

Modulo 1

En este primer módulo se tratarán conceptos básicos sobre python, tales como manejo de datos y sus respectivos casteos.

1.1 - Imprimiendo a pantalla

Nota: Cuando se diga la palabra *Imprimir*, se está haciendo referencia al hecho de tener una salida a pantalla, no a la acción de imprimir físicamente por medio de impresoras o fax

Para la impresión por pantalla se utiliza la palabra `print()` y entre los paréntesis se pone la cadena que se desea imprimir.

Esta cadena de texto o palabra, se debe poner ente `"""` comillas dobles `print("Palabra a imprimir")`, o bien con comillas simples `print('Palabra a imprimir')`.

¿Cuál es la diferencia? Al poner entre comillas dobles, se puede abarcar mas caracteres que con comilla simple, siendo que con comilla doble podemos imprimir el carácter de la comilla simple

```
print("'Palabra'")
```

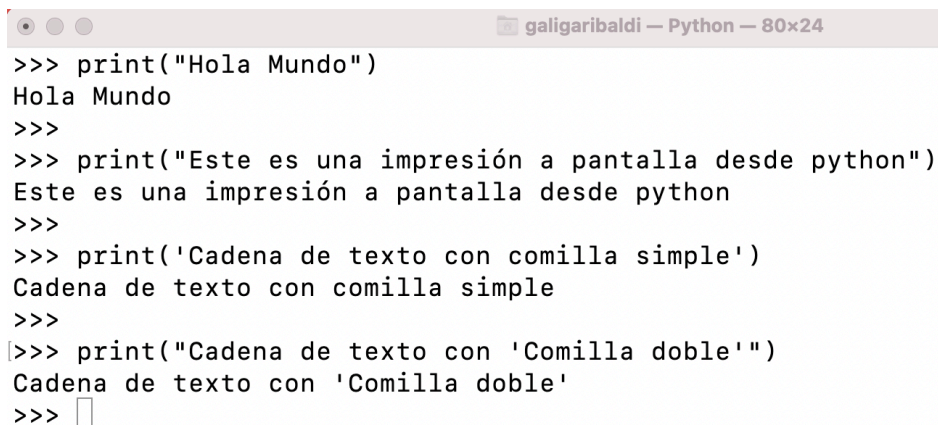
Ejemplo:

```
print("Hola Mundo")

print("Este es una impresión a pantalla desde python")

print('Cadena de texto con comilla simple')

print("Cadena de texto con 'Comilla doble'")
```



```
galigaribaldi — Python — 80x24

>>> print("Hola Mundo")
Hola Mundo
>>>
>>> print("Este es una impresión a pantalla desde python")
Este es una impresión a pantalla desde python
>>>
>>> print('Cadena de texto con comilla simple')
Cadena de texto con comilla simple
>>>
>>> print("Cadena de texto con 'Comilla doble'")
Cadena de texto con 'Comilla doble'
>>> 
```

1.2 - Tipos de dato (Int, String, float, Bool)

Concepto de Variable: Una variable es un área de memoria donde nosotros podemos almacenar valores, estos valores nosotros los definimos y decidimos cuando y como usarlos, por lo regular en casi todos los lenguajes de programación se utiliza la siguiente sintaxis



```
NombreVariable = ValorVariable
```

Donde: **NombreVariable* = es el nombre que llevará nuestra variable, *ValorVariable* = Valor de la variable

Ejemplo:

```
variable1 = "Esta es una variable de texto"  
variable2 = 12  
variable3 = 12.4
```

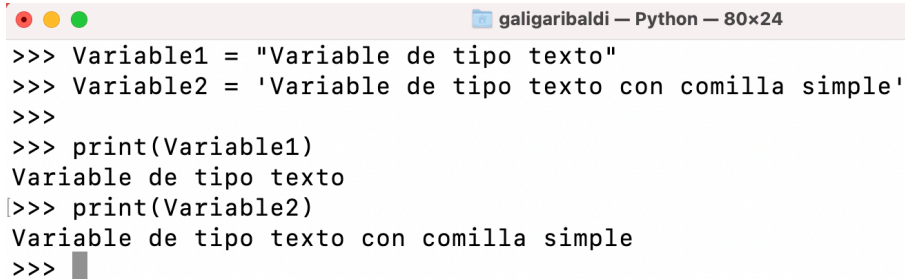
En python existen diferentes tipos de datos, los cuales tienen diferentes utilidades, a continuación se describe una lista de los más comunes:

