Introducción a python

Luis, Gutierrez cinema.nightmare@gmail.com
Carlos Rodríguez carlosrdz.isd@gmail.com

February 29, 2020

Outline

Constantes y variables

Constantes y variables

- Una constante es un valor que se mantiene fijo durante todo el programa. Ejemplos: 5, 'a', 38, "www.google.com"
- Una variable es un espacio de almacenamiento en memoria que contiene cierta información, la cuál puede ser modificada durante la ejecución del programa. Ejemplos: variable, a, a5 = 32, var = "hola"

Tipos de datos

Tipos de datos

- Entero (integer)
- Flotante (float)
- Número complejo
- Booleano (boolean)
- String
- Lista
- Tupla
- Diccionario

Operadores aritméticos

Operadores aritméticos

- + (suma)
- - (resta)
- * (multiplicación)
- / (división)
- % (módulo)
- ** (exponente)
- // función suelo

Operadores lógicos

Operadores lógicos

- >(mayor)
- <(menor)
- == (igual a)
- != (diferente de)
- >= (mayor o igual)
- <= (menor o igual)

Operadores lógicos

- and (y)
- or (o)
- not
- y otros...

Outline

Hola python!

Hola python!

- # En este programa desplegamos un saludo
- print(”Hola grupo de python!”)

Comentarios

Comentarios

- Los comentarios sirven para hacer más entendible el código, tanto para nosotros como para los demás
- Python sabe que es un comentario al escribir el hashtag al inicio de la línea
- # Así por ejemplo
- Otra forma de escribir un comentario es con triple comilla doble al inicio y al final del comentario
- """Este es otro ejemplo de
- comentarios"""

Outline

Especificando a python

Especificando a python

- Para que python sepa qué tipo de dato deseamos utilizar, es necesario realizar un ”cast” al tipo de dato. En caso de no especificar el tipo de dato que deseamos, python otorgará el que el crea más apropiado a la variable, por ejemplo:
- str(texto)
- int(numero)
- float(texto)

Especificando a python

- Para que python sepa qué tipo de dato deseamos utilizar, es necesario realizar un ”cast” al tipo de dato. En caso de no especificar el tipo de dato que deseamos, python otorgará el que el crea más apropiado a la variable, por ejemplo:
- str(texto)
- int(numero)
- float(texto) #esto es válido ;)

Números

Números

- Los dos tipos principales de números en python son los enteros (integers) y reales (floats)
- En caso de no especificar el tipo de número (o de dato) python decide cuál es el más apropiado (a veces se equivoca)
- Integers (int): 2, 5, 4000, -2
- Floats (float): 2.5, 2.9, -3.1

Texto

Texto

- Una variable que almacena texto se llama cadena o string, y para que python las reconozca deben estar en comillas simples o dobles... o triples si necesita varias líneas Ejemplo: ”Texto de ejemplo”
- el operador + une dos strings...

Outline

input y print

input y print

- Para desplegar algún dato de nuestro programa, basta con utilizar el comando print, mientras que para asignar algún valor dado por el usuario a una variable, se utiliza el comando input. Ejemplos:
- print(”Desplegando un texto”)
- var=input(”Ingresa un texto”)
- print(var)

Outline

Arreglos (colecciones)

Arreglos (colecciones)

- Existen 4 tipos de arreglos en python (legacy), los cuáles permiten almacenar varios datos en una sola variable:
- Listas (List): Ordenada e intercambiable. Permite miembros duplicados.
- Tuplas (Tuple): Coleccion ordenada y no intercambiable. Permite miembros duplicados.
- Conjuntos (Set): Desordenada y no indexada. No permite miembros duplicados.
- Diccionarios (Dictionary): Desordenada, intercambiable e indexada. No permite miembros duplicados.

Listas

Listas

- La lista es una colección ordenada e intercambiable.
- Para crear una lista en python se escriben sus elementos entre corchetes [a, b, c,”hola”].
- mi_lista=[”un_elemento”,”otro_elemento”,”tercer_elemento”, ”otro”, ”otro”, ”ya”, ”son”, ”muchos”]
- print(mi_lista)
- Para accesar a una lista, especificamos el número de indexación al que queremos accesar (el primer elemento se indentifica con el número 0).
- print(mi_lista[1])

Listas

Listas

- python permite indexamiento negativo...

Listas

- python permite indexamiento negativo... no lo usen)=
- Es posible accesar a una sublista de un rango especificado
- print(mi_lista[2:5])
- print(:5)

Operaciones de listas

Operaciones de listas

- Las operaciones más habituales de Python son las siguientes:
- lista[i]: Devuelve el elemento que está en la posición i de la lista.
- lista.pop(i): Devuelve el elemento en la posición i de una lista y luego lo borra.
- lista.append(elemento): Añade elemento al final de la lista.
- lista.insert(i, elemento): Inserta elemento en la posición i.
- lista.extend(lista2): Fusiona lista con lista2.
- lista.remove(elemento): Elimina la primera vez que aparece elemento.

