Python para el análisis de datos- Lectura 3

Ing. Pedro Rotta

Universidad de Piura - Vida Universitaria

Enero-2022

Condicional

Normalmente en nuestro día a día cuando tenemos un problema siempre tenemos la posibilidad de elegir entre 2 o más opciones de solución. De acuerdo al camino que se tome, uno o más efectos sucederán.

Proposiciones lógicas

En el mundo de la programación sucede lo mismo. Un *algoritmo* es una sucesión de comandos que permite solucionar un problema. Dicho problema puede tener **propociones lógicas** para el usuario o que se verifican en el flujo del mismo.

Proposiciones lógicas

En el mundo de la programación sucede lo mismo. Un *algoritmo* es una sucesión de comandos que permite solucionar un problema. Dicho problema puede tener **propociones lógicas** para el usuario o que se verifican en el flujo del mismo.

Existen 5 condiciones comparativas que se cumplen en los lenguajes de programación y 3 conjuciones lógicas que unen a estas condiciones.

Proposiciones lógicas

En el mundo de la programación sucede lo mismo. Un *algoritmo* es una sucesión de comandos que permite solucionar un problema. Dicho problema puede tener **propociones lógicas** para el usuario o que se verifican en el flujo del mismo.

Existen 5 condiciones comparativas que se cumplen en los lenguajes de programación y 3 conjuciones lógicas que unen a estas condiciones.

x<y	x es menor que y
x>y	x es mayor que y
x==y	x es igual a y
x<=y	x es menor o igual que y
x>=y	x es mayor o igual que y

Table: "Condiciones comparativas"

Conjunciones lógicas

a **and** b	tiene que cumplirse a *y* b
a **or** b	tiene que cumplirse a *o* b
not(a)	*no* es a

Table: "Conjunciones lógicas"

Variables booleanas

Sirven para identificar el valor de verdad de una proposición lógica y son:

- **True** o 1
- **False** o 0

Condicional if

if es el comando para verificar si se cumple una condición lógica interna o externamente. Por ejemplo analicemos el primer problema del colab.

Condicional if

if es el comando para verificar si se cumple una condición lógica interna o externamente. Por ejemplo analicemos el primer problema del colab.

Aclaración importante: La indexación del código es importante, se puede observar que lo que cumple la condición está indexado dentro de la proposición lógica. *if*.

else y elif

Normalmente existen 2 o más caminos que seguir en un algoritmo, por lo que con frecuencia nos encontraremos situaciones que necesiten más de una proposición lógica para ser evaluados.

else y elif

Normalmente existen 2 o más caminos que seguir en un algoritmo, por lo que con frecuencia nos encontraremos situaciones que necesiten más de una proposición lógica para ser evaluados.

En estos casos se usa la condición **else** que significa *sino* y la condición **elif** que significa *sino,si...*. Para entenderlo veamos el 2do problema de Colab.

If Nested

Se pueden colocar condiciones dentro de una condición, es decir si se cumple una condición, aún pueden haber más caminos que cumplir. Esto se logra usando **condicionales indexados**.

Colecciones

Cuando tenemos muchas variables(objetos únicos) vamos a tener que *coleccionarlos*. Por ejemplo varios nombres de alumnos, varios emails, colecciones de notas, datos varios.

Las formas comunes para guardar colecciones de datos son:

- listas
- diccionarios
- tuplas
- conjuntos

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.
- El primer elemento de las listas es el elemento 0, a este valor se le conoce como **index**.

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.
- El primer elemento de las listas es el elemento 0, a este valor se le conoce como **index**.
- Para seleccionar un elemento de la lista se debe colocar la sintaxis *nombrelista[index]*

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.
- El primer elemento de las listas es el elemento 0, a este valor se le conoce como **index**.
- Para seleccionar un elemento de la lista se debe colocar la sintaxis *nombrelista[index]*
- Para seleccionar un **rango** de valores de la lista, escribimos *nombrelista[index1:index(n-1)]*

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.
- El primer elemento de las listas es el elemento 0, a este valor se le conoce como **index**.
- Para seleccionar un elemento de la lista se debe colocar la sintaxis *nombrelista[index]*
- Para seleccionar un **rango** de valores de la lista, escribimos *nombrelista[index1:index(n-1)]*
- Para agregar un elemento a una lista colocamos *nombrelista.append(elemento)*