- **String - Char (Cadenas de texto):** Tipo de dato que python nos da por default, es el que nos da por default al recibir datos desde el teclado.
 - `variable = 'Cadena'`
 - `variable = "Cadena"`

Ejemplo:

```
Variable1 = "Variable de tipo texto"
Variable2 = 'Variable de tipo texto con comilla simple'

print(Variable1)
print(Variable2)
```



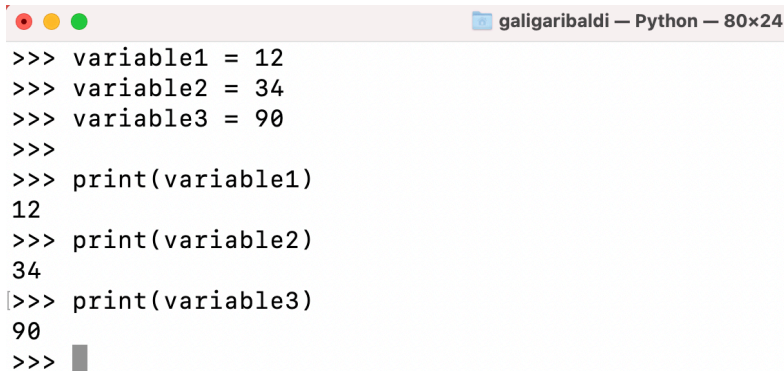
```
galigaribaldi — Python — 80x24
>>> Variable1 = "Variable de tipo texto"
>>> Variable2 = 'Variable de tipo texto con comilla simple'
>>>
>>> print(Variable1)
Variable de tipo texto
>>> print(Variable2)
Variable de tipo texto con comilla simple
>>> █
```

- **Int (Tipo de dato numérico):** Tipo de dato numérico entero. Para este tipo de dato se tienen los números naturales positivos y negativos

Ejemplo:

```
variable1 = 12
variable2 = 34
variable3 = 90

print(variable1)
print(variable2)
print(variable3)
```



```
galigaribaldi — Python — 80x24
>>> variable1 = 12
>>> variable2 = 34
>>> variable3 = 90
>>>
>>> print(variable1)
12
>>> print(variable2)
34
>>> print(variable3)
90
>>> █
```

- **Float** (Tipo de dato con punto Flotante o Punto Decimal): En éste caso tenemos un tipo de dato numérico pero con punto decimal o bien punto flotante.

Ejemplo:

```
variable1 = 12.2
variable2 = 21.1
variable3 = 34.9

print(variable1)
print(variable2)
print(variable3)
```



The screenshot shows a terminal window titled "galigaribaldi — Python — 80x24". The prompt is ">>>". The code entered is: variable1 = 12.2, variable2 = 21.1, variable3 = 34.9, followed by print statements for each variable. The output shows the values 12.2, 21.1, and 34.9 printed on separate lines.

```
>>> variable1 = 12.2
>>> variable2 = 21.1
>>> variable3 = 34.9
>>>
>>> print(variable1)
12.2
>>> print(variable2)
21.1
>>> print(variable3)
34.9
>>> █
```

- **Bool** (Tipo de dato True or False): Tipo de dato que denota si es verdadero o falso

Ejemplo:

```
#####Tipos de dato Booleano
Variableverdadera = True
VariableFalsa = False
```

- **List** (Listas): Tipo de dato compuesto, en este caso tenemos un arreglo ordenado de valores, los cuales comienzan por el valor 0 y terminan en el valor n , podemos meter diferentes tipos de valores en él.
- **Tuple** (tuplas): Tipo de dato compuesto similar a las listas, con la gran diferencia que en este caso no podemos agregar o quitar valores, es decir, es un tipo de dato estático.
- **Dictionary** (Diccionario): Tipo de dato el cual contiene una serie de elementos, ordenados por *llave: Valor*, en el cual se puede acceder al dato por medio de la llave

1.3 - Operaciones entre tipos de dato

Entre los diferentes tipos de dato, se puede hacer operaciones aritméticas y lógicas así como operaciones propias de python, a continuación se describe una lista

- Operaciones entre Int - Int
 - **Suma:** Suma entre diferentes variables o números con el operador +

- **Resta:** Resta entre diferentes variables o números con el operador -
- **Multiplicación:** Multiplicación entre diferentes variables o números con el operador *
- **División:** Multiplicación entre diferentes variables o números con el operador /
- Operaciones entre Float - Float: Similares a las operaciones de Int

