Fundamentos de Programación y Estructura de Datos

Sesión Conceptual 1

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercicios
- Quiz

- Cierre

Inicio

10 minutos

{desafío} latam_

Objetivo

/* Identificar los hábitos de estudio y estrategias de estudio necesarias para el desarrollo de competencias.*/

Desarrollo

70 minutos

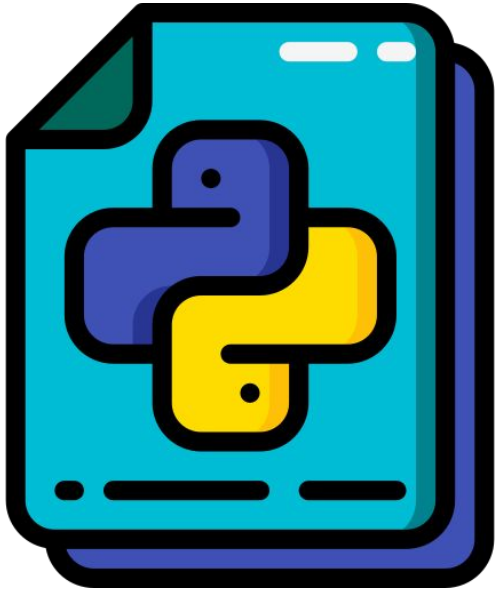
{desafío} latam_

/* Conociendo Python */

{desafío}
latam_

Aspectos generales

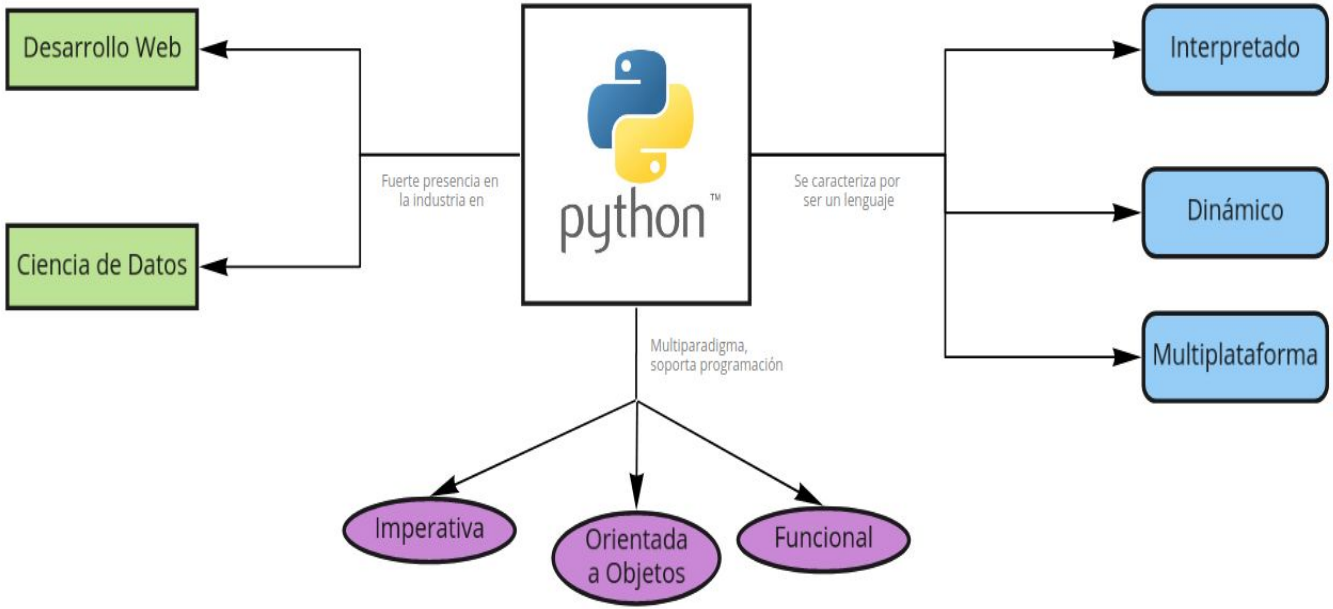
En este capítulo, revisaremos los aspectos más importantes para poder ejecutar un código mediante el Lenguaje de programación Python.



