UPB Formación Continua
Modalidad Virtual

Clase 2
Tipos de datos

En programación, un tipo de dato se refiere al valor que puede ser almacenado en una variable, y este valor puede comportarse de manera diferente según su tipo

Los tipos de datos de Python son **categorizaciones de datos** que representan diferentes tipos de datos.

Estos datos definen cómo se puede almacenar y manipular información.

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos

En Python, se clasifica los tipos de datos en dos grandes grupos:

1. Tipos de datos primitivos.
2. Tipos de datos compuestos

1. Tipos de datos primitivos
Los tipos de datos primitivos son aquellos que no se pueden descomponer en partes más pequeñas y se manipulan directamente con su valor. En Python, hay 5 tipos de datos primitivos fundamentales:

- Strings (str).
- Integers (int).
- Floats (float).
- Booleans (bool)
- None (caso especial).

Mgr. Jose Luis Vera

Tipos de datos

Strings

En Python, los Strings representan una cadena de texto y son un tipo de dato inmutable, lo que significa que no se pueden modificar después de ser creados

Es importante destacar que incluso las cadenas vacías son consideradas Strings y admiten todas las operaciones que se aplican a las cadenas en general.

Esto incluye métodos y operaciones que se pueden realizar con Strings, aunque estén vacíos.

La inmutabilidad de los Strings implica que, una vez que se crea una cadena, su contenido no puede ser alterado directamente, sino que se deben crear nuevas cadenas para realizar cambios.

```
>>> name = "Hello world!"
    empty_string = ""
>>> nombre=" jose luis vera"
>>> cadena_vacia=""
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Los métodos y propiedades más utilizadas para trabajar con cadenas de texto son los siguientes:

1- len(): Devuelve la longitud (número de caracteres) de la cadena.

```
>>> longitud=len(name)
>>> print(longitud)
 12
>>> len2=len(empty_string)
>>> print(len2)
 0
>>>
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

2-upper() y lower(): Convierten la cadena a mayúsculas o minúsculas, respectivamente

```
>>> cadena="Hola Mundo"
>>> mayus=cadena.upper()
>>> minus=cadena.lower()
>>> print(mayus)
 HOLA MUNDO
>>> print(minus)
 hola mundo
>>>
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

3-capitalize() y title(): capitalize() convierte el primer carácter a mayúsculas, y title() convierte el primer carácter de cada palabra a mayúsculas.

```
>>> cadena="hola mundo"
>>> capitalized=cadena.capitalize()
>>> con_titulo=cadena.title()
>>> print(capitalized)
 Hola mundo
>>> print(con_titulo)
 Hola Mundo
```

Mgr. Jose Luis Vera

UPB Formación Continua

Modalidad Virtual

Tipos de datos/Métodos

4- count(subcadena):

Cuenta cuántas veces aparece una subcadena en la cadena.

```
>>> repeticiones=cadena.count("hola")
>>> cadena="hola, Hola, hola, mundo"
>>> repeticiones=cadena.count("hola")
>>> print(repeticiones)
 2
```

Mgr. Jose Luis Vera

UPB Formación Continua

Modalidad Virt

Tipos de datos/Métodos

Integers

En Python, el tipo de dato Int representa números enteros, es decir, números sin parte fraccionaria o decimal. Los enteros pueden ser positivos, negativos o cero.

```
>>> natural=34
>>> negativo=-56
>>> cero=0
>>> print(type(natural))
 <class 'int'>
>>> print(type(negativo))
 <class 'int'>
>>> print(type(cero))
 <class 'int'>
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Los métodos y propiedades más utilizadas para trabajar con enteros son los siguientes:

1- bit_length(): Devuelve el número de bits necesarios para representar el entero en binario, excluyendo el signo y los ceros iniciales

```
>>> numero= 42
>>> bits=numero.bit_length()
>>> print(bits)
 6
```

2- abs(): Devuelve el valor absoluto del entero

```
>>> numero=-45
>>> n_absoluto=abs(numero)
>>> print(n_absoluto)
 45
```

3- round(): Redondea el entero a un número especificado de dígitos decimales (en este caso, a 0 dígitos decimales).

```
>>> num = 3.14159
>>> redondeado=round(num)
>>> print(redondeado)
 3
```

```
>>> num = 3.14159
>>> redondeado= round(num,4)
>>> print(redondeado)
 3.1416
```

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

4- int(x): Convierte otro tipo de dato (como un float o una cadena) a un entero.

```
>>> num=3.14
>>> numero_entero=int(num)
>>> print(numero_entero)
 3
```

5- bin(), oct(), hex(): Convierten el entero a su representación binaria, octal o hexadecimal, respectivamente

```
>>> num=42
>>> binario=bin(num)
>>> octal=oct(num)
>>> headecimal=hex(num)
```

```
>>> print(binario)
 0b101010
>>> print(octal)
 0o52
>>> print(headecimal)
 0x2a
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

6- chr(): Convierte un entero en su representación de carácter Unicode

```
>>> codigo_unicode = 65
>>> caracter = chr(codigo_unicode)
>>> print(caracter)
A
```

7- ord(): Devuelve el valor Unicode del carácter.

