Estructura de Datos

2 Tipos de Datos:

- 1-Datos simples
- 2-Datos estruturados

Datos Simples:

También llamados primitivos

No se pueden descomponer

El lenguaje de programación los provee por defecto

Están estrechamente asociados al procesador

Cada tipo de dato suele tener un tamaño fijo en bytes

Hay cuatro tipos tradicionalmente:

- Números enteros
  - El tamaño es importante para el procesador pues las instrucciones necesitan hacer referencia a posiciones de memoria y éstas consideran longitudes
  - Si el número no cabe en el tipo de dato el valor se desborda (se toman los primeros N bytes de acuerdo al tamaño del tipo, lo que puede dar valores que parecen extraños)
- Números flotantes
  - Se caracterizan por llevar un .
  - El procesador los manipula de forma diferente a los enteros
  - Se le suele aplicar truncado cuando hay muchos decimales (posiblemente infinitos)
  - En Python sólo hay el tipo float dinámico
- Caracteres
  - Su tamaño es variable de acuerdo a la codificación, aunque tradicionalmente en C es de 1 byte
  - En Python no existe este tipo de dato, en caso que lo necesites se pueden crear cadenas de longitud 1
- Booleanos
  - Representan valores lógicos verdadero o falso
  - En Python tenemos los valores literales True y False

==========

Datos complejos

- También llamados estructurados
- Son tipos de datos que se componen de otros tipos (primitivos u otros datos estructurados)
- A estos tipos también se les suele llamar “Estructuras de datos”
- Las estructuras de datos son centrales en la programación
- Existen estructuras de datos lineales, no lineales y jerárquicas

Lineales:

Cada elemento (excepto el primero y último) tienen un (y sólo uno) elemento predecesor y un elemento sucesor

También se les llama indexados, puesto que se pueden recuperar sus elemento a través de un índice

Tipos principales de estructuras lineales en Python:

- Listas
  - Similares a los arreglos de otros lenguajes
  - Son dinámicas (más sobre esto después)
  - Son mutables (más sobre esto después)
- Tuplas
  - Similares a las listas, pero son no mutables y estáticas
- Cadenas
  - Internamente son arreglos de caracteres
  - Son no mutables y estáticas

NO LINEALES:

No tienen un orden

No son estructuras de datos pensadas para ser recorridas (aunque se puede hacer)

En Python se tienen estos tipos principales:

Diccionarios

- En otros lenguajes son conocidos como Hash maps o Hash tables
- Cada elemento consta de dos partes: una llave y un valor asociado

- Permiten la recuperación aleatoria de elementos (se puede acceder directamente a posiciones de memoria sin necesidad de visitar otras posiciones antes)
- La llave es como el índice, pero puede ser de otros tipos además de enteros (más sobre esto en el tema del curso de diccionarios)

Conjuntos (sets)

- Se pueden entender en el sentido matemático
- Son colecciones de elementos no ordenados donde no importan las repeticiones
- Muy útiles cuando se necesitan operaciones sobre conjuntos como unión, diferencia e intersección

Objetos

- Son tipos de datos especiales de la programación orientada a objetos (POO)
- Sirven para encapsular varios valores a través de atributos
- También pueden tener comportamiento a través de métodos
- Para crear objetos antes debes definir clases

==========

Jerárquicas

Los elementos tienen un orden jerárquico, esto es pueden tener ancestros y descendientes

El tipo principal son los árboles

Los árboles se componen de nodos

==========

Representación estática y dinámica

Los datos estructurados pueden ser estáticos o dinámicos

Cuando son estáticos se debe definir su tamaño puesto que en memoria se reserva el tamaño necesario

El tamaño de las estructuras estáticas no puede ser cambiado en tiempo de ejecución (cuando el programa ya está corriendo)

En cambio, las estructuras dinámicas pueden crearse y destruirse en tiempo de ejecución, variando su tamaño a conveniencia

La ventaja de las estructuras estáticas es que son más eficientes puesto que suelen utilizar espacios de memoria contiguos (caso de los arreglos), lo que permite el acceso aleatorio a elementos

En cambio las estructuras dinámicas suelen estar dispersas en memoria, lo que hace más costoso el acceso a elementos

En Python la mayoría de estructuras de datos son dinámicas, aunque se intenta mantener espacios contiguos en memoria

Es común que en lenguajes interpretados se tengan estructuras dinámicas por defecto, mientras que en los lenguajes compilados estructuras estáticas

