

**NTER TECH SERVICES**

**Python – Ejercicios: Día 1**

Contenido

[1. Tipos básicos. Operadores 3](#_Toc168889317)

[2. Listas, Tuplas y Diccionarios 5](#_Toc168889318)

[3. Estructuras de control 6](#_Toc168889319)

# Tipos básicos. Operadores

**Ejercicio 1**

1. Explica el concepto de Tipado Dinámico en Python
2. ¿Cuáles son los principales tipos de datos en Python?
3. Describe la diferencia entre '==' e 'is'

**Ejercicio 2**

1. Define dos números de tipo complejo.
2. Muestra por pantalla la parte real del primer número definido en el punto 1
3. Muestra por pantalla la parte imaginaria del segundo número definido en el punto 1
4. Define la variable f con el resultado de la multiplicación de los dos números complejos
5. Define una variable *num* con un número entero.
6. Utiliza el operador bit a bit AND (&) para verificar si el número es par.

**Ejercicio 3: Cadenas de caracteres**

1. Define una variable con un string con el valor “ E.T., teléfono, mi casa ”
2. Muestra la longitud de la variable anterior.
3. Elimina los caracteres de espacio en blanco al comienzo y final del string
4. Muestra el primer carácter del string.
5. Muestra el penúltimo carácter del string.
6. Convierte la cadena a minúsculas
7. Busca la posición de la subcadena 'mi' en la cadena.
8. Reemplaza la palabra 'casa' por 'coche'
9. Divide la cadena en una lista de subcadenas utilizando la coma como delimitador

**Ejercicio 4: F-strings**

1. Incorpora variables en un string:
   1. Define dos variables: nombre y edad.
   2. Utiliza un f-string para mostrar una cadena que diga "Hola, mi nombre es [nombre] y tengo [edad] años."
2. Realiza operaciones aritméticas:
   1. Define dos variables: a y b con valores enteros.
   2. Utiliza un f-string para mostrar el resultado de sumar estas dos variables en una cadena que diga "Cinco más diez es [resultado]."
3. Formatea números:
   1. Define una variable pi con el valor 3.14159265.
   2. Utiliza un f-string para mostrar el valor de pi redondeado a dos decimales en una cadena que diga "El valor de pi redondeado a dos decimales es [valor\_redondeado]."
4. Incluye expresiones condicionales:
   1. Define una variable ‘score’.
   2. Utiliza un f-string con una expresión condicional para mostrar "Tu calificación es [aprobado/suspendido]" basado en si ‘score’ es mayor o igual a 50.

# Listas, Tuplas y Diccionarios

**Ejercicio 1:**

1. Explica las diferencias entre una lista y una tupla
2. ¿Cómo puedes invertir una lista en Python?
3. ¿Cuál es la diferencia entre los métodos append() y extend() para listas?

**Ejercicio 2:**

En este ejercicio, deberás eliminar los elementos duplicados de una lista, ordenar los elementos resultantes y mostrar el número de elementos únicos junto con la lista ordenada.

* Define una lista ‘nums’ con los valores [1, 5, 5, 2, 1].
* La salida del código deberá ser:
  + Número de elementos no duplicados: 3 -> [1, 2, 5]
* Crea otra lista ‘num2’ con los valores [1, 1, 2]
* Encuentra los elementos que aparecen en ambas listas y los elementos de la primera lista que no se encuentran en la segunda. Puedes convertir las listas a sets

**Ejercicio 3:**

Para realizar este ejercicio tienes un diccionario con las claves: coche, motocicleta, camión y los valores: 10, 20 y 30 respectivamente.

* Define el diccionario usando dict().
* Define el diccionario usando { }.
* Define el diccionario usando zip().
* Muestra los valores del diccionario.
* Muestra las claves del diccionario.
* Muestra el valor de coche.
* Añade al diccionario avión con valor 100.
* Muestra los elementos del diccionario.

# Estructuras de control

**Ejercicio 1:**

1. ¿Cuál es el propósito de la función *enumerate* en Python?
2. Explica el concepto de list comprehensions o comprensiones de listas

**Ejercicio 2 – Palíndromos:**

En este ejercicio, vamos a identificar palíndromos en una lista de palabras. Un palíndromo es una palabra o frase que se lee igual de adelante hacia atrás que de atrás hacia adelante, ignorando espacios y mayúsculas/minúsculas.

* Define una lista palabras con las siguientes palabras: ["Ana", "12321", "Una manzana", "radar", "No es palíndromo"].
* Convierte todas las palabras a minúsculas y elimina cualquier espacio en blanco alrededor de ellas.
* Filtra las palabras que son palíndromos, es decir, aquellas que se leen igual de adelante hacia atrás.
* Usa comprensiones de listas.
* Puedes usar *slicing* con índice negativo ([::-1]) para invertir las cadenas.

**Ejercicio 3:**

En este ejercicio, vamos a contar la frecuencia de las letras en una palabra utilizando un diccionario normal, defaultdict y Counter de la colección collections.

* Define una variable ‘palabra\_ejemplo’ con el valor "marsupilami".

1. Usando un Diccionario Normal:
   * Inicializa un diccionario vacío ‘contador1’
   * Recorre cada letra en palabra\_ejemplo.
   * Si la letra no está en el diccionario, agrégala con una frecuencia inicial de 1.
   * Si la letra ya está en el diccionario, incrementa su frecuencia en 1.
   * Imprime las frecuencias ordenadas alfabéticamente.
2. Usando defaultdict:

* Importa defaultdict del módulo collections.
* Inicializa ‘contador2’ como un defaultdict con int como valor por defecto (0).
* Recorre cada letra en palabra\_ejemplo y aumenta su frecuencia directamente.
* Imprime las frecuencias ordenadas alfabéticamente.

1. Usando Counter de collections:

* Importa Counter del módulo collections.
* Inicializa ‘contador\_palabras2’ directamente con ‘palabra\_ejemplo’.
* Imprime las frecuencias ordenadas alfabéticamente.

**Ejercicio 4:**

Deberás verificar si una cadena de paréntesis está correctamente balanceada utilizando una estructura de pila. Una cadena está correctamente balanceada si cada tipo de paréntesis se abre y cierra en el orden adecuado.

* Puedes probar con diferentes cadenas, por ejemplo, s = "{[()()]()}" o s = "{[()()]()]".
* Usa deque de collections como una pila.
* Define un diccionario ‘brackets’ para mapear los paréntesis de apertura con sus correspondientes paréntesis de cierre.

brackets = {"(": ")", "{": "}", "[": "]"}