Universidad Nacional de Colombia
Facultad de Ciencias Exactas y Naturales
Sede Manizales

Relatoría Tercer Corte

Informática III

Sofia Londoño Toro

20 de noviembre de 2022

Resumen

Mediante la presente relatoría se pretende mostrar los avances y aprendizajes del tercer corte de la asignatura Informática III, donde se aprendió un nuevo paradigma de programación basado en la implementación de módulos y paquetes, sean integrados o creados por nosotros mismos, también se enfatizó en la manipulación y análisis básico de datos mediante el uso de paquetes integrados de python como Numpy, Pandas y Matplotlib.

Tema 9 (12/10/2022 - 26/10/2022) : MÓDULOS

Para facilitar la programación modular, el desarrollo de un programa, requiere organizar las funciones en diferentes archivos (módulos), esto permite organizar un programa conforme a las necesidades, por ejemplo: Si se requiere hacer un módulo para la interfaz, otro módulo para la lógica y otro para el almacenamiento de datos. Puedo crear 3 módulos (archivos) en los cuales organizo mis funciones.

EJEMPLO: ¿Cómo crear mi primer módulo?

Simplemente se crea un archivo y se le asigna un nombre, que será el nombre del módulo. En este archivo solo se crearan las funciones o variables de interés.

EJEMPLO: Crear dos módulos uno llamado interfaz.py y otro llamado logica.py luego un archivo principal main.py donde pueda ejecutar las funciones de los módulos anteriores

- Con la opción nombredelmodulo.__doc__ muestra la documentación asociada al módulo
- Con la opción nombredelmodulo.__dic__ muestra todas las funciones integradas del módulo, sus especificaciones y funciones heredadas.
- Para hacer el testeo de un módulo:

```
if __name__ == "__main__":
# Cuando ejecute el módulo en main, estas lineas no ocurriran
*Acá se ejecuta la función y lo que se quiera probar
```

MÓDULOS INTEGRADOS

Existen infinidad de modulos pre-instalados que podemos utilizar ahorrando gran cantidad de trabajo.

Ejemplos de modulos utiles:

- random: Generador de numeros aleatorios, con sus respectivas distribuciones estadisticas
- math: Provee funciones matematicas utiles. y constantes numericas, pi, euler.
- tqdm: Herramienta para que el programador estime el tiempo de ejecucion de un programa.
- pandas: Permite hacer hojas de calculo faciles de manipular semejantes a excel y estructuras sql.

- skitlearn: Funciones y herramientas útiles hacer machine learning.
- matplotlib: Generador de distintos tipos de graficas, basado en matlab.
- numpy: Provee estructuras de datos potentes, que facilitan su manipulación.
- os: Para realizar operaciones en el terminal desde python.
- sys: Organiza la ubicacion de las herramientas, modulos funciones, o paquetes de python

Para usar estas librerias o módulos integrados de manera local en el pc, se deben instalar desde el terminal o cdm del computador, ejecutando el comando: `pip instal *Nombre de la libreria que se quiera descargar*`

Para acceder a un módulo en un archivo se deben de importar de la siguiente manera `import *módulo a usar* as *Abreviación del nombre*`, las abreviaciones sirven para utilizar la libreria de manera más rápida, pues cuando se quiere llamar una función de dicho módulo se debe escribir el nombre del módulo o dicha abreviación seguido de un punto y la función a llamar. Existen algunas abreviaciones comunes para las librerias como, np para numpy, pd para pandas, plt para matplotlib, entre otras. Ejemplo:

```
import numpy as np # Importamos la libreria
np.array([lista]) # Usamos la abreviación para usar la función integrada de numpy
'array'
```

También es posible solo importar ciertas funciones de un módulo, esto optimiza el código y hace más sencilla su escritura, pues en este caso no es necesario llamar la función utilizando el nombre del módulo:

```
from numpy import array # Solo importamos la función array
array([]) # Usamos la función
```

Tema 10 (26/10/2022 - 09/11/2022) : MANEJO DE LIBRERIAS PARA ESTRUCTURAS DE DATOS (NUMPY, PANDAS)

Estos dos módulos son los más usados para estructurar y analizar datos, las estructuras de datos más usadas son los arrays de Numpy y las Series y DataFrames de Pandas.