Diccionarios

Diccionarios

- Un diccionario (tal como los convencionales) es una palabra que tiene asociado ”algo”, y a diferencia de las listas, los diccionarios no tienen orden.
- Para crear un diccionario se ponen sus elementos entre llaves {”a”:”Almidon”,”b”:”bueno”,:}. se les denomina llaves (keys) a las palabras (a y b) y valores a sus definiciones (Almidon y bueno). No es posible tener dos llaves iguales, aunque si dos valores.
- Ejemplo:
- diccionario = {’Piloto 1’:’Fernando Alonso’, ’Piloto 2’:’Kimi Raikkonen’, ’Piloto 3’:’Felipe Massa’}
- print(diccionario)

Operaciones más comunes de los diccionarios

Operaciones más comunes de los diccionarios

- diccionario.get('key'): Devuelve el valor que corresponde con la key introducida.
- diccionario.pop('key'): Devuelve el valor que corresponde con la key introducida, y luego borra la key y el valor.
- diccionario.update({'key':'valor'}): Inserta una determinada key o actualiza su valor si ya existiera.
- "key" in diccionario: Devuelve verdadero (True) o falso (False) si la key (no los valores) existe en el diccionario.
- "definicion" in diccionario.values(): Devuelve verdadero (True) o falso (False) si definición existe en el diccionario (no como key).

Outline

Conjuntos

Conjuntos

- También existen los conjuntos (sets) aunque son menos utilizados.
- Tal como los diccionarios, se crean usando llaves, pero sus elementos se separan por coma como si se tratara de una lista
- Ejemplo: conjunto = {'Fernando Alonso', 'Kimi Raikkonen', 'Felipe Massa'}
- Los sets permiten realizar operacionas matemáticas típicas de conjuntos como unión, intersección, etc.

Operaciones más comunes en conjuntos

Operaciones más comunes en conjuntos

- A — B: Unión entre el conjunto A y B (Los elementos del conjunta A y los elementos del conjunto B)
- A & B: Intersección entre el conjunto A y B (los elementos que están en ambos conjuntos)
- A – B: Diferencia entre el conjunto A y B (los elementos que están en A pero no están en B)
- A^B: Diferencia simétrica entre el conjunto A y B (los elementos que están en A o en B pero no en los dos)

Outline

Indentación

Indentación

- Python considera es sensible a la indentación, por lo que es necesario que sus líneas de código se encuentren agrupadas con el mismo número de espacios a la izquierda de cada línea. Es recomendado utilizar bloques de cuatro espacios, sin embargo cualquier otra cantidad de espacios (o tabuladores) pueden ser utilizados de igual forma (no se recomienda).

Outline

Control de flujo y condicionales

Control de flujo y condicionales

- Todo programa con una ligera complejidad necesita ”tomar decisiones” en algunas bifurcaciones del problema, donde, dependiendo de alguna condición se realiza una u otra acción.
- Esta bifurcación se lleva a cabo con el comando if (condición principal), con sus opcionales elif (condiciones adicionales, pueden ponerse tantos como sean deseados) y else (el default en caso de no cumplirse ninguno).

Ejemplo

Ejemplo

- Revisar ejercicio ”ifs.py”

Condiciones en python

Condiciones en python

- Las condiciones utilizadas con más frecuencia son:
- a == b –>Indica si a es igual a b
- a <b
- a >b
- not –>NO: niega la condición que le sigue.
- and –>Y: junta dos condiciones que tienen que cumplirse las dos
- or –>O: junta dos condiciones y tienen que cumplirse alguna de las dos.

while en python

while en python

- Cuando una tarea debe repetirse hasta que cierta condición se cumpla (no sabemos cuántas veces se repetirá), es recomendado utilizar el comando while, que se usa de la siguiente forma:
- vuelta=1
- while vuelta <10:
- print(”vuelta ”+str(vuelta))
- vuelta=vuelta+1

for en python

for en python

- En ocasiones necesitamos repetir varias veces una determinada tarea, pero conocemos la cantidad de veces que la repetiremos. Para esto es recomendado utilizar en Python el comando for, que se usa de la siguiente forma:
- for vuelta in range(1,10):
- print(”vuelta ”+str(vuelta))
- PD. En el caso del for (a diferencia del while) no es posible realizar un loop infinito.
- es posible utilizar el for con cualquier objeto que pueda ser iterado (recorrer sus elementos)

