

# 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 colección de elementos con un orden particular. Puedes hacer listas que incluyan letras el alfabeto, dígitos, nombres de familiares, etc. La lista puede contener cualquier dato y no necesariamente debe de existir una relación entre los elementos de la lista.

En Python, se utilizan los [] para indicar una lista, los elementos individuales de la lista están separados 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 cual fue el resultado de la ejecución tal cual aparece en la terminal: crea la lista y nos imprime la lista que se acaba de crear

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

```
edwin@edwin-virtual-machine:~/Desktop$ python Comandos.py  
['trek', 'cannondale', 'redline', 'specialized']
```

## 1.1 Ingresando a los elementos de la lista

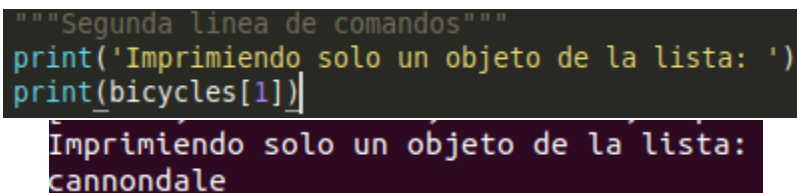
Las listas son colecciones ordenadas, entonces se puede acceder a cada elemento de la lista indicando la posición o el índice del elemento deseado. Para acceder a algún elemento de la lista, se debe de escribir el nombre de la lista seguido del índice entre corchetes.

---

```
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



```
"""Segunda línea de comandos"""  
print('Imprimiendo solo un objeto de la lista: ')  
print(bicycles[1])  
Imprimiendo solo un objeto de la lista:  
cannondale
```

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

---

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

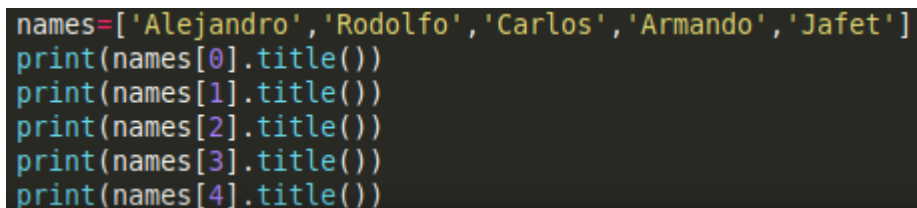
---

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.



```
names=['Alejandro', 'Rodolfo', 'Carlos', 'Armando', 'Jafet']  
print(names[0].title())  
print(names[1].title())  
print(names[2].title())  
print(names[3].title())  
print(names[4].title())
```

```
Alejandro
Rodolfo
Carlos
Armando
Jafet
```

- Mensaje: En la lista creada anteriormente, además de imprimir el nombre de cada persona imprime un mensaje personalizado para cada persona.

```
"""Mensaje por cada persona"""
for name in names:
    print('hola amigo, '+str(name)+'!!!')
```

```
hola amigo, Alejandro!!!
hola amigo, Rodolfo!!!
hola amigo, Carlos!!!
hola amigo, Armando!!!
hola amigo, Jafet!!!
```

- 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*”.

```
wishes=['Terminar mi carrera','Conseguir novia','alcanzar todas mis metas',
        'Mejorar mis calificaciones', 'Conseguir un buen trabajo','Mejorar mi nivel de programacion',
        'Ser el orgullo de mis padres','aprender a andar mejor en moto', 'mejorar mi nivel de ingles',
        'enamorarme de nuevo','ser feliz','viajar por el mundo','comprarme una moto mas grande',
        'ganar un campeonato de Videojuegos','Regresarle todo lo que me han dado mis padres a mis hijos']
```

```
Conseguir Un Buen Trabajo
Aprender A Andar Mejor En Moto
Mejorar Mi Nivel De Programacion
```

## 2 Modificando, Agregando y Removiendo elementos de la lista

### 2.1 Modificando los elementos de una lista

Para modificar los elementos 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)
```

---

```
print('Parte 1 del ejercicio 2')
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles[0] = 'ducati'
print(motorcycles)
```

```
Parte 1 del ejercicio 2
['honda', 'yamaha', 'suzuki']
['ducati', 'yamaha', 'suzuki']
```

## 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 este método el elemento agregado será el último de la lista.

---

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

---

```
'''2.2 Agregando elementos a la lista'''
print('Parte 2 del ejercicio 2')
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.append('ducati')
print(motorcycles)
```

```
Parte 2 del ejercicio 2
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha', 'suzuki', 'ducati']
```

## 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)
```

---

```
'''2.3 Insertando elementos a una lista'''
print('Parte 3 del ejercicio 2')
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.insert(0, 'ducati')
print(motorcycles)
```

