Python Semestral Básico

Tipos de Datos

PROTECO

Octubre de 2019

Variables

- Una variable está formada por un espacio en la memoria principal (RAM) de una computadora y un nombre simbólico (identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad de información conocida o desconocida, es decir un valor.

Reglas para nombrar variables

- Se debe iniciar con guión bajo o letra seguida de cualquier cantidad de letras, dígitos o guiones bajos:
  - `>>_spam, spam, Spam_1` ✓
  - `>>1_Spam, spam$, (@#!)` ✗

*Python diferencia MAYÚSCULAS de minúsculas: `SPAM` es diferente de `spam`.

Reglas para nombrar variables

- No use **palabras reservadas** como variables de usuario, porque se sobreescribirán.
- Por respeto a las convenciones privadas de definición en Python, no use doble guión bajo al inicio y al final de sus variables. Ej:

```
>>> __init__
```

Tipos Numéricos

- Python puede tiene diferentes tipos para manejar valores numéricos, los cuales son:
  - » Enteros: `42`
  - » Flotantes: `3.14`
  - » Complejos `1+1j`
  - » Enteros Largos y otras bases

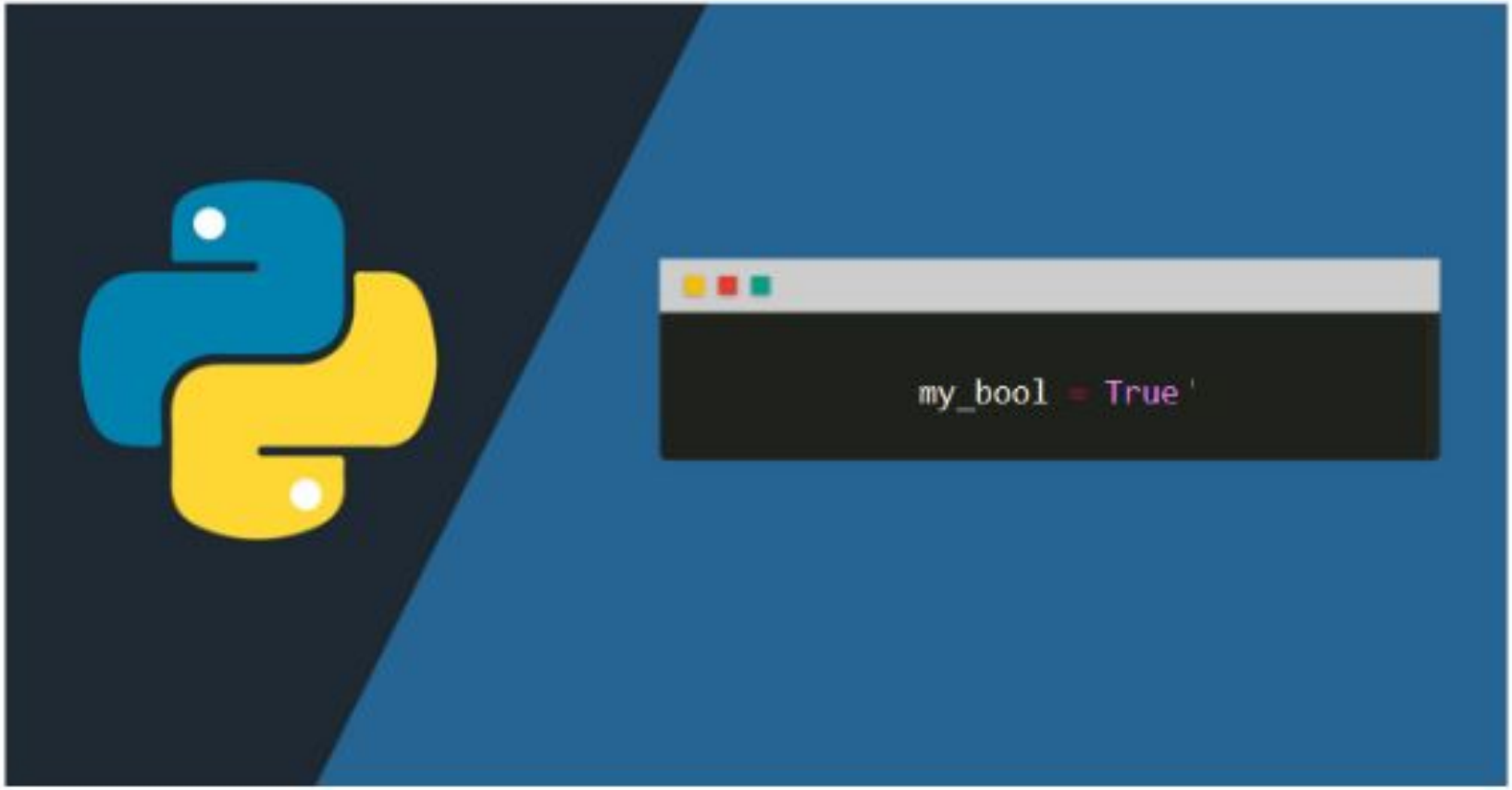
Tipos Numéricos

int	long	float	complex
10	51924361L	0.0	3.14j
100e-10	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080e3	0xDEFABCECBDAECBFBAEl	32.3e-18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J

Booleanos

- Los valores booleanos son los dos objetos constantes falso (False) y verdadero (True) que se utilizan para representar valores de verdad.
- La función incorporada bool () puede utilizarse para convertir cualquier variable a un valor booleano, si su valor puede interpretarse como un valor de verdad.

Booleanos



Cadenas

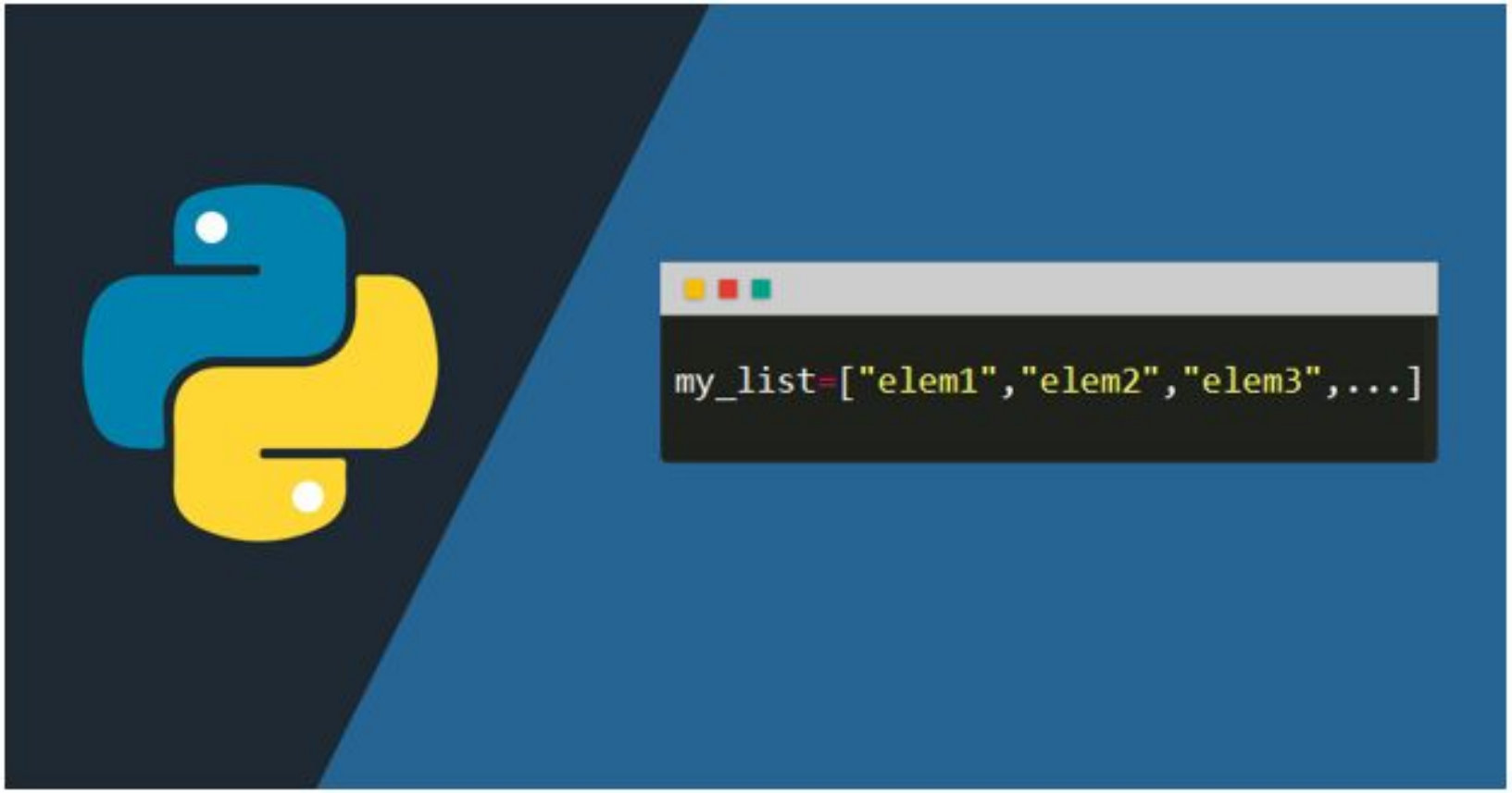
- Python puede manipular cadenas de texto, las cuales pueden ser expresadas encerrando texto entre comillas simples (‘ ’) o dobles(“”).
- Se puede indicar que una cadena contiene varias linea encerrándola entre tres pares de comillas dobles (“””” “”””) o separando cada linea con el caracter de barra invertida (\)