¿Qué es Python?

Python es un lenguaje de programación interpretado y multiparadigma, ya que soporta parcialmente la **orientación a objetos**, **programación imperativa** y, en menor medida, **programación funcional.** Es un lenguaje **interpretado**, **dinámico** y **multiplataforma**, y es administrado por la Python Software Foundation y posee una licencia de código abierto.

Principales características de Python



Ejercicio guiado

"Mi primer código en Python"

{desafío} latam_

Salida de datos: print()

print es una función para mostrar valores de variables o resultados de operaciones en la pantalla.

`"hola mundo"` es el argumento de la función print, y Python mostrará dicho valor en pantalla.

Comentarios

Todo texto a continuación de un signo `#` es un **comentario**.

```
# Esta línea es un comentario
# Los comentarios son ignorados
# Pueden ser una línea nueva
print(2 + 2)  # O puede acompañar una línea de código existente
# print(2 + 3)
# Si comentamos un código válido en Python, no se ejecutará

4
```

Comentarios en múltiples líneas

Consiste en envolver todo el comentario entre tres comillas dobles al

```
"""
Comentario multilínea:
Python
lo
ignorará
"""

print("hola")
```

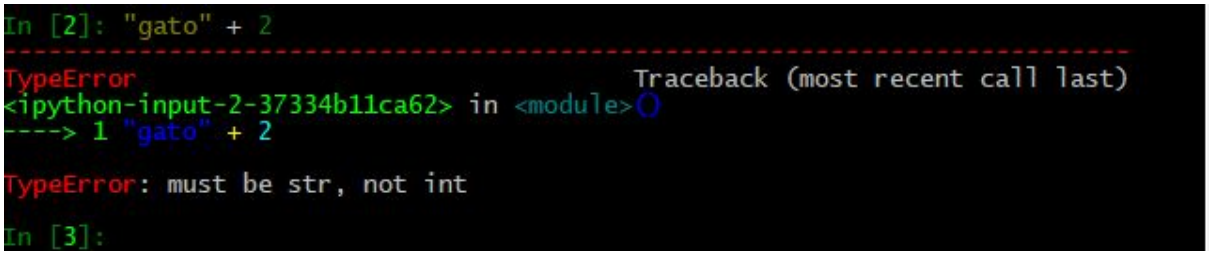
Python incluye operaciones básicas como las sumas, restas, multiplicaciones, divisiones, entre otros.

Python como una calculadora

{desafío} latam_

Limitantes

Si bien Python se caracteriza por ser un lenguaje de programación de sintaxis simple, esto no significa que no tenga reglas para escribir el código, como por ejemplo no permite sumar letras y números.



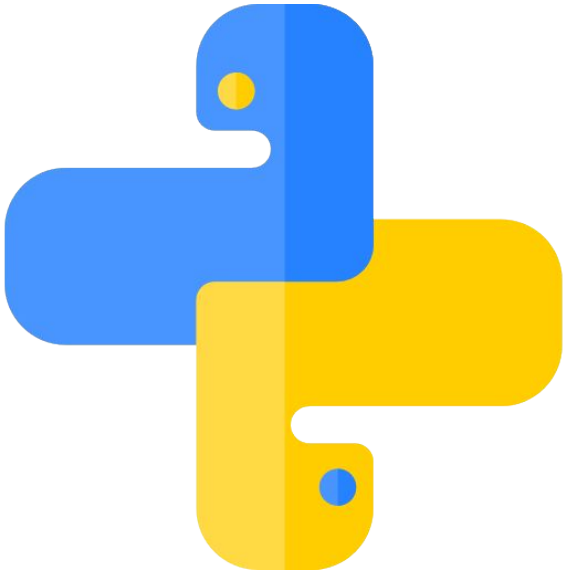
Quiz

{desafío}
latam_

/* Reglas básicas en Python */

{desafío}
latam_

Aspectos generales



Python tiene reglas, y en este capítulo entenderemos una de ellas: los tipos de datos, los que al ser utilizados de manera apropiada, evitará errores indeseados que impidan el correcto funcionamiento de nuestros programas.