Ejemplo:

```
###Tipo Int
varibaleInt = 34
variableInt2 = 45.2
print(varibaleInt, variableInt2)

##Para saber el tipo de dato que es, usamos type()
print("La variable variableInt es: ", type(varibaleInt))
print("La variable variableInt 2 es: ", type(variableInt2))

###Operaciones entre int

###Operaciones entre sumas
print("Suma: ", varibaleInt + variableInt2)
print("Suma con diferentes números: ", varibaleInt + 89)
###Operaciones entre resta
print("Resta: ", varibaleInt - variableInt2)
print("Resta con diferentes números: ", varibaleInt - 100)
###Operaciones entre Multiplicacion
print("Multiplicacion: ", varibaleInt * variableInt2)
print("Multiplicacion entre diferentes numeros: ", varibaleInt * 10)
###Operaciones entre division
print("Division: ", varibaleInt / variableInt2)
print("Division entre diferentes numeros: ", varibaleInt / 100)
#Si se divide entre 0, python marcarà un error
#print("Division entre diferentes numeros: ", varibaleInt / 0)
```

```
34 45.2
La variable variableInt es: <class 'int'>
La variable variableInt 2 es: <class 'float'>
Suma: 79.2
Suma con diferentes números: 123
Resta: -11.200000000000003
Resta con diferentes números: -66
Multiplicacion: 1536.8000000000002
Multiplicacion entre diferentes numeros: 340
Division: 0.7522123893805309
Division entre diferentes numeros: 0.34
La variable 1 es: de tipo: <class 'bool'> y contiene el valor: True
La variable 2 es: de tipo: <class 'bool'> y contiene el valor: False
```

- Operaciones entre String - String:

- **Concatenación:** Unión de 2 cadenas
- **Slices:** Particionamiento de una palabra en subpartes

```
cadena1 = "Cadena 1"
cadena2 = "Cadena 2"
Cadena3 = cadena1 + cadena2
print(Cadena3)
Cadena3 = cadena1 + " "+cadena2
print(Cadena3)
print(cadena1[0:2])
print(cadena1[0:6])
```

```
>>> cadena1 = "Cadena 1"
>>> cadena2 = "Cadena 2"
>>> cadena1 + cadena2
'Cadena 1Cadena 2'
>>> cadena1 + " "+cadena2
'Cadena 1 Cadena 2'
>>> cadena1[0:2]
'Ca'
>>> cadena1[0:6]
'Cadena'
>>> █
```

1.4 - Casteo entre tipos de dato

Cuando necesitamos convertir un tipo de dato a otro (esto debido al manejo u operaciones del mismo), hacemos una operación de casteo, la cual consiste en transformar un tipo de dato a otro.

NOTA: Este casteo puede afectar la integridad del dato original, ocasionando perdidas

El ejemplo mas común de un casteo es cuando se usa para recibir datos del teclado. Para esto se usa la palabra reservada `input("Cadena de textos")`

Ejemplo:

```
###Casteo de datos entre numerico y cadena

respuesta = input("Digame su edad? >> ")
print(respuesta + " Años")

respuesta1 = input("Cuantos helado puede comer? >> ")
print(respuesta1 + " Helados")

respuesta2 = int(input("Digame su edad y la multiplicaré por 10>> "))
print(respuesta2 *10 )
```

```
galigaribaldi@MacBook-Pro-de-Hernan Ejemplos % python3 ejemplo-04.py
Digame su edad? >> 24
24 Años
Cuantos helado puede comer? >> 23
23 Helados
Digame su edad y la multiplicaré por 10>> 24
240
```

En lecciones futuras se verá como castear mas tipos de dato, por el momento solo se deja algunos ejemplos en código

La mayoría de las veces sólo es necesario poner el tipo de dato y envolver la variable que se desea convertir para poder convertir el tipo de dato al deseado

```
###Entero a Flotante
entero = 45
flotante = float(entero)
print(type(flote))
###flotante a Entero
flotante = 45.34
entero = int(flote)
print(type(entero))
```

1.5 - Manejando Listas y Tuplas

Cómo se explicó en materiales pasados, las listas y las tuplas comparten similitudes. Siendo conjuntos ordenados de valores, donde estos valores pueden ser básicamente cualquier valor que nosotros deseemos.