Cadenas

Listas

- Una lista es un tipo de colección ordenada similar a lo que en otros lenguajes se conoce como arrays o vectores.
- Las listas pueden contener cualquier tipo de dato: números, cadenas, booleano e incluso otras listas.

Listas



Listas

Método	Descripción
`lista.append(x)`	Agrega x al final de la lista.
`lista.extend(lista2)`	Agrega cada elemento de `lista2` a la lista.
`lista.insert(i, x)`	Agrega el elemento `x` en la posición `i` y recorre una posición todos los demás.
`lista.remove(x)`	Elimina el primer elemento de la lista con el valor de `x`.
`lista.pop(i)`	Regresa y elimina el elemento en la posición `i`.
`lista.clear()`	Eliminar todos los elementos de la lista.

Listas

Método	Descripción
`lista.index(x)`	Regresa el índice de la lista donde se encuentra el primer elemento con el valor de x.
`lista.count(x)`	Regresa el número de veces que aparece el valor de x en la lista.
`lista.sort()`	Ordena los valores de la lista.
`lista.reverse()`	Invierte el orden de todos los elementos de la lista.
`lista.copy()`	Regresa una copia de la lista.

Tuplas

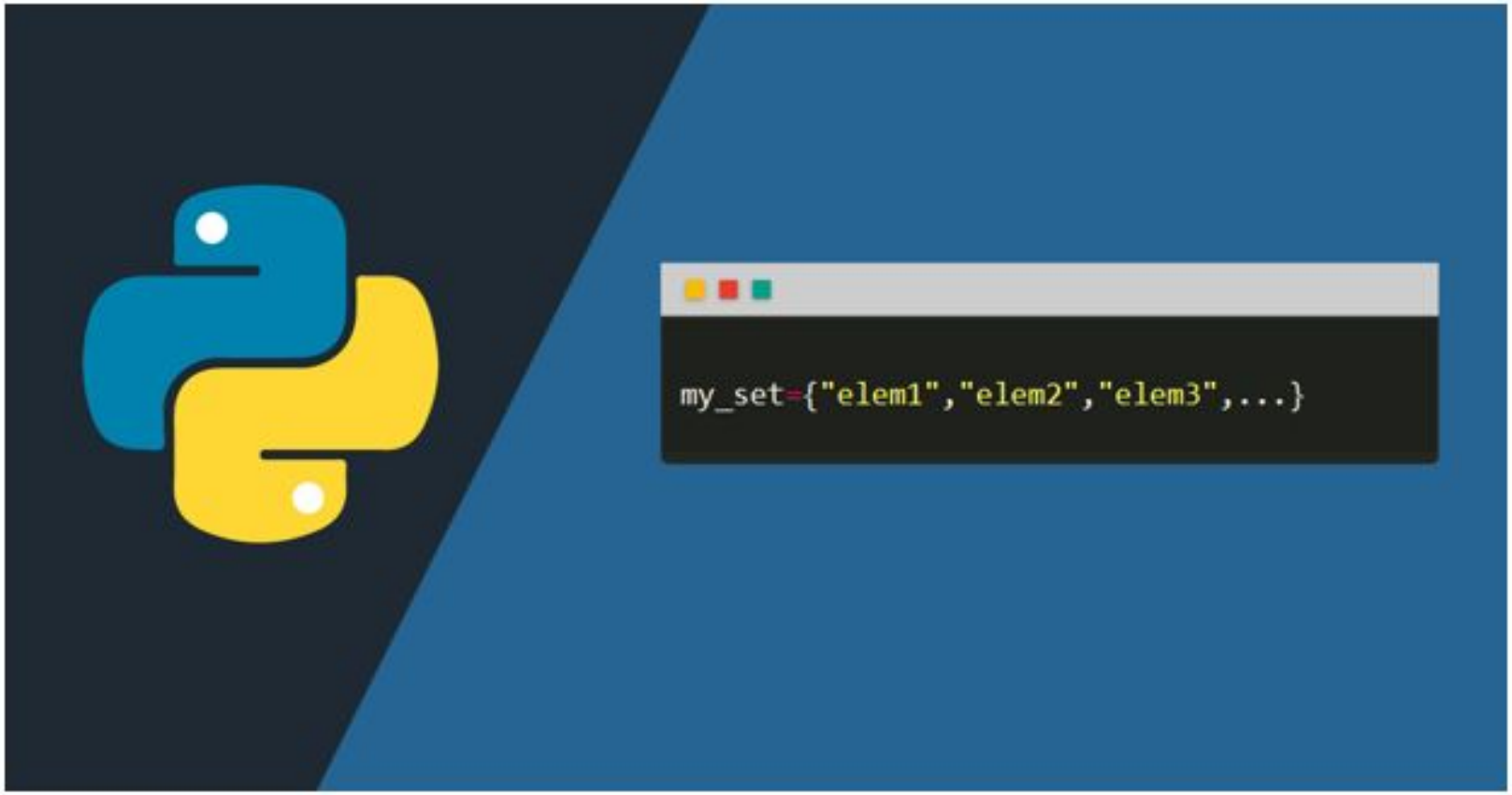
- Una tupla es una secuencia de objetos inmutables de Python. Las tuplas son secuencias, al igual que las listas.
- La diferencia entre las tuplas y las listas es que los elementos de la tupla no se pueden modificar. Además las tuplas se declaran con paréntesis, a diferencia de las listas que se declaran con corchetes.

Tuplas

Conjuntos

- El conjunto se compone de una lista no ordenada de datos en la cual no hay ningún dato repetido.
