Material complementario

CODERHOUSE

# Glosario Python

Este glosario fue pensado para facilitarte los conceptos principales que veremos en el curso. ¡Te invitamos a que lo complementes con tus apuntes!.
Al finalizar cada definición encontrarás especificado en que clase se encuentra desarrollado en el caso de que quieras buscar más información al respecto.

## A

- **Add:** . Esta función permite agregar un nuevo ítem al set. La misma se escribe mi_conjunto.add(ítem_a_agregar). (Clase: 6)
- **Agregación:** (tipo de relación) Esta relación se presenta entre una clase TODO y una clase PARTE que es componente de TODO. (Clase: 12)
- **Archivos:** son la forma más antigua, primitiva y simple de almacenar datos. Pero créanlo o no, aún se sigue utilizando este mecanismo en más de un aplicativo; incluso en programas bastante sofisticados. (Clase: 8)
- **And:** Es el operador de conjunción, es decir, el que agrupa a través de la unión, es el operador lógico AND, en castellano conocido como Y. (Clase: 3)
- **Asociación:** (tipo de relación) Especifica una relación semántica entre objetos no relacionados. Este tipo de relaciones permiten crear asociaciones que capturan los participantes en una relación semántica. (Clase: 12)
- **Atributos:** Cosas de lo más cotidianas como una computadora o un coche pueden ser representadas con clases. Estas clases tienen diferentes características, que en el caso de la computadora podrían ser la marca o modelo. Llamaremos a estas características, atributos. (Clase: 13)

# Material complementario

CODERHOUSE

## B

- **BackEnd**: Es la parte que conecta con la base de datos y el servidor que usa el sitio web, se le dice lado del servidor.  Desarrollo de API del lado de los servidores y bases de datos. Se conecta la base de datos con la interfaz de la web/app. (Clase 0)
- **Base de datos:** alternativa para almacenar y explorar los datos. Estas son complejas y por lo general necesitan de la instalación de algún motor o programa que nos permita utilizarlas. (Clase: 8)
- **Break:** En Python, la instrucción break le proporciona la oportunidad de cerrar un bucle cuando se activa una condición externa. Debe poner la instrucción break dentro del bloque de código bajo la instrucción de su bucle, generalmente después de una sentencia if condicional. (Clase: 5)
- **Bucles en Python:** usar bucles en Python nos permite automatizar y repetir tareas de manera eficiente. Sin embargo, a veces, es posible que un factor externo influya en la forma en que se ejecuta su programa. (Clase: 5)

## C

- **Cadenas (o strings):**  son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str y se delimitan mediante el uso de comillas simples o dobles. (Clase 1)
- **Capitalize**: sta función integrada sirve para hacer que se devuelva la misma cadena pero con su primer carácter en mayúscula y el resto de caracteres hacerlos minúscula, usando el método capitalize().  Se escribe como:

string.capitalize() (Clase: 7)

- **Clear**: Igual que en las listas, podremos borrar todos los valores de un set simplemente usando la función clear. (Clase: 6)
- **Collections:** El módulo collections nos ayuda a completar y manipular las estructuras de datos de forma eficiente.
- **Copy:** esta función integrada sirve para hacer que se devuelva una copia de un set.  Se escribe como: set.copy() (Clase: 7)
- **Condicional:** las sentencias de control condicionales, son aquellas que nos permiten evaluar si una o más condiciones se cumplen, para decir qué acción vamos a ejecutar. La evaluación de condiciones, solo puede arrojar 1 de 2 resultados: True o False (verdadero o falso). (Clase: 4)
- **Conjunto o set:** s una colección no ordenada de objetos únicos, es decir, no tiene elementos duplicados. Python provee este tipo de datos por defecto al igual que otras colecciones más convencionales como las listas, tuplas y diccionarios. (Clase: 6)
- **Continue**: La instrucción continue da la opción de omitir la parte de un bucle en la que se activa una condición externa, pero continuar para completar el resto del bucle. Es decir, la iteración actual del bucle se interrumpirá, pero el programa volverá a la parte superior del bucle. (Clase:5)
- **Count:** Si necesitamos saber cuantas veces aparece una subcadena dentro de la misma cadena, usando el método count().  Se escribe como: string.count() (Clase: 7)
- **CRUD:** es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete). También es conocido como ABM en otros entornos (Altas, Bajas y Modificaciones). Es el acto de agregar, modificar, eliminar o acceder a datos de una base de datos. (Clase: 22)