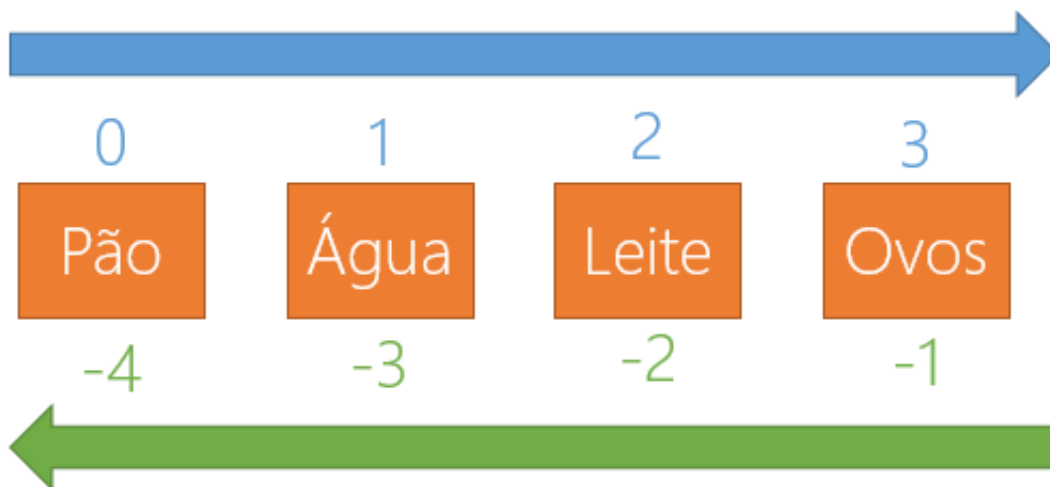
Definiendo una Lista - Tupla

Sin embargo existe una principal diferencia, las tuplas son tipos de dato **estáticos**, mientras que las listas son tipos de dato **Dinámico**, esto quiere decir que una vez definida la tupla, no podremos modificar el conjunto de datos. Mientras que en las listas si definimos una lista vacía, podremos agregarle n valores hasta que python nos lo permita.

Para el manejo de ambos tipos de dato se puede observar la siguiente imagen.



Como podemos observar, en una lista en python se tienen diferentes tipos de valores almacenados y se pueden acceder a ellos a través de su *índice* o *posición*, esta posición empieza desde el lugar 0 hasta el lugar n , donde n , es el límite del mismo.



Éste índice puede ser positivo (a la derecha), o negativo (hacia la izquierda).

****NOTA:** Lo mas común es usar los índices de manera

```
##Lista exclusivamente de números
listaNumeros = [10,20,30,40,50]

##Accediendo a un elemento en las listas
print("Accediendo al elemento 0: ", listaNumeros[0])
print("Accediendo al elemento 1: ", listaNumeros[1])
```



```

##Lista exclusivamente de Cadenas de texto
listaCadenas = ["Cadena 1", "Cadena 2", "Cadena 3"]

##Accediedo a un elemento en las listas
print("Accediendo al elemento 0: ", listaCadenas[0])
print("Accediendo al elemento 1: ", listaCadenas[1])

##Tupla exclusivamente de Cadenas de texto
tuplaCadenas = ("Cadena 1", "Cadena 2", "Cadena 3")

##Accediedo a un elemento en las listas
print("Accediendo al elemento 0: ", tuplaCadenas[0])
print("Accediendo al elemento 1: ", tuplaCadenas[1])

```

Operaciones con Listas

Las listas tienen un gran número de operaciones que podemos hacer entre ellas, entre las mas comunes se encuentran:

- Agregar al final
- Eliminar el último elemento
- Contar un número específico
- Generar una copia de la lista

```

###Operaciones con listas

###Agregar al final
lista1 = [1,2,3]
print("Lista Vieja: ", lista1)
lista1.append(4)
print("Lista Nueva: ", lista1)

###Quitar Valores al final
print("Lista Vieja: ", lista1)
lista1.pop()
print("Lista Nueva: ", lista1)

###Contar numero de elementos
print("Existen ", lista1.count(1), " Numeros 1")

###Generar una copia de la lista
lista2 = lista1.copy()

```

1.6 - Manejando Diccionarios

Un diccionario en python está definido por la siguiente nomenclatura: *llave: Valor*, en donde gracias a la llave que definamos podemos encontrar el valor especificado.

De manera similar siempre se recomienda que la llave del diccionario siempre sea un entero, mientras que el valor de el diccionario - llave, sea una cadena, lista o cualquier otro tipo de dato u objeto.

```
diccionario = {1:"Valor 1", 2: "Valor 2", "Llave 1": "Valor 1"}

###Acceder a un diccionario por su llave
print("El valor 1: ", diccionario[1])
print("El valor 2: ", diccionario[2])
print("Obteniendo todas las llaves: ", diccionario.keys())
print("Obteniendo todas los Valores: ", diccionario.values())

##Modificando un valor por su llave
diccionario[1] = 1
print("El valor 1: ",diccionario[1] )
```