Listas

- Las colecciones de *tipo* lista comienzan con corchetes, cada elemento se separa por comas.
- El primer elemento de las listas es el elemento 0, a este valor se le conoce como **index**.
- Para seleccionar un elemento de la lista se debe colocar la sintaxis *nombrelista[index]*
- Para seleccionar un **rango** de valores de la lista, escribimos *nombrelista[index1:index(n-1)]*
- Para agregar un elemento a una lista colocamos *nombrelista.append(elemento)*
- Para eliminar un elemento de una lista colocamos *nombrelista.pop(index)*

Veamos el problema 4 de colab

Diccionarios

Se usan para guardar datos que tienen varios atributos.Normalmente son una forma de expresar bases de datos(BD).

Diccionarios

Se usan para guardar datos que tienen varios atributos.Normalmente son una forma de expresar bases de datos(BD).

En el 5to problema de Colab, se estudia la sintaxis de un diccionario. En este caso para expresar un conjunto de usuarios. Cada usuario tiene nombre, email y celular, además tiene peliculas favoritas. Dentro de las películas favoritas hay algunos que tienen 2 películas, otros 3 películas.

Diccionarios

Se usan para guardar datos que tienen varios atributos.Normalmente son una forma de expresar bases de datos(BD).

En el 5to problema de Colab, se estudia la sintaxis de un diccionario. En este caso para expresar un conjunto de usuarios. Cada usuario tiene nombre, email y celular, además tiene peliculas favoritas. Dentro de las películas favoritas hay algunos que tienen 2 películas, otros 3 películas.

Los diccionarios se usan para organizar BD no relacionales. Hablaremos más adelante mucho más acerca de ellos.

Tuplas

- Las tuplas, tal como las listas son conjuntos de datos desordenados.

Tuplas

- Las tuplas, tal como las listas son conjuntos de datos desordenados.
- Sintácticamente las tuplas comienzan con paréntesis y sus elementos van separados por comas.

Tuplas

- Las tuplas, tal como las listas son conjuntos de datos desordenados.
- Sintácticamente las tuplas comienzan con paréntesis y sus elementos van separados por comas.
- La diferencia con las listas, es que las tuplas no aceptan ni agregar ni eliminar elementos.

Tuplas

- Las tuplas, tal como las listas son conjuntos de datos desordenados.
- Sintácticamente las tuplas comienzan con paréntesis y sus elementos van separados por comas.
- La diferencia con las listas, es que las tuplas no aceptan ni agregar ni eliminar elementos.
- En el problema 6 vemos cómo se pueden usar las tuplas.

Conjuntos

- Los conjuntos son colecciones de datos ordenadas.
- Sintácticamente se escriben entre corchetes sus elementos se separan por comas.
- Sirven para ordenar datos y además eliminar datos repetidos

Método len() y conversiones

Para saber el número de elementos de cualquier colección existe la función len.

Método len() y conversiones

Para saber el número de elementos de cualquier colección existe la función len.

La sintaxis es *len(colección)*

Método len() y conversiones

Para saber el número de elementos de cualquier colección existe la función len.

La sintaxis es *len(colección)*

Las conversiones de colección también son importantes. Para cambiar una colección la sintaxis es:
coleccion(coleccion a cambiar).

Podemos revisar el problema 14.

Métodos

Las colecciones también son objetos dentro de Python y así como las personas tenemos ciertas funciones propias de nosotros, como por ejemplo comer, los objetos de programación también tienen funciones propias.

Métodos

Las colecciones también son objetos dentro de Python y así como las personas tenemos ciertas funciones propias de nosotros, como por ejemplo comer, los objetos de programación también tienen funciones propias.

A estas funciones que realizan o que se realizan sobre los objetos en python se les llama *métodos*.

Métodos

Las colecciones también son objetos dentro de Python y así como las personas tenemos ciertas funciones propias de nosotros, como por ejemplo comer, los objetos de programación también tienen funciones propias.

A estas funciones que realizan o que se realizan sobre los objetos en python se les llama *métodos*.

Para entrar a los métodos de las colecciones podemos escribir la sgte sintaxis: *colección.método*

Puede revisar la sintaxis y practicar lo aprendido en los siguientes problemas del Colab.