Tipos de datos

Valores numéricos

En Python existen principalmente 2 tipos de datos para representar números: los **integers** y los **float**. Un integer será un número entero, es decir, sin decimales, en cambio, un float será un número que incluye decimales.

```
print(3.5 + 12)
print(20 * 45.6)
```

Strings

Los strings son **elementos encerrados entre comilla simple (') o comilla doble (").** Para definirse como un string válido, la comilla al inicio y al final debe ser la misma.

```
# estos son valores string
print('hola')
print("150")
print('este es un texto más largo')
```

Tipos de datos

Concatenación

Si intentamos sumar dos strings obtendremos una concatenación.

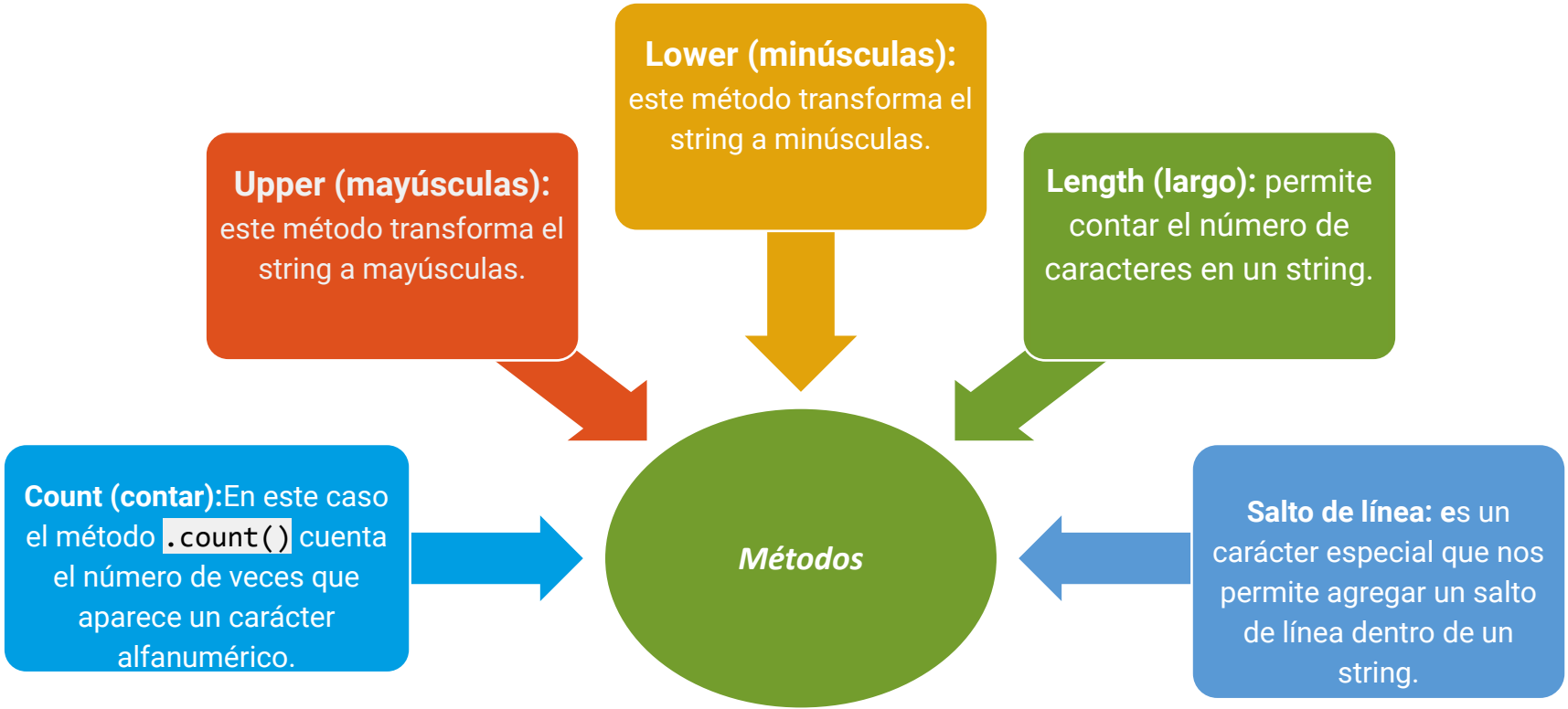
```
# Concatenación
print("Carlos" + "Santana") #
CarlosSantana
print("Carlos " + "Santana") #
Carlos Santana
```