```
>>> caracter = 'A'
>>> unicode_valor = ord(caracter)
>>> print(unicode_valor)
65
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Floats

En Python, el tipo de dato float se utiliza para representar números de punto flotante, es decir, números que pueden tener una parte fraccionaria o decimal.

```
>>> float1=3.14
>>> float2=-3.6
>>> print(float1,float2)
3.14 -3.6
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Los métodos y propiedades más utilizadas para trabajar con flotantes son los siguientes:

1- is_integer(): Comprueba si el número de punto flotante tiene una parte decimal igual a cero

```
>>> numero=6.0
>>> es_entero=numero.is_integer()
>>> print(es_entero)
 True
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

2- max() y min(): Encuentran el valor máximo o mínimo entre varios números de punto flotante

```
>>> numeros=(2.0, 4.5, 1.2)
>>> type(numeros)
<class 'tuple'>
>>> print(max(numeros))
 4.5
>>> print(min(numeros))
 1.2
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

3- math.sqrt(): Calcula la raíz cuadrada de un número de punto flotante.

Para la utilización del método sqrt(), es necesario incluir una librería Esta es la librería **math**

Para ello, se utiliza la instrucción **import**

```
>>> import math
>>> raiz=math.sqrt(16.0)
>>> print(raiz)
 4.0
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

3- math.sqrt(): Calcula la raíz cuadrada de un número de punto flotante.

```
>>> import math
>>> numero=49.00
>>> raiz=math.sqrt(numero)
>>> print(raiz)
 7.0
>>> numero2=49
>>> raiz2=math.sqrt(numero2)
>>> print(raiz2)
 7.0
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Booleans

En Python, el tipo de dato bool (booleano) representa los valores de verdad: True (verdadero) o False (falso).

Los booleanos son utilizados en expresiones lógicas y en la toma de decisiones en estructuras de control de flujo, como declaraciones if, bucles while y for, entre otros

Ejemplos

```
>>> 6==7
False
>>> 10==10
True
```

```
>>> numero1=12345
>>> numero2=12346
>>> numero1==numero2
False
```

```
>>> numero1=12345
>>> numero3=12345.0
>>>
>>> numero1==numero3
True
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

2. Tipos de datos compuestos

Los tipos de datos compuestos en Python son aquellos que permiten agrupar múltiples tipos de datos en una única variable.

Pueden contener tanto tipos de datos primitivos como otros tipos de datos compuestos.

Estos tipos de datos ofrecen la flexibilidad de estructurar y organizar información de manera más compleja, permitiendo la creación de estructuras más avanzadas.

Mgr. Jose Luis Vera

Tipos de datos/Métodos

En Python existen 2 principales tipos de datos compuestos, los cuales son:

- Listas (list).
- Diccionarios (dict).

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Listas

En Python, una lista es un tipo de dato mutable que sirve como equivalente a los arrays en otros lenguajes de programación.
Los elementos almacenados dentro de una lista pueden ser modificados después de su creación.
Se accede a estos elementos mediante índices, y se almacenan en ubicaciones contiguas y sucesivas en la memoria.
Las listas proporcionan una forma eficiente de organizar y manipular conjuntos de datos, permitiendo un acceso rápido y estructurado a través de índices numéricos

```
>>> frutas=["manzana","naranja","sandía", "melon","piña"]
>>> print(frutas[2])
 sandía
>>> print(frutas[0])
 manzana
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Los métodos y propiedades más utilizadas para trabajar con listas son los siguientes:
1- append(elemento): Agrega un elemento al final de la lista.

```
>>> frutas.append("mandarina")
>>> frutas=["manzana","naranja","sandía", "melon","piña"]
>>> frutas.append("mandarina")
>>> print(frutas)
 ['manzana', 'naranja', 'sandía', 'melon', 'piña', 'mandarina']
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

2- extend(iterable): Agrega los elementos de un iterable (como otra lista) al final de la lista.

```
>>> lista_1=["calos", "marcelo", "jose"]
>>> lista_2=["lorena", "maría", "fernanda"]
>>> listaFinal=lista_1+lista_2
>>> print(listaFinal)
 ['calos', 'marcelo', 'jose', 'lorena', 'maría', 'fernanda']
>>> miLista=[]
>>> miLista.extend(lista_1)
>>> print(miLista)
 ['calos', 'marcelo', 'jose']
>>> miLista.extend(lista_2)
>>> miLista
['calos', 'marcelo', 'jose', 'lorena', 'maría', 'fernanda']
```

