrolle dos métodos `describe_restaurant()` que imprima en pantalla la información y el segundo `open_restaurant()` que imprima si el restaurante es abierto. Genere un objeto tipo restauratne y verifique su funcionamiento.

- **Tres restaurantes:** Retome la clase del ejercicio anterior, genere tres restaurantes con nombres diferentes e imprima los atributos de los tres restaurantes.

- **Usuarios:** Genere una clase llamada `User` con dos atributos `first_name` y `last_name`, agregue más atributos que se utilizan para cuentas de usuario. Desarrolle un método `describe_user()` que imprima de manera ordenada y detallada la infomación del usuario. Genere un segundo atributo `greet_user()` que imprima un saludo personalido para el usuario. Cree 10 usuarios y ejecute los dos métodos para cada usuario.