- Funcionan como un conjunto matemático, permitiendo realizar operaciones de conjuntos.

Conjuntos



Conjuntos

Método	Descripción
`set.add(x)`	Agrega x al conjunto si no se encuentra en él.
`set.clear()`	Elimina todos los elementos del conjunto.
`set.copy()`	Regresa una copia del conjunto.
`set.remove(x)`	Elimina el elemento x del conjunto si existe. Si no existe, se lanza un error.
`set.discard(x)`	Elimina el elemento x del conjunto si existe.
`set.pop()`	Elimina y regresa un elemento arbitrario del conjunto.

Conjuntos

Método	Descripción
`set.union(set2)`	Realiza una unión de conjuntos entre `set` y `set2`.
`set.intersection(set2)`	Realiza una intersección de conjuntos entre `set` y `set2`.
`set.difference(set2)`	Realiza una diferencia de conjuntos entre `set` y `set2`.
`set.isdisjoint(set2)`	Regresa True si la intersección entre `set` y `set2` es nula.
`set.issubset(set2)`	Regresa True si `set` es un subconjunto de `set2`.
`set.issuperset(set2)`	Regresa True si `set2` es un subconjunto de `set`.

Diccionarios

- Consiste en varios pares llave-valor, donde cada llave tiene asociado un valor y sólo podemos acceder a dicho valor por medio de su llave.
- La llave es inmutable y representa un mapeo de los objetos, por lo que debe ser única.

Diccionarios

Diccionarios

Método	Descripción
`dic.clear()`	Elimina todos los elementos del diccionario.
`dic.copy()`	Regresa una copia del diccionario.
`dic.fromkeys(sec, valor)`	Crea un nuevo diccionario utilizando `sec` (una secuencia) para las llaves y `valor` como valor a asignar a cada llave.
`dic.get(llave)`	Regresa el valor correspondiente a la `llave`.