En lenguajes compilados también se pueden definir estructuras dinámicas usando apuntadores (de forma directa o indirecta en caso de que el lenguaje no permita manipular manualmente apuntadores)

==========

estructuras mutables

- Listas
- Diccionarios
- Objetos
- Árboles
- Sets (conjuntos

estructuras no mutables

Tuplas

Cadenas

==========

Ejecución de programas

Para ejecutar un programa desde línea de comandos es necesario saber tres cosas:

Cómo se llama el programa: esto es, el nombre del archivo principal que contiene al programa

El archivo correspondiente es ejecutable:

Windows: tiene la extensión .exe

Linux: tiene permisos de ejecución

La ruta donde se encuentra dicho archivo: la ruta es el conjunto de directorios para llegar al archivo

==========

parámetros a un programa

Es una de las formas más sencillas de pasar entradas a un programa

Muy similar a pasar parámetros a una función

No confundir con leer de entrada estándar, como lo hace la función input, los parámetros se procesan de forma diferente

En general, siempre prefiere usar parámetros en vez de entrada estándar (excepto en el sistema del curso donde sólo se pueden pasar entradas por entrada estándar)

Los parámetros permiten automatizar programas con scripts de sistema de forma más sencilla (cubierto en otro materia de la carrera)

El tipo de parámetros más simple son los posicionales (sólo se cubrirá ese tipo en el curos), aunque existen otros (modificadores, variables)

==========

Manejo de errores

En general existen 3 tipos de errores en un programa:

-Errores en tiempo de compilación/traducción

-Errores en tiempo de ejecución (excepciones)

-Errores lógicos

Son errores que arroja el traductor antes de que se pueda correr el código, osea que suceden como parte del proceso de traducción

Este tipo de errores tiene 3 categorías:

Errores léxicos:

Se refieren a no poder separar apropiadamente partes del programa (tokens)

Ejemplos puede ser empezar un identificador con un número, utilizar símbolos no soportados como @, usar guiones - en un identificador

Errores sintácticos

Suceden cuando no te apegas a la gramática del lenguaje

La gramática es un conjunto de reglas formales que establecen las formas válidas del lenguaje (sentencias y expresiones)

Errores semánticos

Son errores que suceden cuando algo no tiene sentido

Principalmente tratar de hacer cosas entre tipos que no son compatibles entre si

==========

Errores en tiempo de ejecución

También llamados excepciones

Son errores que suceden mientras el programa se ejecuta, esto es, cuando ya se convirtió en un proceso del sistema (verán procesos en sistemas operativos)

Se dan principalmente por problemas en el ambiente de ejecución o por entradas externas (como las de un usuario)

Estos problemas son impredecibles (no puedes saber si van a ocurrir o no a priori)

Este tipo de errores no pueden ser atrapados por el traductor

==========

Errores lógicos

Son el tipo de error más complicado de detectar y corregir

Son errores que no necesariamente generan excepciones (aunque podrían hacerlo), por lo que pueden ser silenciosos

Tienen que ver con que el programa no haga lo que se espera, arrojando resultados inválidos

Suelen requerir de herramientas (como los depuradores o debuggers) y técnicas de depuración para poder corregirlos

==========

Terminología

Texto plano:

Se refiere al texto que no ha sido procesado de forma especial

Cuando hablamos de texto en este curso, hacemos referencia a texto plano

Un archivo de texto plano, sólo tiene texto, no lo acompaña otros datos binarios

==========

Parsing de texto

Se refiere en general a procesar texto

Está asociado al procesamiento en modo lectura (sin modificar el texto original)

Se utiliza para lograr algunas de las siguientes cosas:

Recuperar elementos particulares en el texto

Transformar el texto entre formatos (renderizar a imagen por ejemplo)

Construir una estructura de datos que facilite el acceso a elementos: por ejemplo, construir un árbol a partir de un archivo html

Unicode:

Un problema de ASCII es que no cubre muchos de los caracteres de varios idiomas: japones, griego, árabe, etc.