# Material complementario

CODERHOUSE

D

- **Datos:** conjunto discreto, de factores objetivos sobre un hecho real. Dentro de un contexto empresarial, el concepto de dato es definido como un registro de transacciones. Un dato no dice nada sobre el por qué de las cosas y por sí mismo tiene poca o ninguna relevancia o propósito. (Clase 0)
- **Del**: to del dict, si el elemento pasado como argumento a del() no está dentro del dict es simplemente ignorado.Se escribe como del mi_dict[“clave”]. (Clase: 6)
- **Diagrama de flujo**: Expresan nuestros algoritmos en forma de diagrama mediante una representación gráfica basada en figuras geométricas que varían según la estructura de código. (Clase: 4)
- **Diccionario:** es una colección no ordenada de objetos. Es por eso que para identificar un valor cualquiera dentro de él, especificamos una clave (a diferencia de las listas y tuplas, cuyos elementos se identifican por su posición). (Clase:6)
- **Difference**: Esta función encuentra todos los elementos no comunes entre dos set, es decir, nos devuelve un set de ítems diferentes entre cada set. Se escribe como: set1.difference(set2) (Clase: 7)
- **Difference_update:** similar al difference, pero esta función nos guarda los ítems distintos en el set originales, es decir, le asigna como nuevo valor los ítems diferentes. Se escribe como: set1.difference_update(set2) (Clase: 7)
- **Discard:** Si add te deja agregar un ítem al set, discard hace todo lo contrario, elimina el ítem del set, sin modificar el resto del set, si el elemento pasado como argumento a discard() no está dentro del conjunto es simplemente ignorado. (Clase: 6)
- **Duck Typing:** El término polimorfismo visto desde el punto de vista de Python es complicado de explicar sin hablar del duck typing. Es un concepto que aplica a ciertos lenguajes orientados a objetos. (Clase: 14)

# Material complementario

CODERHOUSE

## E

- **Enumerate:** a función incorporada enumerate(lista/tupla_de_valores) toma como argumento un objeto iterable y retorna otro cuyos elementos son tuplas de dos objetos. (Clase: 5)
- **Entornos virtuales:** Los entornos virtuales se pueden describir como directorios de instalación aislados. Este aislamiento te permite localizar la instalación de las dependencias de tu proyecto, sin obligarte a instalarlas en todo el sistema. (Clase: 17)
- **Entrada o input:** es un programa son los datos que llegan al programa desde el exterior. Actualmente, el origen más habitual es el teclado. Python tiene una función llamada input() la cual permite obtener texto escrito por teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro. (Clase 1)
- **Excepciones:** Los errores detectados durante la ejecución se llaman excepciones, y no son incondicionalmente fatales. (Clase: 11)
- **Expresiones:** conjunto que forman los operandos y la operación. (Clase:3)
- **Expresiones anidadas:** que existen un montón de expresiones distintas y como pueden suponer, es posible crear combinaciones entre estas. A esto, se lo denomina expresiones anidadas. (Clase: 3)

## F

- **Find:** Si necesitamos averiguar el índice en el que aparece una subcadena dentro de la misma cadena, usamos el método find(). Se escribe como: string.find(). Si no encuentra la cadena devuelve un -1. (Clase: 7)

# Material complementario

CODERHOUSE

- **Flujo:** es una forma de entender la sucesión de las instrucciones de un programa, estas instrucciones se ejecutan una después de otras de forma ordenada y suelen tener el objetivo final de manipular información. (Clase:4)
- **Frameworks:** o marco de trabajo, es un conjunto de herramientas y clases que nos permiten solucionar un problema o funcionalidad. Están escritos en uno o más lenguajes de programación. Establecen, además, una estructura determinada para el código y todos los archivos, como así también una metodología armada del proyecto. (Clase 0)
  Es un entorno de trabajo, un ecosistema que nos facilita crear algo, en este caso un sitio web, sin mayores esfuerzos. (Clase: 17)
- **Funciones:** agrupaciones de código las cuales se pueden ejecutar múltiples veces gracias a un nombre único que las identifica. (Clase: 9)
- **Funciones integradas:** Los conjuntos en Python tienen muchas funciones para utilizar, entre todas ellas vamos a nombrar las más importantes. (Clase: 6)
- **FrontEnd:** Es la parte de un sitio web que interactúa con los usuarios, se le dice lado del cliente. Maquetación, lógica visible de un sitio web. Es lo que puede ver un cliente, y con lo que interactúa. (Clase 0)

## G

- **Git:** Git es un sistema de control de versiones gratuito y de código abierto, diseñado para manejar desde pequeños a grandes proyectos de manera rápida y eficaz. Se entiende como control de versiones a todas las herramientas que nos permiten hacer modificaciones en nuestro proyecto. Un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo. (Clase: 16)
- **Github:** Github es una especie de “red social” de programadores. Con este sitio podemos subir nuestros proyectos y lograr que otras personas colaboren.