Duplicación

Si intentamos multiplicar un string por un integer positivo, entonces este string se duplicará.

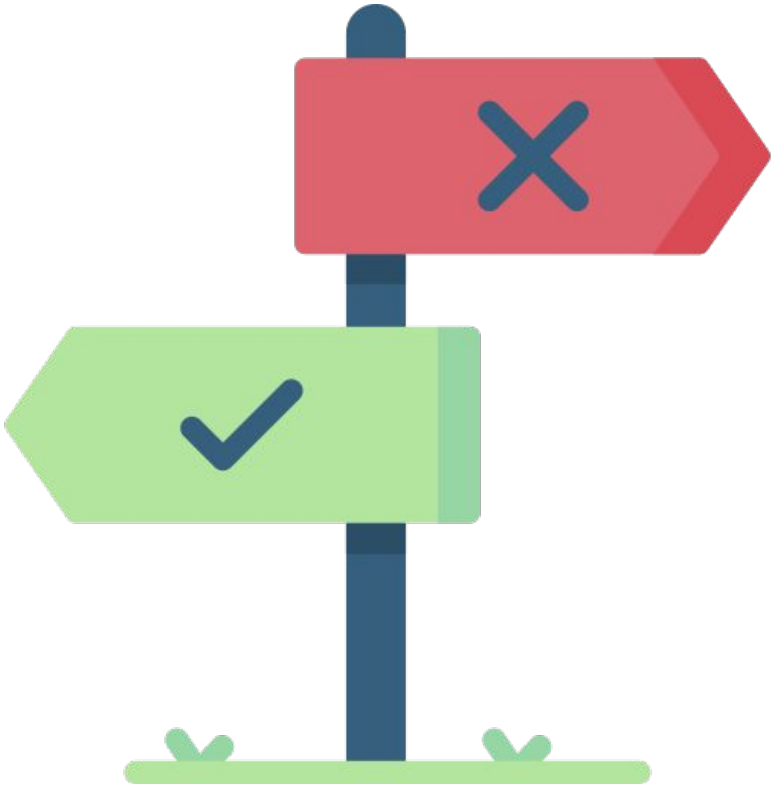
```
# Duplicación
print(3 * "Carlos") # CarlosCarlosCarlos
print(5 * "12") # 1212121212
```

Métodos


Lower (minúsculas): este método transforma el string a minúsculas.
Upper (mayúsculas): este método transforma el string a mayúsculas.
Length (largo): permite contar el número de caracteres en un string.
Count (contar):En este caso el método .count() cuenta el número de veces que aparece un carácter alfanumérico.
Métodos
Salto de línea: es un carácter especial que nos permite agregar un salto de línea dentro de un string.

Valores Booleanos

Son valores lógicos que pueden tomar sólo dos valores: **True** y **False**.



Asignando un valor a una variable

Podemos entender las variables como contenedores que pueden almacenar valores. Una variable se compone de:

- Un nombre.
- Un valor.