Esto obligaba a tener muchos estándares diferentes, lo cual causa muchos problemas de compatibilidad

En la era de Internet este es un problema serio

Unicode es el estándar moderno que cubre todos los posibles caracteres de todos los idiomas

Actualmente tiene definidos al rededor de 140,000 caracteres con espacio para definir más si hace falta

Esto incluye cosas como emoticones

Es una extensión de ASCII, esto es, los primeros 256 caracteres son los mismos

Un problema de unicode es que ya no basta un byte para representar caracteres

En python, la función ord regresa el code point correspondiente a un carácter

Mientras que la función chr hace la operación inversa

Clases

Son plantillas para crear objetos

Varios objetos pueden ser de la misma clase

Una clase define un tipo de dato, es la forma que tienen los programadores de crear sus propios tipos

Python implementa sus tipos de datos mediante clases (muchas cosas en Python son objetos)

Son elementos estáticos, esto es, no son instancias (no están siendo administradas por el proceso)

Una práctica común es nombrar las clases con una mayúscula al inicio, así se distinguen de otro tipo de identificadores (como variables, constantes o nombres de funciones)

Atributos

También se les puede llamar propiedades

Son similares a las variables, pero sólo existen dentro del objeto

Se pueden crear desde el método especial __init__

Este método es el que se invoca implícitamente al crear el objeto (instanciarlo)

Para recuperar el valor de una propiedad a partir de un objeto, se utiliza el operador .

Una práctica común es que los parámetros de __init__ correspondan en nombre con el de las propiedades

Dentro de la clase, para hacer referencia a una propiedad se utiliza la referencia especial self

Al conjunto de valores de las propiedades de un objeto en un momento dado se le llama estado del objeto

Notar que los objetos pueden ser estáticos/dinámicos o mutables/no mutables dependiendo del tipo de sus propiedades

Dada su flexibilidad (y métodos especiales de python) los objetos pueden comportarse como lineales, no lineales o jerárquicos (dependiendo de lo que quiera el programador)

==========

Tipos principales

Listas ligadas:

Se componen de nodos, donde cada nodo tiene una referencia al nodo siguiente (como se vio en el ejercicio de clase)

Listas doblemente ligadas

También se componen de nodos, pero cada nodo tiene una referencia al nodo siguiente y al anterior, también hay una referencia al primer (cabeza) y último elemento (cola)

==========

Hay dos variantes principales de operaciones en Python:

Método sort de lista: in-place

Función sorted: no mutable, para diferentes estructuras de dat

==========

Pilas

son una estructura de datos similar a las listas

Son estructuras lineales (aunque no indexables), dinámicas pero no mutables

A diferencia de una lista, el acceso a elementos está restringido, sólo se puede tener acceso a un elemento a la vez (el tope de la pila)

En esta estructura, el último elemento que se agrega es el primero que puede salir

Esta estructura tiene tres operaciones básicas:

- Push: agrega un elemento al tope
- Pop: saca y regresa el elemento al tope
- Peek: sólo regresa el valor del elemento del tope sin sacarlo

==========

Cola

append: agrega un elemento al final (igual que en una lista)

shift: saca el elemento del frente y lo regresa

peek: sólo regresa el valor del elemento del frente sin sacarlo s

Conceptos avanzados sobre funciones

Ámbito de identificadores (scope)

Un identificador es cualquier cosa que el programador pueda nombrar en el programa: variables, nombres de funciones, nombres de clases, etc.

El ámbito o scope se refiere al área del programa donde se puede tener acceso a un identificador

Tipos:

Global

- Es el ámbito más general, en Python puede identificarse fácilmente al ver que no hay identación
- El ámbito se crea cuando inicia el programa
- Este ámbito muere hasta que termina el programa

Función/método

- Son los identificadores que se crean dentro del cuerpo de una función o método
- Esto incluye a los parámetros de la función
- Se crea un ámbito de este tipo por cada invocación a función/método
- El ámbito muere cuando la función/método termina
- Visualmente son los identificadores que están definidos dentro del cuerpo de la función/método

Clase/objeto

- Se crea un ámbito de este tipo cada vez que se crea un objeto
- El ámbito muere cuando el objeto pierde todas sus referencias (ver subtema de recolección de basura)
- Visualmente son los identificadores definidos dentro del cuerpo de una clase

Bloque

- Es un tipo de ámbito que se crea cuando hay sentencias de bloque (como las sentencias de control)
- El ámbito muere cuando termina el bloque

Tipos de parámetros en las funciones

Posicionales obligatorios

- Son el tipo de parámetros que se han estado utilizando hasta el momento
- Dependen de un orden, por lo que se les llama posicionales
- En Python es obligatorio pasar todos los parámetros posicionales al invocar la función