# Material complementario

CODERHOUSE

## H

- **Heterogéneos**: En otros lenguajes, las colecciones tienen una restricción la cual solo permite tener un tipo de dato. Pero en Python, no tenemos esa restricción. Podemos tener un conjunto heterogéneo que contenga números, variables, strings, o tuplas. (Clase: 6)
- **Herencia:** es un proceso mediante el cual se puede crear una clase hija que hereda de una clase padre, compartiendo sus métodos y atributos. (Clase: 14)

## I

- **IDE:** es un sistema de software para el diseño de aplicaciones que combina herramientas comunes para desarrolladores en una sola interfaz de usuario gráfica (GUI). (Clase: 15)
- **In:** para determinar si un elemento pertenece a un dict, utilizamos la palabra reservada in. (Clase: 6)
- **Información**: tiene significado, relevancia y propósito. No solo puede formar potencialmente a quien la recibe, sino que está organizada para algún propósito. Los datos se convierten en información cuando su creador les añade significado. Transformamos datos en información añadiéndoles valor en diversos sentidos. (Clase 0)
- **Iterar:** En matemática, se refiere al proceso de iteración de una función, es decir, aplicando la función repetidamente, usando la salida de una iteración como la entrada a la siguiente.
  En programación, Iteración es la repetición de un segmento de código dentro de un programa de computadora. Puede usarse tanto como un término genérico (como sinónimo de repetición) como para describir una forma específica de repetición. (Clase:5)

# Material complementario

CODERHOUSE

- **Isdisjoint:** esta función comprueba si el set es distinto a otro set, es decir, si no hay ningún ítem en común entre ellos. Se escribe como: set1.isdisjoint(set2) (Clase: 7)
- **Issubset**: Esta función comprueba si el set es subset de otro set, es decir, si todos sus ítems están en el otro conjunto. Se escribe como: set1.issubset(set2) (Clase:7)
- **Issuperset**: sta función es muy similar al issubset, la diferencia es que esta comprueba si el set es contenedor de otro set, es decir, si contiene todos los ítems de otro set. Se escribe como: set1.issuperset(set2) (Clase: 7)

## J

- **Join:** Esta función integrada sirve para devolver una cadena separada a partir de una especie de separador. Se escribe como: "separador".join("cadena"). (Clase: 7)

## L

- **Listas**: en ellas hay funciones que son muy interesantes e importantes, las funciones integradas. Las listas en Python tienen muchas funciones para utilizar, entre todas ellas vamos a nombrar las más importantes. (Clase 2)
- **Lower:** se puede convertir una cadena de caracteres a minúsculas, usando el método lower(). Se escribe como: string.lower() (Clase: 7)

# Material complementario

CODERHOUSE

## M

- **Math:** módulo incorporado que puede usarse para tareas matemáticas. (Clase: 15)
- **Métodos:** Por otro lado, las clases tienen un conjunto de funcionalidades. En el caso de la computadora  podría ser imprimir o reproducir. Llamaremos a estas funcionalidades métodos. (Clase 13)
- **Métodos especiales:** Los métodos mágicos de las clases Python son aquellos que comienzan y terminan con dos caracteres subrayados. Son de suma importancia ya que muchas de las operaciones que se hacen con clases en Python utilizan estos métodos, como puede ser la creación del objeto (__init__) o generar una cadena que los represente (__str__).  (Clase: 13)
- **Metodologías ágiles**: Son las metodologías que permiten adaptar la forma de trabajo a las condiciones del proyecto, dando flexibilidad e inmediatez al amoldarse a las circunstancias del entorno. (Clase 0)
- **Modelo:** El modelo es la parte de nuestro proyecto que almacena, borra, modifica y manipula el caudal principal de los datos, apoyándose en alguna bb.dd. (Clase:18).
- **Mutabilidad**: Los diccionarios al igual que las listas son mutables, es decir, que podemos reasignar sus ítems haciendo referencia con el índice. (Clase: 6)

## N

- **Not:** es la negación o también conocida como el NO. Es un poco especial, ya que solo afecta a los tipos lógicos True y False; solo requiere un operando en una expresión. (Clase: 3)