Numpy

Los arrays de numpy son estructuras que dan soporte a los iterables de python. Poseen mayor cantidad funcionalidades, están optimizadas y son de gran utilidad para el calculo numerico. Mediante los arreglos, es posible crear vectores, matrices y tensores de manera eficiente.

- En una dimensión:

```
NombreArreglo1D = [3,4,5,6]
Nombre[1 (columna)] = 5
```

- En dos dimensiones:

NombreArreglo2D = [[1, 2],[3, 4]]
NombreArreglo2D[1 (fila), 0 (columna)] = 3

- En tres dimensiones:

```
NombreArreglo3D = [[[1, 2], [3, 4]],[[5, 6],[ 7, 8]],[[9, 10], [11, 12]]]
NombreArreglo3D[1 (profundidad), 1 (columna), 0 (fila)] = 4
```

Funciones integradas:

- Creación de arreglos:
  - `np.array(<iterable>)`
  - `np.arange(<inicio>,<fin>,<salto>) # Crea arreglos en un rango de números, con sus números separados en un salto determinado`
  - `np.ones((<filas>,<columnas>)) # Crea arreglos solo de unos`
  - `np.zeros(<filas>, <columnas>) # Crea arreglos solo de ceros`
- Redimensionamiento:
  - `np.reshape((<filas>, <columnas>)) # Sirve para reorganizar los datos de un arreglo, pasar de vector a matriz, etc..`
- Apilamiento:
  - `np.hstack((<elemento1>, <elemento2>)) # Apila vectores, matrices o tensores, al lado derecho del elemento 1, creando un arreglo más ancho`
  - `np.vstack((<elemento1>, <elemento2>)) # Apila vectores, matrices o tensores, debajo del elemento 1, creando un arreglo más largo`

  Métodos para los arreglos:
  - Valor mínimo: `arreglo.min()`
  - Valor máximo: `arreglo.max()`
  - Promedio: `arreglo.mean()`
  - Desviación estándar:`arreglo.std()`
  - Suma del arreglo: `arreglo.sum()`

  Numpy también tiene funciones mátematicas integradas, como:
  - `np.sin(<arreglo>) # trigonometricas`
  - `np.cos(<arreglo>) # trigonometricas`
  - `np.exp(<arreglo>) # trigonometricas`
  - `np.log(<arreglo>) # Logaritmo`
  - `np.gcd(<arreglo>) # Maximo comun divisor`
  - `np.lcm(<arreglo>) # Minimo comun multiplo`
  - `np.dot((<arreglo1>, <arreglo2>)) # Producto punto`
  - `np.linalg.det(<arreglo>) # Determinante`

- `np.linalg.inv(<arreglo>) # Inversa`
- `np.linalg.norm(<arreglo>) # Norma`
- `np.linalg.solve(<a>, <b>) # Resolver sistema de ecuaciones`

Pandas

Series:
Es una estructura unidimensional eficiente. Se construye sobre un arreglo numpy. Está compuesto por indices numericos =>0,1,2,3... indices claves =>'Enero", "Febrero",...

Además se pueden indexar, opcionalmente utilizando ya sea indices numericos o indices claves, de la siguiente manera:

```
serie[0] # Valor en el indice 0, primer valor
serie['Enero"] # Valor con clave Enero
```

Para crear una serie:

```
import pandas as pd

serie = pd.Series(data = <arregloNumpy>, index = [....]) # El arreglo es la
serie como tal el index es la clave de cada dato de la serie.
# Ejemplo:
serie = pd.Series(data = np.Array([5,4,3.5]), index = ['Estu 1", 'Estu 2",
'Estu 3"])
```

Las series tienen algunos atributos útiles para conocer su contenido:

- `serie.index : retorna los indices clave`
- `serie.values : retora la data`
- `serie.shape : retorna el tamaño`

También tienen métodos muy útiles para analizar datos:

- `serie.describe() : devuelve otra serie, describiendo la serie original`
- `serie.mean() : devuelve la media`
- `serie.std() : devuelve la desviación estándar`
- `serie.min() : devuelve el valor mínimo`
- `serie.max() : devuelve el valor máximo`
- `serie.idxmin() : devuelve la clave del mínimo`
- `serie.idxmax() : devuelve la clave del máximo`
- `serie.value_counts() : devuelve las frecuencias de la serie`
- `serie.str.contain() : Devuelve los valores de la serie que contengan el str dado`

DataFrames:
Son estructuras de datos en 2-D. (Filas y columnas) De manera semejante a los archivos excel, sql, csv.