Opcionales

- Como su nombre lo indica son opcionales, esto es, se pueden pasar o no al invocar la función
- Son posicionales en el sentido de que se corresponden con la posición de acuerdo a como

Keyword (nombrados)

- Son parámetros no posicionales, esto es, los puedes pasar en cualquier orden
- Se declaran igual que los parámetros opcionales, esto es, en Python, cualquier parámetro definido como opcional es también keyword (esto no es así en todos los lenguajes)
- Al igual que los opcionales, se deben definir después de los parámetros posicionales obligatorios
- Al invocar la función, se puede pasar directamente la relación de valores del parámetro utilizando su nombre
- A las estructuras asociativas se les llama de diversas formas: mapas de hash, diccionarios, tablas de hash

Los elementos de una estructura asociativa se componen de dos partes:

- Llave: es el índice con el que se puede recuperar el valor
- Valor: es el valor en si que se desea almacenar

Funcionamiento interno

- Un hash-map se implementa tradicionalmente a partir de un arreglo interno (que puede ser de dos dimensiones, a lo que se le llama una tabla)
- Es en el arreglo interno donde en realidad se guardan los elementos
- El arreglo interno permite que el accesos a los elementos sea aleatorio
- La idea es convertir la llave de un elemento a un índice numérico del arreglo interno
- Esta conversión se logra mediante una función de hash
- Con el índice numérico se recupera directamente el valor

Funciones de hash

- A este tipo de funciones también se les llama de resumen
- Es una función que recibe algún tipo de objeto (para nuestro caso el tipo de la llave) y regresa un valor en un rango predefinido (normalmente un número)
  - Como el dominio de la función es un conjunto infinito pero el codominio es un conjunto finito es obvio que varios (de hecho un número infinito) de los valores del dominio se mapearan con el mismo valor del codominio. Al hecho anterior se le llama Colisión
  - Entre más grande sea la aridad del codominio menor será la probabilidad de que dos elementos colisionen entre si

En una estructura asociativa deben definirse:

- Una función hash para convertir de llave a índice
- Una política de manejo de colisiones

Una política común es encadenamiento, esto es, se usa una lista ligada de los valores que colisionaron en la misma posición (de esta forma el arreglo interno guarda realmente listas ligadas)

La recursividad

En términos simples una función recursiva es aquella que se define en términos de ella misma, esto es, en algún punto de su propio cuerpo se manda a llamar a si misma (también puede ser de forma indirecta como se menciona más adelante)

A muchos programadores se les dificulta el uso de recursividad, incluso a programadores profesionales

Esto se debe a su naturaleza declarativa, siendo que la programación se enseña normalmente de forma imperativa

Programación imperativa

- Se refiere a darle a la computadora paso por paso las instrucciones que debe ejecutar
- Es trabajo del programador dar todos los detalles del control del programa
- Esto es, el programador debe definir cómo resolver el problema

Programación declarativa

- El programador se preocupa principalmente por definir el problema
- Esto es, el programador define qué es el problema
- Existe un mecanismo de resolución automático que recibe definiciones de problemas y genera soluciones
- Este mecanismo puede ser por ejemplo un motor de resolución lógica (usado en programación lógica)

Manejo de memoria

Memoria RAM

- La memoria RAM es una memoria de acceso rápido (a diferencia del disco duro)
- Básicamente se utiliza para tener acceso a los datos de los programas que se encuentran en ejecución (llamados procesos)
- Todo programa en ejecución está cargado en la RAM (al menos parcialmente si no hay suficiente memoria)
- La memoria RAM puede verse como una matriz de celdas, cada celda guarda un byte
- Cada celda tiene una dirección de memoria definida lo que permite que el acceso a la celda sea aleatorio (no es necesario recorrer otras celdas para encontrar la celda de interés)

Espacios de memoria de un proceso

- Al ejecutarse un proceso, el sistema operativo reserva espacio en memoria para guardar los datos del proceso

Lo procesos tienen 4 áreas de memoria principales (puede variar entre sistema operativos):

- Call stack (pila de llamadas)
- Heap
- Datos (global)
- Código

Call stack

- También llamado simplemente stack
- Es una zona de memoria que se comporta como una pila
- Su función principal es recordar el orden en que se hacen llamadas anidadas entre
- Esta zona de memoria es la que permite que exista recursividad (explicado más adelante)