# Material complementario

CODERHOUSE

O

- **Operadores:** son aplicaciones, cálculos que se llevan a cabo sobre dos argumentos conocidos como operandos. (Clase: 3)
- **Operadores relacionales**: son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto, la expresión es considerada verdadera (True), y en caso contrario será falsa (False). (Clase: 3)
- **Operador de igualdad:** sirve para preguntarle a nuestro programa si ambos operandos son iguales. Devolverá True si son iguales, y False si son distintos. Este operador se escribe con dos signos igual (==). (Clase: 3)
- **Operador de desigualdad:** sirve para preguntarle a nuestro programa si ambos operandos son distintos. Devolverá True si son distintos, y False si son iguales. (Clase: 3)
- **Operador menor:** sirve para preguntarle a nuestro programa si el primer operando es menor que el segundo operando. (Clase: 3)
- **Operador menor igual que:** sirve para preguntarle a nuestro programa si el primer operando es menor que el segundo operando o si ambos son iguales. (Clase: 3)
- **Operador mayor que:** sirve para preguntarle a nuestro programa si el primer operando es mayor que el segundo operando. (Clase: 3)
- **Operador mayor igual que:** sirve para preguntarle a nuestro programa si el primer operando es mayor que el segundo operando, o si ambos son iguales. (Clase: 3)
- **Operadores Lógicos:** Existen varios tipos de operadores lógicos en Python. Pero nos estaremos enfocando en los tres más básicos y utilizados: Not, Or, And. (Clase: 3)
- **Or:** operador de disyunción denominado Or en castellano O. Si el AND unía, el OR separa. Es decir, si a Python le pregunto por dos afirmaciones, y al menos una es (verdadera)True, Python me dirá que esta afirmación es True. (Clase: 3)

# Material complementario

CODERHOUSE

## P

- **Paradigma de programación:** es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. (Clase: 12)
- **Pass:** la instrucción pass permite manejar la condición sin que el bucle se vea afectado de ninguna manera; todo el código continuará leyéndose a menos que se produzca la instrucción break u otra instrucción. (Clase: 5)
- **Persistencia:** es lo que nos va a permitir guardar y recuperar los datos que se generaron en algún programa (ya sea nuestro o ajeno). (Clase: 8)
- **POO:** Es un modo o paradigma de programación, que nos permite organizar el código pensando el problema como una relación entre "cosas", denominadas objetos. (Clase: 13)
- **Pop:** La función pop retorna un elemento en forma aleatoria (no podría ser de otra manera, ya que los elementos no están ordenados). Así, el siguiente bucle imprime y remueve uno por uno los miembros de un conjunto. (Clase:6)
- **Programador:** desarrolla aplicaciones y programas informáticos, sirviéndose de las bases de un software existente para crear una interfaz para los usuarios con fines comerciales, profesionales o recreativos. (Clase 0)
- **Programación:** es el proceso con el que se crea algún tipo de aplicación o software, para materializar un concepto o proyecto que requiere de la utilización de un lenguaje informático y así poder llevarse a cabo. (Clase 0)
- **Python**: Es un lenguaje de programación de código abierto, orientado a objetos, simple y fácil de entender. Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único. Es un lenguaje interpretado con tipado dinámico, multiparadigma y multiplataforma. (Clase 0)

## R

- **Recursividad/Recursión:** es un proceso de repetición en el que algo se repite a sí

mismo. Es el efecto que sucede cuando se ponen dos espejos frente al otro. (Clase: 10)

- **Remove**: unciona igual al discard, pero con una diferencia, en discard si el ítem a remover no existe, simplemente se ignora. En remove en este caso nos indica un error. Se escribe como :mi_conjunto.remove(item _a_ remove) (Clase:6)
- **Reverse**: Esta función integrada sirve para dar vuelta una lista. Se escribe como: lista.reverse() (Clase: 7)
- **Replace**: Esta función integrada sirve para devolver una cadena reemplazando los sub caracteres indicados. Se escribe como: cadena.replace("caracter_a_remplazar", "caracter_que_reemplaza"). (Clase: 7)
- **Rfind**: s exactamente igual al método find() los diferencia en que rfind() devuelve el índice, pero de la última ocurrencia de la subcadena, es decir, la última vez que aparece en la cadena. Se escribe como: string.rfind(). Si no encuentra la cadena devuelve un -1. (Clase:7)