Además deben:

- Comenzar con minúscula.
- Unir las palabras por un guión bajo.
- No pueden tener espacios ni comenzar con un número.
- Poseer un tipo de dato asociado, según el valor que se le asigne.

Transformando los datos

Interpolación

Es un mecanismo que nos permite introducir una variable, un dato o una operación válida de Python dentro un String. Para interpolar simplemente tenemos que introducir la variable (o dato) y estos serán agregados dentro de las llaves {}.

```
nombre = 'Carlos'
apellido = 'Santana'
# Interpolación
print("Mi nombre es {} {}".format(nombre,
apellido))

Mi nombre es Carlos Santana
```

Precisión de decimales

Una propiedad bastante cómoda que tiene el **f-string** es que permite controlar la precisión de los decimales de manera muy sencilla.

```
# Muestro sólo 2 decimales
print(f'El resultado es {1/9:.2f}')

El resultado es 0.11
```

{desafío} latam_

Ingresando datos de manera interactiva

Con frecuencia nuestro script necesitará interactuar con el usuario, ya sea para seleccionar una opción de un menú o para simplemente ingresar un valor sobre el cuál nuestro script va a operar. Podemos capturar datos introducidos por un usuario ocupando la instrucción `input`.

Ingreso de datos con input:
Al invocar `input()` la consola quedará bloqueada hasta que ingresemos una secuencia de caracteres y presionemos la tecla enter.

```
>>> input("Ingrese su Nombre: ")
Ingrese su Nombre: Carlos Santana
```

```
>>> nombre = input("Ingrese su Nombre: ")
Ingrese su Nombre: Carlos Santana
```

Ejercicio guiado

"Presentándome con Python"

{desafío} latam_

Otros operadores Matemáticos

Podemos definir formalmente los operadores nativos en Python, donde además de los +, -, *, y / podemos adicionar los siguientes:

```
2 ** 4 # Exponenciación/Potencia. Ej: 2 elevado a 4 es 16
5 // 2 # División Entera. Ej: 5 dividido 2 es 2.
5 % 2 # Módulo. Ej: 5 Módulo 2 es 1. 1 es el resto de 5 // 2.
```

Precedencia de operadores

La precedencia es saber en qué orden se realiza un grupo de operaciones, y podemos conocerla en detalle en la siguiente tabla:

Operador	Nombre
**	Exponenciación (potencia)
* , /, //, %	Multiplicación, división y módulo
+ , -	Suma y resta

Al igual que en matemáticas, los paréntesis cambian el orden en que preceden las operaciones dando prioridad a las operaciones que estén dentro de los paréntesis.

```
print((10 - 5) * 2)
10
```

Librerías/Módulos

Importar librería

Las librerías son extensiones del lenguaje que añaden funcionalidades no disponibles en Python nativo.

```
# esta es una librería que permite
interactuar con el Sistema Operativo
import os
# añade más funcionalidades matemáticas
import math
```

Instalar librerias

Para trabajar con panda, **pip** es el instalador por defecto de PyPI, el cual es uno de los repositorios de librerías con los que cuenta Python. Mediante `pip install` es posible entonces instalar cualquier librería que resida en este repositorio.

Para trabajar con conda, el comando escogido es `conda install.`

Ejercicio guiado

"Contando calorías"

{desafío} latam_

Quiz

{desafío}
latam_

Cierre

10 minutos

{desafío} latam_

Preguntas de cierre

1. ¿Qué es Python y por qué es tan demandado en la Industria?
2. ¿Por qué es importante conocer las reglas básicas del lenguaje Python?
3. ¿Cuáles son los tipos de datos nativos en Python?
4. ¿Es posible sumar cualquier tipo de dato en Python?
5. ¿De qué manera se permite al usuario ingresar datos en Python de manera interactiva?

{desafío} latam_

{desafío} latam_

Academia de talentos digitales