Concatenación simple

Agregado con el método **extend**

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

3- insert(posición, elemento): Inserta un elemento en una posición específica de la lista.

```
>>> miLista=lista_1
>>> miLista
['calos', 'marcelo', 'jose']
>>> miLista.insert(2,"CINTHIA")
>>> miLista
['calos', 'marcelo', 'CINTHIA', 'jose']
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

4- remove(elemento): Elimina la primera aparición del elemento en la lista

```
>>> miLista
['calos', 'marcelo', 'CINTHIA', 'jose']
>>> miLista.insert(0,"CINTHIA")
>>> miLista
['CINTHIA', 'calos', 'marcelo', 'CINTHIA', 'jose']
>>> miLista.remove("CINTHIA")
>>> miLista
['calos', 'marcelo', 'CINTHIA', 'jose']
>>>
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

4- remove(elemento): Elimina la primera aparición del elemento en la lista.

```
>>> miLista
['calos', 'marcelo', 'CINTHIA', 'jose']
>>> miLista.remove('marcelo')
>>> miLista
['calos', 'CINTHIA', 'jose']
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

5- pop([índice]): Elimina y devuelve el elemento en la posición dada (o el último si no se proporciona índice).

```
>>> nombre=miLista.pop()
>>> miLista
['carlos', 'marcelo', 'jose', 'lorena', 'maría']
>>> nombre_2=miLista.pop(2)
>>> nombre
'fernanda'
>>> nombre_2
'jose'
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

6- index(elemento[, inicio[, fin]]): Devuelve el índice de la primera aparición del elemento en la lista. Puedes especificar un rango opcional de búsqueda

```
>>> miLista
['carlos', 'marcelo', 'lorena', 'maría']
>>> posicion=miLista.index('lorena')
>>> posicion
2
>>> posicion=miLista.index('lorena',1,3)
>>> posicion
2
>>> posicion=miLista.index('lorena',2,3)
>>> posicion
2
>>> posicion=miLista.index('lorena',0,1)
 Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
 ValueError: 'lorena' is not in list
>>> posicion=miLista.index('lorena',0,3)
>>> posicion
2
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

7- count(elemento): Cuenta cuántas veces aparece el elemento en la lista

```
>>> miLista
['carlos', 'marcelo', 'lorena', 'maría']
>>> miLista.extend(["carlos", "maría"])
>>> miLista
['carlos', 'marcelo', 'lorena', 'maría', 'carlos', 'maría']
>>> repeticiones=miLista.count('maría')
>>> repeticiones
2
>>> miLista.count('lorena')
1
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

8-sort(): Ordena la lista in-place (**modifica la lista original**).

```
>>> miLista=lista_1+lista_2
>>> miLista
['carlos', 'carlos', 'lorena', 'marcelo', 'maría', 'maría', 'lorena', 'm
aría', 'fernanda']
>>> miLista.sort()
>>> miLista
['carlos', 'carlos', 'fernanda', 'lorena', 'lorena', 'marcelo', 'maría',
'maría', 'maría']
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

9- reverse(): Invierte el orden de los elementos en la lista.

```
>>> miLista=lista_1+lista_2
>>> miLista.reverse()
>>> miLista
['fernanda', 'maría', 'lorena', 'maría', 'maría', 'marcelo', 'lorena', '
carlos', 'carlos']
>>>
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

10- copy() o [:]: Crea una copia superficial de la lista

```
>>> miLista=lista_1+lista_2
>>> copia_l=miLista.copy()
>>> copia_l
['carlos', 'carlos', 'lorena', 'marcelo', 'maría', 'maría', 'lorena', 'm
aría', 'fernanda']
>>> copia_l_2=miLista[:]
>>> copia_l_2
['carlos', 'carlos', 'lorena', 'marcelo', 'maría', 'maría', 'lorena', 'm
aría', 'fernanda']
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Copia un rango de elementos de la lista original

```
>>> miLista
['carlos', 'carlos', 'lorena', 'marcelo', 'maría', 'maría', 'lorena', 'm
aría', 'fernanda']
>>> copia_parcial=miLista[3:]
>>> copia_parcial
['marcelo', 'maría', 'maría', 'lorena', 'maría', 'fernanda']
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

En Python, un diccionario (dict) es un tipo de dato *mutable* que se asemeja a un objeto en JavaScript.

Los elementos dentro de un diccionario se almacenan en pares de clave y valor.

Acceder a un elemento dentro del diccionario se realiza mediante su clave, y esto devuelve el valor asociado.