```
Parte 3 del ejercicio 2
['honda', 'yamaha', 'suzuki']
['ducati', 'honda', 'yamaha', 'suzuki']
```

## 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 puede utilizar el siguiente código

---

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

---

```
'''2.4 Removing Elements from a List'''
print('Parte 4 del ejercicio 2')
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[0]
print(motorcycles)
```

```
Parte 4 del ejercicio 2
['honda', 'yamaha', 'suzuki']
['yamaha', 'suzuki']
```

### 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 Funciones'''
'''3.1 Passing Information to a Function'''
print('Parte 1 del ejercicio 3')
def greet_user():
    """Dipaly a simple greeting"""
    print("Hello!")

greet_user()
```

```
Parte 1 del ejercicio 3
Hello!
```

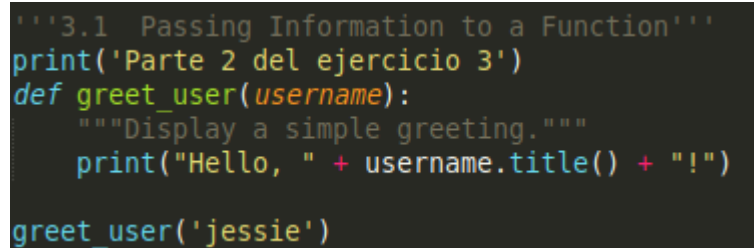
## 3.1 Passing Information to a Function

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

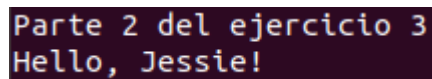
---

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

---



```
'''3.1 Passing Information to a Function'''
print('Parte 2 del ejercicio 3')
def greet_user(username):
    """Display a simple greeting."""
    print("Hello, " + username.title() + "!")
greet_user('jessie')
```



```
Parte 2 del ejercicio 3
Hello, 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')
```

---

```
'''3.2 Paso de argumentos'''
print('Parte 3 del ejercicio 3')
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')
```

```
Parte 3 del ejercicio 3

I have a hamster.
My hamsters name is Harry.

I have a dog.
My dogs name is 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')
```

---

```
describe_pet('dog', 'willie')
'''En comparacion con el codigo'''
print('Parte 4 del ejercicio 3')
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')
```

```
Parte 4 del ejercicio 3

I have a harry.
My harrys name is Hamster.
```



### 3.2.2 Keyword 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')
```

---

```
'''3.2.2 Keyword Arguments'''
print('Parte 5 del ejercicio 3')
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')
```

```
Parte 5 del ejercicio 3

I have a hamster.
My hamsters name is 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')
```

---

```
describe_pet(animal_type='hamster', pet_name='harry')
'''3.3 Default Values'''
print('Parte 6 del ejercicio 3')
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')
```

```
Parte 6 del ejercicio 3
```

```
I have a dog.  
My dogs name is 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.

```
'''3.4 Tarea'''  
print('\nTarea 3.4.1')  
def make_shirt(size_shirt,message_shirt):  
    print("\nhis shirt is: " + size_shirt + ".")  
    print('This is the text that will appear on your shirt: '+''+message_shirt+'')  
make_shirt(size_shirt='big',message_shirt='IPN')
```

```
Tarea 3.4.1
```

```
his shirt is: big.  
This is the text that will appear on your shirt: "IPN"
```

- **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.

```
'''3.4 Tarea'''  
print('\nTarea 3.4.2')  
def make_shirt(size_shirt='big',message_shirt='I <3 Python'):  
    print("\nhis shirt is: " + size_shirt + ".")  
    print('This is the text that will appear on your shirt: '+''+message_shirt+'')  
make_shirt()  
make_shirt(size_shirt='medium')  
make_shirt(size_shirt='small',message_shirt='it Works !!!')
```

#### Tarea 3.4.2

```
this shirt is: big.  
This is the text that will appear on your shirt: "I <3 Python"  
  
this shirt is: medium.  
This is the text that will appear on your shirt: "I <3 Python"  
  
this shirt is: small.  
This is the text that will appear on your shirt: "it Works !!!"
```

- **Ciudades:** Escriba una función llamada `describe_city()` que acepte como argumentos la ciudad y el país. La función debe imprimir un enunciado sencillo como: *París está en Francia*. Defina 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.

```
'''3.4 Tarea'''  
print('\nTarea 3.4.3')  
def describe_city(city, country='Rusia'):  
    print('\nthe '+city+' is in '+country)  
describe_city(city='moscow')  
describe_city(city='paris', country='France')  
describe_city(city='toronto', country='Canada')
```

