MAT-281 APLICACIONES DE LA MATEMÁTICA EN INGENIERÍA

Ayudantía 1: Algoritmos e Introducción a Python

Alberto Rubio S.

October 25, 2015

Universidad Técnica Federico Santa María

Objetivos

1. Algoritmos: Definición, Máquinas de Turing y Recomendaciones.
2. Introducción a Python: Tipos de datos, Operaciones básicas, Estructuras condicionales, Funciones.

ALGORITMOS

Algoritmos

¿Qué es un algoritmo?

- Lo relacionamos con computación.
- Cuando se diseña un programa buscamos crear y describir procedimientos sin lugar a duda, sin ambigüedad.

Definición: Un algoritmo es un procedimiento bien definido y estructurado para resolver un problema dado. Se compone de:

1. Datos de entrada.
2. Proceso o pasos a seguir utilizando la entrada.
3. Salida o resultado.

Máquina de Turing

- Una máquina de Turing es un dispositivo que manipula símbolos sobre una cinta en base a reglas determinadas.
- La idea anterior puede simular la lógica de cualquier algoritmo computacional.
- Ver más en Teoría de autómatas.

Tesis de Church-Turing: Todo algoritmo es equivalente a una máquina de Turing. (Ver más en teoría de la computabilidad).

Recomendaciones

- Usar Ubuntu: Cercano a usuarios Windows, permite convivir con sistema Windows (particionando el disco), rápido y eficiente.
- Para desarrollo de programas usar Geany.

PYTHON I: TIPOS DE DATOS

Tipos de datos

Es importante puesto que determinan: dominio, operaciones posibles, representación interna. En Python existen esencialmente:

- Números enteros (**int**)
- Números reales (**float**): El nombre hace referencia a la aritmética de punto flotante. Son la mejor aproximación.
- Números complejos (**complex**): Siguen la sintáxis a+b**j**, donde **j** denota $\sqrt{-1}$.
- Valores lógicos (**bool**): Datos que sólo pueden ser **True** o **False**.
- Texto (**str**): También llamados **Strings** son cadenas de caractéres.

PYTHON II:OPERADORES BÁSICOS

Operadores aritméticos

En Python podemos aplicar las operaciones básicas a todo tipo de datos.

1. Suma: +
2. Resta: -
3. Multiplicación: $*$
4. División: \
5. Potencias: $**$
6. Módulo: %

Operadores relacionales

Sirven para comparar valores de variables y tienen como resultados valores lógicos. Algunos son:

- Signo igual: **==**
- Signo distinto: **!=**
- Signo mayor: **>**
- Signo mayor o igual: **=>**
- Signo menor: **<**
- Signo menor o igual: **<=**

Operadores lógicos

Son tanto de operando como de resultado lógico. Existen tres:

1. Conjunción lógica **and**
2. Disyunción lógica **or**
3. Negación lógica **not**

Ejercicio

Elabore un programa que calcule lo que debemos pagar en una visita a un restaurant incluyendo IVA y propina.

Solución

Un poco más sobre strings

- Los apóstrofes deben ser usados mediante la sintáxis \', en caso contrario cortan la sintaxis.
- Cada letra en un string posee una ubicación espacial de izquierda a derecha comenzando desde 0. Para acceder a ella se utiliza **[i]**, donde i denota la posición.
- Operaciones con string:
  1. **\len()**: Entrega el largo de un string.
  2. **\lower()**: Escribe todo un string en minúscula. Usa la sintaxis: var.lower()
  3. **\upper()**: Escribe todo un string en mayúscula. Usa la sintaxis: var.upper()
  4. **\str()**: Transfroma en string variables que no lo son.
  5. **Concatenar string**: Se hace mediante el signo +.

PYTHON III: CICLOS Y CONDICIONALES

Condicional if

La sentencia de **if** ejecuta una instrucción si, y solo si, se cumple una determinada condicion. Su sintaxis es la siguiente:

if condición:
    que hacer si se cumple.

Note que dicho condicional no especifica que hacer en caso que la condición **no se cumpla**.

Condicional if-else

A diferencia de la anterior también especifica que hacer en caso que la condición sea falsa. Su sintaxis es la siguiente:

```
if condición:
        que hacer si se cumple.
else:
        que hacer en caso falso.
```

Condicional if-elif-else

Permite evaluar dos o más condiciones de forma progresiva. La sintaxis es la siguiente:

if condición:
    que hacer si se cumple.
elif:
    que hacer si se cumple el 2do. condicional. **else**:
    que hacer en caso que ninguno de los dos se cumpla.

Ciclo while

Los ciclos permiten repetir sucesivamente una determinada operación. El ciclo **while** repite la operación hasta que se cumpla una cierta condición, en general de tolerancia. La sintaxis es:

while condición:
    que hacer cuando es cierto.

Ciclo for

El ciclo **for** repite sucesivamente una sentencia un número determinado de veces. La sintaxis es:

for i in range(N de veces):
    que hacer para cada valor de i

NOTA: El comando **range** entrega una sucesión de números enteros equiespaciados. Existe la variación **range**(N. inicial, N. final, incremento).

Preguntas?