Los diccionarios proporcionan una manera versátil de organizar y estructurar datos, permitiendo la representación de relaciones más complejas mediante la asociación de claves y valores.

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> usuario
{'nombre': 'jose luis', 'Celular': 70179815, 'correo': 'joseluis@hotmail
.com', 'es-papa': True}
>>> print(usuario)
 {'nombre': 'jose luis', 'Celular': 70179815, 'correo': 'joseluis@hotma
 il.com', 'es-papa': True}
>>>
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

Los métodos y propiedades más utilizadas para trabajar con diccionarios son los siguientes:

1- keys(): Devuelve una vista de todas las claves en el diccionario

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> keys_diccionario=usuario.keys()
>>> keys_diccionario
dict_keys(['nombre', 'Celular', 'correo', 'es-papa'])
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

2- values(): Devuelve una vista de todos los valores en el diccionario.

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> valores_usuario=usuario.values()
>>> valores_usuario
dict_values(['jose luis', 70179815, 'joseluis@hotmail.com', True])
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Diccionarios

3- items(): Devuelve una vista de tuplas que contienen pares clave-valor en el diccionario

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> usuario.items()
dict_items([('nombre', 'jose luis'), ('Celular', 70179815), ('correo', '
joseluis@hotmail.com'), ('es-papa', True)])
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Diccionarios

4- get(key[, default]): Devuelve el valor asociado a la clave, o un valor predeterminado si la clave no está presente.

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> valor=usuario.get('Celular')
>>> valor
70179815
>>> valor=usuario.get('celular',0)
>>> valor
0
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

5- pop(key[, default]): Elimina y devuelve el valor asociado a la clave. Si la clave no está presente, devuelve un valor predeterminado, o genera un error si no se proporciona un valor predeterminado

```
>>> usuario={
        'nombre': 'jose luis',
        'Celular': 70179815,
        'correo': 'joseluis@hotmail.com',
        'es-papa':True
        }
>>> valor_borrado=usuario.pop('es-papa')
>>> valor_borrado
True
>>> usuario
{'nombre': 'jose luis', 'Celular': 70179815, 'correo': 'joseluis@hotmail.com'}
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

Diccionarios

6- popitem(): Elimina y devuelve el último par clave-valor insertado en el diccionario.

```
>>> estaturas={
        'jose':1.70,
        'carlos':1.80,
        'mário': 1.58,
        'marcela': 1.67
        }
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67}
>>> par_eliminado=estaturas.popitem()
>>> par_eliminado
('marcela', 1.67)
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58}
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Diccionarios

7- update(iterable): Actualiza el diccionario con elementos de otro diccionario o de un iterable de pares clave-valor.

```
>>> estaturas={
        'jose':1.70,
        'carlos':1.80,
        'mário': 1.58,
        'marcela': 1.67
        }
>>> estaturas_2={
        'ricardo':1.87,
        'paolita': 1.68,
        'tabata':1.58
    }
>>> estaturas.update(estaturas_2)
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67, 'ricardo': 1.87, '
paolita': 1.68, 'tabata': 1.58}
```

Mgr. Jose Luis Vera

UPB Formación Continua
Modalidad Virtual

Tipos de datos/Métodos

Diccionarios

8- clear(): Elimina todos los elementos del diccionario

```
>>> diccionario_extra=estaturas.copy()
>>> diccionario_extra
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67, 'ricardo': 1.87,
paolita': 1.68, 'tabata': 1.58}
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67, 'ricardo': 1.87,
paolita': 1.68, 'tabata': 1.58}
>>> diccionario_extra.clear()
>>> diccionario_extra
{}
>>>
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

UPB Formación Continua
Modalidad Virtual

Diccionarios

9- in operator: Se utiliza para verificar si una clave está presente en el diccionario.

```
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67, 'ricardo': 1.87,
paolita': 1.68, 'tabata': 1.58}
>>> esta_en_diccionario='tabata'in estaturas
>>> esta_en_diccionario
True
>>> esta_en_diccionario='carlitos'in estaturas
>>> esta_en_diccionario
False
```

Mgr. Jose Luis Vera

Tipos de datos/Métodos

UPB Formación Continua
Modalidad Virtual

Diccionarios

10- len(): Devuelve la cantidad de elementos (pares clave-valor) en el diccionario.

```
>>> estaturas
{'jose': 1.7, 'carlos': 1.8, 'mário': 1.58, 'marcela': 1.67, 'ricardo': 1.87, '
paolita': 1.68, 'tabata': 1.58}
>>> num_elementos=len(estaturas)
>>> num_elementos
7
```

Mgr. Jose Luis Vera

UPB Formación Continua

Modalidad Virtual

Tipos de datos/Métodos

FIN

Mgr. Jose Luis Vera