#### Tarea 3.4.3

```
the moscow is in Rusia  
  
the paris is in France  
  
the toronto is in Canada
```

## 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-¿)

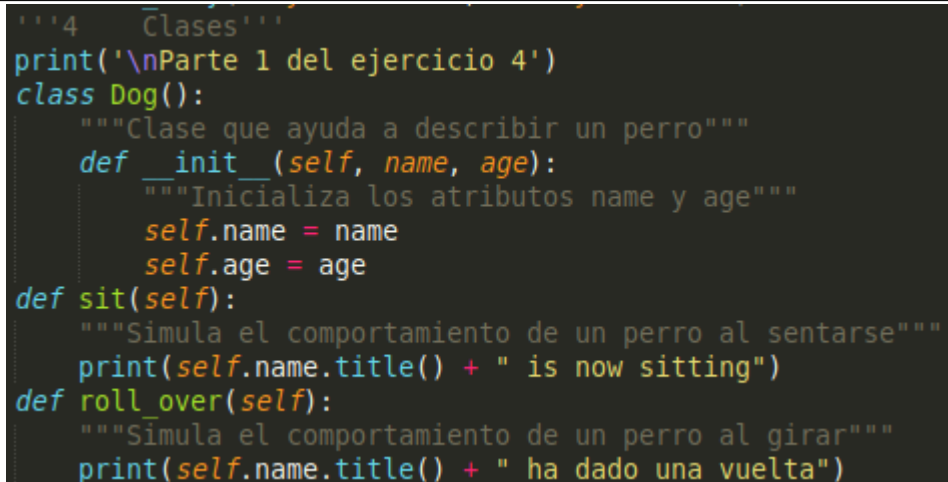
---

```
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 Clases'''
print('\nParte 1 del ejercicio 4')
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 clase 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.")
```

---

```
my_dog = Dog('willie',6)  
print("My dogs 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 `restaurant` y verifique su funcionamiento.

```
class Restaurant():
    '''Clase del Restaurante'''
    def __init__(self, restaurant_name, restaurant_type, restaurant_OC):
        self.restaurant_name=restaurant_name
        self.restaurant_type=restaurant_type
        self.restaurant_OC=restaurant_OC
    def describe_restaurant(self):
        print(self.restaurant_name+'It is the best restaurant in the world.')
        print('This restaurant specializes in:'+self.restaurant_type)
    def open_restaurant(self):
        print(self.restaurant_OC)
```

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

```
my_restaurant=Restaurant('Don Martin','Mexican food','open')
print("\nMy restaurant name is " + my_restaurant.restaurant_name.title() + ".")
print("\nThis restaurant specializes in: " + my_restaurant.restaurant_type)
print("\nThis restaurant is now: " + my_restaurant.restaurant_OC)
my_restaurant=Restaurant('moshi moshi ','japanesse food','close')
print("\nMy restaurant name is " + my_restaurant.restaurant_name.title() + ".")
print("\nThis restaurant specializes in: " + my_restaurant.restaurant_type)
print("\nThis restaurant is now: " + my_restaurant.restaurant_OC)
my_restaurant=Restaurant('Italianis','italian food','open')
print("\nMy restaurant name is " + my_restaurant.restaurant_name.title() + ".")
print("\nThis restaurant specializes in: " + my_restaurant.restaurant_type)
print("\nThis restaurant is now: " + my_restaurant.restaurant_OC)
```

```
My restaurant name is Don Martin.
This restaurant specializes in: Mexican food
This restaurant is now: open
My restaurant name is Moshi Moshi .
This restaurant specializes in: japanesse food
This restaurant is now: close
My restaurant name is Italianis.
This restaurant specializes in: italian food
This restaurant is now: open
```

- **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 información del usuario. Genere un segundo atributo greet\_user() que imprima un saludo personalizado para el usuario. Cree 10 usuarios y ejecute los dos métodos para cada usuario.

```
class user():
    def __init__(self, first_name, last_name, type_blood, allergies, age):
        self.first_name=first_name
        self.last_name=last_name
        self.type_blood=type_blood
        self.allergies=allergies
        self.age=age
    def describe_user(self):
        print('\nName of user: '+self.first_name)
        print('\nlast name of user: '+self.last_name)
        print('\nType of blood: '+self.type_blood)
        print('\ntype of allergy: '+self.allergies)
        print('\nAge: '+str(self.age))
    def greet_user(self):
        print('Hola, '+self.first_name+', welcome to the program.')
```

```
my_user=user('Eduardo', 'Mendez', '0+', 'penisilina', 18)
print(my_user.greet_user())
print(my_user.describe_user())
```

```
Hola, Eduardo, welcome to the program.
None

Name of user: Eduardo

last name of user: Mendez

Type of blood: 0+

type of allergy: penisilina

Age: 18
```