## S

- **Script:** es un "guión" con instrucciones de código, (básicamente, lo que venimos haciendo hasta ahora) guardado con un nombre y ejecutado desde el intérprete (ide), estos scripts pueden tomar datos (argumentos) en el momento de la ejecución. (Clase: 15)
- **Sentencia else:** entro de las sentencias condicionales el else (sino) es una especie de "hermano" de if el cual se puede encadenar al final de un bloque de código if para comprobar los casos contrarios, es decir los False. Al utilizar esta palabra reservada else le estamos indicando a Python que queremos ejecutar una porción de código, o bloque de código, sólo si no se cumple ninguna de las condiciones antes dichas, es decir, si el resultado es False siempre. (Clase: 4)
- **Sentencia elif:** La última sentencia condicional que podemos encontrar es el elif (si no, si), también podríamos decir que es un hermano de if, ya que se utiliza en

continuación al if para poder encadenar muchísimas más comprobaciones. Al utilizar esta palabra reservada elif le estamos indicando a Python que queremos ejecutar una porción de código, o bloque de código, solo si la condición anterior no se cumple, es decir, si el resultado del if o algún elif fue False.
Nos sirve para poder darle múltiples opciones al programa. (Clase: 4)

- **Sentencia For:** sentencia iterativa que podríamos decir es la más usada For (para). Se utiliza para recorrer los elementos de un objeto iterable (lista, tupla...) y ejecutar un bloque de código, o sea, tiene un número predeterminado de veces que itera. (Clase: 5)
- **Sentencia If:** dentro de las sentencias condicionales el if (si) posiblemente sea la más famosa y utilizada en la programación, esto debido a que nos permite controlar el flujo del programa y dividir la ejecución en diferentes caminos. (Clase:4)
- **Sentencia While:** Se basa en repetir un bloque de código a partir de evaluar una condición lógica, siempre que esta sea True (al igual que la sentencia if). (Clase: 5)
- **Sentencia while - else:** este else sirve para ejecutar un bloque de código cuando el bucle while tenga una condición False o haya terminado y no haya sido forzado a salir mediante un break.(Clase: 5)
- **Split:** esta función integrada sirve para devolver una lista con la cadena de caracteres separada por cada índice de la lista. Se escribe como: string.split("cadena_a_separar"). Si no se indica alguna cadena para separar separa por "espacios". (Split: 7)
- **Strip:** Esta función integrada sirve para devolver una cadena borrando todos los caracteres delante y detrás de la cadena. Se escribe como: cadena.strip("caracter_a_borrar") (Clase:7)
- **Sort:** Esta función integrada sirve para ordenar una lista automáticamente por valor, de menor a mayor. Se escribe como: lista.sort() (Clase: 7)

# Material complementario

CODERHOUSE

## T

- **Tuplas:** son unas colecciones de datos parecidas a las listas, una de las diferencias es que estas son inmutables. Se utilizan para asegurarnos que una colección determinada de datos no se pueda modificar. Python utiliza tuplas en algunas funciones para devolver resultados inmutables, por eso, conviene saber identificarlas. A su vez, dependiendo de lo que queramos hacer, las tuplas pueden ser más rápidas que las listas. (Clase 2)
- **Tipo lógico:** es el tipo de dato más básico de la información racional, y representa únicamente dos posibilidades: Verdadero ó Falso. (Clase: 3)
- **Title:** esta función integrada sirve para hacer que se devuelva la misma cadena pero con el primer carácter de cada palabra en mayúscula y el resto de caracteres hacerlos minúscula, usando el método title(). Se escribe como: string.title() (Clase: 7)

## U

- **Unión:** Esta función une un set con otro, y devuelve el resultado en un nuevo set. Se escribe como: set1.union(set2) (Clase: 7)
- **Unit test:** En castellano, Pruebas Unitarias, se refiere al método de comprobación de las “unidades” más pequeñas del software. Los componentes más pequeños que pueden probarse y cuyos resultados más significativos son los módulos. (Clase: 23)

# Material complementario

CODERHOUSE

- **Update**: Para añadir múltiples elementos a un set se usa la función update(), que puede tomar como argumento una lista, tupla, string, conjunto o cualquier objeto de tipo iterable.  La misma se escribe: mi_conjunto.update(ítem_a_agregar) (Clase: 6)
- **Upper:** sta función integrada sirve para hacer que se devuelva la misma cadena pero con sus caracteres en mayúscula, usando el método upper().  Se escribe como: string.upper() (Clase: 7)
-

V

-

W

-