Outline

No reinventes la rueda

No reinventes la rueda

- Un módulo es un programa que viene en un ”paquete” de python y existen para contener rutinas que son utilizadas habitualmente y que no sea necesario reescribirlas cada que son necesitadas. Estos módulos son creados muchas veces por programadores como tu, y se comparten en Github principalmente asi que si hay alguna rutina que uses mucho y no existe ningun paquete que la contenga, es tu oportunidad de brillar ;)
- Gracias a estos módulos es posible realizar cosas complejas en pocas líneas de código

Instalando un módulo

Instalando un módulo

- Los módulos son instalados de forma distinta en los diversos sistemas operativos, en el caso de Linux, basta con bajar el programa pip y escribir en nuestra línea de comandos ”pip install <módulo>” para instalarlo... en algunas distribuciones es incluso más fácil que eso, pero recuerda que internet es tu amigo (internet, google no)

Módulos ”legacy” más usados

Módulos ”legacy” más usados

- os
- datetime
- time
- sys
- locale
- MySQLdb
- Cada uno de estos módulos contiene funciones ya incluídas que pueden ser llamadas si esl módulo se encuentra instalado y cargado.

Invocando funciones de módulos

Invocando funciones de módulos

- Para llamar a una función de un módulo es necesario primero cargar el módulo a nuestro programa, con el comando:
- import modulo
- Posteriormente basta con llamar a la función deseada en la forma:
- modulo.funcion()

Declaración de funciones

Declaración de funciones

- Cuando existe alguna rutina que queremos repetir constantemente durante nuestro programa, es útil crear una función, la cuál nos permitirá llamar esta rutina con un solo comando (en lugar de escribir todo). Ejemplo:
- def myfunc(): #Aquí especificamos el nombre de la función
- x = 300
- print(x)
- myfunc() #Esto llama a la función

Scope

Scope

- Una variable se encuentra disponible solamente dentro de la región donde fue creada, y a esto de le denomina ”scope” o ”alcance”.
- Una variable que es creada dentro de una función pertenece solamente al ”scope local” de esa función, y solamente podrá ser usada dentro de esa función
- ***Recuerda que la indentación afecta al scope de las variables

Outline

Interactuando con archivos

Interactuando con archivos

- Con Python podemos manipular archivos (leer, escribir...) de una forma bastante sencilla

Creando archivos de texto

Creando archivos de texto

- Para crear un archivo hay que seguir los siguientes pasos:
- Paso 1) Abre un archivo y asignalo a una variable (con la opción "w+")
- Paso 2) Escribimos la informacion que queramos que contenga el archivo
- Paso 3) Cerramos el archivo

Creando archivos de texto (Ejemplo)

Creando archivos de texto (Ejemplo)

- Paso 1) f=open(”archivo.txt”, ”w+”)
- Paso 2) for i in range(10):
- f.write(”This is line %d \r \n” % (i+1))
- Paso 3) f.close()

Escribir datos sobre un archivo

Escribir datos sobre un archivo

- Para escribir sobre un archivo hay que seguir los siguientes pasos:
- Paso 1) Abre un archivo y asignalo a una variable (con la opción ”a+”)
- Paso 2) Escribe lo que desees con el comando <variable >.write(”texto”)
- Paso 3) Cerramos el archivo

Escribir datos sobre un archivo (Ejemplo)

Escribir datos sobre un archivo (Ejemplo)

- Paso 1) f=open(”archivo.txt”, ”a+”)
- Paso 2) for i in range(2):
- f.write(”Appended line %d \r \n” % (i+1))
- Paso 3) f.close()

Leer un archivo

Leer un archivo

- Para Leer un archivo hay que seguir los siguientes pasos:
- Paso 1) Abre un archivo y asignalo a una variable (con la opción "r+")
- Paso 2) Leemos la información deseada con el comando <variable >.read()... y es probable que quieras almacenarla en alguna variable
- Paso 3) Cerramos el archivo

Leer un archivo (Ejemplo)

Leer un archivo (Ejemplo)

- Paso 1) f=open(”archivo.txt”, ”r”)
- Paso x) if f.mode == ’r’: #esto es opcional, pero recomendado
- Paso 2) contents =f.read()
- Paso 3) f.close()

Outline

frame con cuadrito

- (=

frame con cuadrito

- (=

Bibliografías

https://www.tutorialspoint.com/python/python_basic_ope
https://www.tutorialsteacher.com/python/python-compari
https://www.learnpython.org/es/Hello,%20World!
https://www.w3schools.com/python/python_lists.asp
https://www.guru99.com/reading-and-writing-files-in-py
https://www.tutorialpython.com/modulos-python/