Heap

- Es un área especial del proceso designada para el uso de los programadores
- En algunos lenguajes como C, el programador puede reservar y borrar memoria en esta área
- El sistema operativo no limpia esta memoria (como si lo hace con el stack)
- En lenguajes de más alto nivel como Python el programador no toma decisiones sobre dónde colocar datos, se utiliza la opción más robusta de forma transparente

Datos

- Es un espacio especial para almacenar variables globales yconstantes
- Es un espacio fijo que se designa en tiempo de compilación (en lenguajes compilados)

Código

- En este espacio se almacena el código del programa asociado al proceso
- De esta forma se puede llevar un control para que el procesador sepa que instrucción debe ejecutar a continuación
- Es importante recordar en qué punto se encuentra la ejecución dado que los procesos entran y salen constantemente del procesador (el calendarizador del SO se encarga de esta función)

Funciones recursivas

Como ya se mencionó, una función recursiva es aquella que se define en términos de ella misma

Las funciones recursivas tienen casos base y casos recursivos

Un caso base es una entrada de la función para la cual se puede entregar directamente un valor

Un caso recursivo es aquel que hace referencia a la propia función (hace una llamada a la propia función)

Cómo funcionan las funciones recursivas

- Cada vez que se hace una llamada recursiva se apila la llamada en el stack, quedando pendiente la finalización de la función
- Al alcanzar un caso base se empieza a desapilar propagando resultados en la pila
- Dado que se trabaja con una pila (como se vio en el tema de pilas) las cosas suceden en el orden inverso después de la llamada recursiva

Diseño de soluciones recursivas

- Un error que cometen muchos programadores al crear funciones recursivas es pensar constantemente en lo que pasa en el stack
- NO hagas eso: es importante saber cómo funcionan las llamadas recursivas, pero pensar en ello aumenta la carga cognitiva y la complejidad
- En vez de eso debes pensar en los casos, esto es programación declarativa, no hay que pensar en el proceso detallado, sólo en definir el problema
- El manejo de la pila de llamadas es tu mecanismo de resolución automático, no es necesario pensar en cómo funciona

Recursividad directa

- Es la forma que se ha visto hasta el momento en el tema
- Se refiere a que en el cuerpo de la función se mande a llamar a la propia función

Recursividad indirecta

- Sucede cuando entre dos o más funciones se hacen llamadas entre si
- Es menos común y útil que la recursividad directa

Tipos

- En este curso se verán dos tipos de árboles:
  - Árboles generales
  - Árboles binarios de búsqueda

Árboles generales

- Cada nodo puede tener cualquier número de hijos
- No cumplen ninguna propiedad en especial

Árboles binarios de búsqueda

- Cada nodo puede tener a lo mucho dos hijos
- Están pensados para indexación rápida de datos
- Por ejemplo, se utilizan en bases de datos para recuperar de forma eficiente registros
- Tienen restricciones:
  - Cada nodo tiene un índice numérico
    - El hijo del lado izquierdo tiene un valor de índice menor al padre
    - El hijo del lado derecho tiene un valor de índice mayor al padre

Implementación

Árbol general

- Python no cuenta en su biblioteca estándar con esta estructura de datos
- Se verá una implementación basada en objetos
- La implementación se divide en dos clases:
  - Una clase que represente cualquier nodo del árbol
    - Se tiene una propiedad con los nodos hijos
  - Una clase que representa al árbol en si

- Se tendrá una propiedad con una referencia al nodo raíz
- Durante el tema se realizará una implementación gradual en clase

Operaciones de árbol

- Crear árbol
- Agregar nodo
- Recuperar valor
- Imprimir árbol
- Borrar nodo

Rutas de archivo

- En el sistema todos los archivos se almacenan en alguna ruta lógica
- Esa ruta depende de una estructura de árbol
- El sistema de archivos se estructura mediante un árbol, donde cada directorio representa un nodo

Rutas absolutas

- Si la ruta se da desde la raíz

Rutas relativas

- No empieza con diagonal
- Es relativa al directorio de trabajo actual
- Rutas especiales relativas:
  - . referencia a la ruta actual
  - . referencia al directorio padre

Abrir un archivo existente para lectura

- Para abrir archivos se utiliza la función open
- A esta función se le pasan principalmente dos parámetros:
  - Ruta del archivo
  - Modo de uso del archivo: si es de texto, binario, lectura, escritura o append