Programación Imperativa IV

Pila

- En Python, el tipo lista provee los métodos necesarios para poder usar una lista como una pila
- También, podemos importar el módulo LifoQueue que nos da una implementación de Pila

```
from queue import LifoQueue
pila = LifoQueue()
```

- Operaciones implementadas en el tipo:
  - apilar: ingresa un elemento a la pila
    - put
  - desapilar: devuelve y quita el último elemento insertado
    - get
  - tope: devuelve (sin sacar) el ultimo elemento insertado
    - No está implementado
  - vacia: retorna verdadero si está vacía
    - empty

Cola

- En Python, el tipo lista provee los métodos necesarios para poder usar una lista como una cola
- También, podemos importar el módulo Queue que nos da una implementación de Cola

```
from queue import Queue
cola = Queue()
```

- Operaciones implementadas en el tipo:
  - encolar: ingresa un elemento a la pila
    - put
  - desencolar: saca el primer elemento insertado
    - get
  - vacia: retorna verdadero si está vacía
    - empty

Diccionarios

Un diccionario es una estructura de datos que permite almacenar y recuperar valores asociados a claves únicas. También se conoce como tabla hash o mapa. A diferencia de las listas o arreglos que se acceden mediante índices numéricos, los diccionarios permiten acceder a los elementos a través de claves.

En un diccionario, los datos se organizan en **pares clave-valor**. Cada clave debe ser única y se utiliza para acceder a su respectivo valor. Los diccionarios son eficientes para buscar y recuperar elementos, ya que utilizan una función hash interna para mapear las claves a una ubicación en la memoria donde se almacenan los valores asociados.

Diccionario

Un diccionario es una estructura de datos que permite almacenar y organizar pares clave-valor.

- Las claves deben ser inmutables (como cadenas de texto, números, etc), mientras que los valores pueden ser de cualquier tipo de dato.
- La clave actúa como un identificador único para acceder a su valor correspondiente.
- Los diccionarios son mutables, lo que significa que se pueden modificar agregando, eliminando o actualizando elementos.
- No ordenados: Los elementos dentro de un diccionario no tienen un orden específico. No se garantiza que se mantenga el orden de inserción de los elementos.

diccionario = clave1:valor2, clave2:valor2, clave3:valor3

- Operaciones basicas de un diccionario:
  - Agregar un nuevo par Clave-Valor
  - Eliminar un elemento
  - Modificar el valor de un elemento
  - Verificar si existe una clave guardada
  - Obtener todas las claves
  - Obtener todas los elementos

- El valor puede ser cualquier tipo de dato, en particular podría ser otro diccionario

```
infoPaisFrancia = {'Capital':'París',
                   'Campeonatos de Mundo':2}

infoPaisArgentina = {'Capital':'Buenos Aires',
                   'Campeonatos de Mundo':3}

infoPaisChile = {'Capital':'Santiago',
                   'Campeonatos de Mundo':0}

infoPaises = {'Chile': infoPaisChile ,
                  'Argentina': infoPaisArgentina,
                  'Francia':infoPaisFrancia}
```

Manejo de archivos

El manejo de archivos, también puede pensarse mediante la abstracción que nos brindan los TADs

- Necesitamos una operación que nos permita abrir un archivo
- Necesitamos una operación que nos permita leer sus lineas
- Necesitamos una operación que nos permita cerrar un archivo

```
# Abrir un archivo en modo lectura
archivo = open("archivo.txt", "r")

# Leer el contenido del archivo
contenido = archivo.read()
print(contenido)

# Cerrar el archivo
archivo.close()
```

```
archivo = open("PATH AL ARCHIVO", MODO, ENCODING)
```

- Algunos de los modos posibles son: escritura (w), lectura (r), texto (t - es el default)
- El encoding se refiere a como está codificado el archivo: UTF-8 o ASCII son los más frecuentes.

Operaciones básicas

- Lectura de contenido:
  - read(size): Lee y devuelve una cantidad específica de caracteres o bytes del archivo. Si no se especifica el tamaño, se lee el contenido completo.
  - readline(): Lee y devuelve la siguiente línea del archivo.
  - readlines(): Lee todas las líneas del archivo y las devuelve como una lista.
- Escritura de contenido:
  - write(texto): Escribe un texto en el archivo en la posición actual del puntero. Si el archivo ya contiene contenido, se sobrescribe.
  - writelines(lineas): Escribe una lista de líneas en el archivo. Cada línea debe terminar con un salto de línea explícito.

Problema con manejo de archivos

```
problema invertirTexto(in archivoOrigen: string, in archivoDestino: string) : {
  requiere: {El archivo nombreArchivo debe existir.}
  asegura: {Se crea un archivo llamado archivoDestino cuyo contenido será el resultado de hacer un reverse en cada una de sus filas}
  asegura: {Si el archivo archivoDestino existia, se borrará todo su contenido anterior}
}
```

API

Una API (Application Programming Interface) es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. Es una interfaz de software que define las formas en que los componentes de software deben interactuar y cómo se deben intercambiar datos.

Una API proporciona un conjunto de funciones, procedimientos, métodos y estructuras de datos que pueden ser utilizados por otros programas como una *capa de abstracción*. Esto permite que los desarrolladores utilicen funcionalidades predefinidas sin necesidad de conocer los detalles internos de implementación.

Algunos ejemplos comunes de API son:

- API web: Es utilizada para permitir la comunicación entre aplicaciones a través de internet. Estas API suelen seguir el protocolo HTTP y se utilizan para obtener y enviar datos a través de solicitudes y respuestas.
- API de bibliotecas: Son conjuntos de funciones y métodos predefinidos que proporcionan funcionalidades específicas en un lenguaje de programación determinado. Los desarrolladores pueden utilizar estas API para aprovechar las capacidades y características de una biblioteca de software sin necesidad de conocer su implementación interna.
- API de sistema operativo: Son interfaces proporcionadas por los sistemas operativos que permiten a los desarrolladores acceder a características y servicios del sistema, como el sistema de archivos, el acceso a dispositivos de hardware, la gestión de memoria, entre otros.
- API de servicios: Son interfaces que permiten a las aplicaciones interactuar con servicios externos, como servicios de almacenamiento en la nube, servicios de pago, servicios de redes sociales, entre otros. Estas API definen cómo se deben enviar y recibir datos entre la aplicación y el servicio externo.

Las API son fundamentales en el desarrollo de software, ya que permiten la reutilización de código, la integración de diferentes sistemas y la creación de aplicaciones más complejas y escalables. Al proporcionar una interfaz bien definida, las API facilitan la interacción entre las aplicaciones y promueven la interoperabilidad.

Un paso más allá: ¿Qué es una API?

Un poquito fuera del alcance de la materia...

- El término API es muy usado actualmente y está relacionado con poder usar desde un programa funcionalidades de otro programa.
- API significa *Application Programming Interface* (Interfaz de Programación de Aplicaciones, en español). Una API define cómo las distintas partes de un software deben interactuar, especificando los métodos y formatos de datos que se utilizan para el intercambio de información.
- En el contexto de desarrollo de software, una API puede ser considerada como un contrato entre dos aplicaciones.
- Una API encapsula el comportamiento de otro programa y en muchos casos, su utilización es similar al uso de un TAD. Detrás de este encapsulamiento se esconden un gran números de problemas a resolver como ser: conexiones de red, uso de protocolos, manejo de errores, transformaciones de datos, etc (y son muchos etcs).