Diccionarios

Método	Descripción
`dic.keys()`	Regresa una lista con todas las llaves del diccionario.
`dic.update(dic2)`	Agrega los pares llave-valor de otro diccionario (`dic2`) al diccionario.
`dic.values()`	Regresa una lista con todos los valores del diccionario.
`dic.items()`	Regresa una lista con tuplas de los pares llave-valor del diccionario.

Operaciones sobre secuencias

- Funciones que pueden aplicarse sobre cadenas, listas, tuplas, conjuntos y diccionarios:

Función	Descripción
`len(sec)`	Regresa la longitud de la secuencia `sec`.
`min(sec)`	Regresa el elemento con el menor valor de la secuencia `sec`.
`max(sec)`	Regresa el elemento con el mayor valor de la secuencia `sec`.

Mutabilidad e inmutabilidad

- Característica de un objeto donde su estado no puede ser alterado una vez creado.
- El objeto puede ser reemplazado por completo, pero no se puede alterar sin cambiar su identidad (id).

Tipos de Datos Inmutables

Tipo (clase)	Cualidad
Numérico (int, float, complex)	Inmutable
Cadena (str)	Inmutable
Lista (list)	Mutable
Tupla (tuple)	Inmutable
Conjunto (set)	Mutable
Conjunto inmutable (frozenset)	Inmutable
Diccionario (dict)	Mutable

Operadores

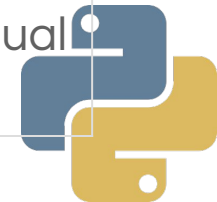
- Los operadores son las construcciones que pueden manipular el valor de los operandos.
- Considere la expresión 4 + 5 . Aquí, 4 y 5 se llaman operandos y + se llama operador.

Operadores aritméticos

Operador	Descripción
+	Suma los valores de ambos lados del operador.
-	Resta el valor del operando derecho del izquierdo.
*	Multiplica los valores de ambos lados del operador.
/	Divide el valor del operando izquierdo entre el derecho.
%	Regresa el residuo de la división (/) entre dos operandos.
**	Eleva el valor del operando izquierdo a la potencia del derecho.
//	Realiza una división (/) entre dos operandos y trunca los decimales.

Operadores de comparación

Operador	Descripción
==	Verifica si dos operandos son iguales.
!=	Verifica si dos operandos son diferentes.
>	Verifica si el valor a la izquierda del operador es mayor que el de la derecha.
<	Verifica si el valor a la izquierda del operador es menor que el de la derecha.
>=	Verifica si el valor a la izquierda del operador es mayor o igual que el de la derecha.
<=	Verifica si el valor a la izquierda del operador es menor o igual que el de la derecha.



Operadores de asignación

Operador	Descripción
=	Asigna al operando izquierdo el valor del derecho.
+=	a += b es equivalente a a = a + b.
-=	a -= b es equivalente a a = a - b.
*=	a *= b es equivalente a a = a * b.
/=	a /= b es equivalente a a = a / b.
%=	a %= b es equivalente a a = a % b.
**=	a **= b es equivalente a a = a ** b.
//=	a //= b es equivalente a a = a // b.

Operadores de bits

Operador	Descripción
&	Operación AND a nivel de bits entre dos operandos.
|	Operación OR a nivel de bits entre dos operandos.
^	Operación XOR a nivel de bits entre dos operandos.
~	Realiza el complemento a dos de un operando.
<<	Corrimiento de bits a la izquierda. Se recorre el operando izquierdo la cantidad de bits indicada por el operando derecho.
>>	Corrimiento de bits a la derecha. Se recorre el operando izquierdo la cantidad de bits indicada por el operando derecho.

Operadores lógicos

Operador	Descripción
`and`	Realiza una operación de conjunción lógica entre dos operandos de tipo booleano.
`or`	Realiza una operación de disyunción lógica entre dos operandos de tipo booleano.
`not`	Invierte el valor lógico de un operando booleano.

Operadores de pertenencia

Operador	Descripción
`in`	Regresa True si el operando de la izquierda se encuentra dentro de la secuencia de la derecha.
`not in`	Regresa True si el operando de la izquierda NO se encuentra dentro de la secuencia de la derecha.

Operadores de identidad

Operador	Descripción
`is`	Regresa True si ambos operandos apuntan al mismo objeto.
`is not`	Regresa True si ambos operandos no apuntan al mismo objeto.