- funcionan como las hojas de excel
- Para indexar trabajan igual como en las listas y diccionarios. Indexado por indice numérico o por indice clave

- `DataFrame[*Nombre de la columna*] # Accede a toda una columna, llamada por la clave respectiva.`
- `DataFrame.loc[*Indice clave*] # Accede a toda una fila, llamada por el indice respectivo a esta.`
- `DataFrame.loc[*Columna*, *Fila*] # Accede a un valor especifico buscándolo por las respectivas claves para dicho valor`
- `DataFrame.iloc[*Filas*,*Columnas*] # A un pedazo del dataFrame mediante indices números.`

Para crear un DataFrame:

```
import pandas as pd

hoja1 = pd.DataFrames(data = <arreglo 2D numpy>, columns = [çolumna1", ...., çolumnaN"], index = ["fila1",..., "filaN"]) # data representan los datos que pertenecerán al DataFrame, columns son las claves para las columnas asi como index son las claves para las filas.
```

Algunos de los métodos de los dataFrames son:

- `DataFrame.columns : Devuelve el nombre de las claves de las columnas`
- `DataFrame.index : Devuelve el nombre de las claves de los indices`
- `DataFrame.shape : Muestra el tamaño del dataFrame`

Funciones integradas:

- `DataFrame.mean(axis) # Calcula la media`
- `DataFrame.median(axis) # Calcula la mediana`
- `DataFrame.std(axis) # Calcula la desviación estándar`
- `DataFrame.min(axis) # Calcula el mínimo`
- `DataFrame.max(axis) # Calcula el máximo`

Estas funciones pueden usarse para todo el dataframe o pueden usarse para un pedazo de este, fila o columna, utilizando el indexado.

Cabe recalcar que las filas y columnas de un DataFrame son Series, por lo que se pueden aplicar las funciones de estas.

Tema 11 (11/11/2022 - 16/11/2022) : MANIPULACIÓN DE DATOS

Para poder manipular los datos, se utilizarán herramientas presentes en los arreglos, series, dataframes ya vistos. En el proceso de conocer y manipular datos es importante determinar algunos valores estadísticos, anteriormente descritos:

La media, mediana, desviacion, valor min/max ...

Además es importante visualizar la información, para ello utilizamos matplotlib de la siguiente manera:

```
import matplotlib.pyplot as plt
```

```
plt.figure() # Se crea el lienzo plt.plot(x, y, style) # donde x, y contienen la data, plt.show() # style: el estilo y color de linea
```

Matplotlib.pyplot cuenta con algunas funciones para mejorar el aspecto visual de las gráficas como:

- `plt.grid(True) # Le pone grilla a la gráfica`
- `plt.title(*Nombre*) # Le pone título a la gráfica`
- `plt.subplot(*cantidad de filas*,*cantidad de columnas*, *ubicación*) # Crea un lienzo dividido en filas y columnas, las ubicaciones comienzan a contarse desde arriba a la izquierda con el 1 y siguen de manera horizontal`

=En el análisis de datos, la derivada y la integral son valores importante para ciertos propósitos, por los tanto debe calcularse de manera numérica.

Para la derivada el método más común son las diferencias finitas, con la función integrada de numpy diff, se pueden calcular facilmente:

```
from numpy import diff
```

```
derivada = diff(y) / diff(x) # calculamos una derivada de una función f(x)
```

Para la integral se usa el método del trapecio, se calculan áreas, esto también se puede realizar fácilmente con la función integrada de numpy trapz:

```
import numpy as np
```

```
integral = np.trapz(y,x) # Calcula la integral de una función f(x)
```

CÓDIGOS

Para ver códigos de ejemplos acceder al siguiente link:
https://github.com/solondonotor/Inform-tica-III.git