Guide 1 conditionals and loops

September 6, 2018

1 Computación II - 2018

Guía No. 2: Condicionales y ciclos en Python
Ingeniería en Estadística - Universidad de Valparaíso
Profesor: Eduardo Jorquera - eduardo.jorquera@postgrado.uv.cl

1.0.1 Condicionales

La sintaxis básica de los if y else en Python tiene la siguiente estructura:
if expression: statement(s)
else: statement(s)
Un ejemplo sencillo es el siguiente

```
In [3]: a=2
        if a>2:
            print("a es mayor que 2, porque a = ", a)
        else:
            print("a es menor o igual que 2, porque a = ", a)
```

```
a es menor o igual que 2, porque a =  2
```

- Qué hace el siguiente código? en particular qué hace la línea No. 2?

```
In [4]: import numpy as np
        a = np.random.random_sample() #Qué hace esta línea?
        print ("a: " + str(a))
        if a<0.5:
            result=1
        else:
            result=0
        print ("result:"+str(result))
```

```
a: 0.6479006854610555
result:0
```

- Modifique el código anterior para que se ejecute 10 veces usando un *for*, no se preocupe por guardar el resultado. Escriba el código modificado bajo este párrafo:

Otra forma de hacer el problema anterior es el siguiente, describa qué elementos nuevos aparecen en este código? (ej. np.random.rand, np.where) qué hace cada uno de ellos?

```
In [1]: import numpy as np
        a = (np.random.rand(1,10)) #create a random (1,10) array
        print ("a: \n" + str(a))
        result = (np.where(a>0.5,1,0))
        print ("result: \n" +str(result))
```

```
a:
[[0.44784672 0.87537097 0.18479674 0.38373104 0.49595293 0.95969931
  0.45123738 0.92915686 0.50981046 0.86915955]]
result:
[[0 1 0 0 0 1 0 1 1 1]]
```

Los ciclos no solo pueden funcionar en arreglos de números, también de conjuntos:

```
In [7]: animals = ['cat', 'dog', 'monkey']
        for animal in animals:
            print(animal)
```

```
cat
dog
monkey
```

Modifique el código anterior para incluir la longitud de la cadena de caracteres para cada animal, es decir debe imprimir "cat : 3". Incluya el código en la siguiente celda:

- Combinemos ahora *for* e *if*, qué hace el siguiente código?

```
In [3]: nums = [0, 1, 2, 3, 4]
        even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
        print(even_num_to_square)
```

```
{0: 0, 2: 4, 4: 16}
```

Modifique el código anterior para que ahora nums sea un arreglo de números entre 1 y 100 y además la lista incluya el logaritmo natural para cada uno de los valores en nums. Incluya el código modificado a continuación:

Los ciclos *while* pueden ser útiles para iterar un evento hasta que cierta condición suceda; funciona de la siguiente forma:

```
In [22]: i = 1

         while True:
             print(i)
             i = i + 1
             if(i > 3):
                 break
```

```
1
2
3
```

Existe una función para generar números aleatorios entre 0 y 1, dentro de al librería numpy, donde el argumento determina el largo del vector creado. Ésta es la siguiente:

```
In [23]: import numpy as np
         np.random.rand(5)
```

```
Out[23]: array([0.94851006, 0.59629726, 0.20316919, 0.05352698, 0.04322013])
```

Actividad

- Haga un ciclo que sirva para generar números aleatorios entre 0 y 1 (uno tras otro). Deténgase hasta que se genere el valor 0.99, guarde cada uno de los resultados en en un vector
- Haga un gráfico de los valores generados anteriormente.

Las funciones en python pueden definirse de la siguiente manera, entregando dentro del primer paréntesis los argumentos que se utilizarán. Una función tiene la siguiente estructura básica:

```
In [30]: def suma(a,b):
             print("La suma es: ",a+b)

         suma(5,2)
```

```
La suma es:  7
```

Actividad

- Genere 1.000 números aleatorios, grafique la razón entre uno y la suma de todos los valores anteriores.

TAREA (se revisará al principio de la siguiente clase)

- Usando la librería numpy, genere 10.000 valores aleatorios x entre 0 y 1, imprima los valores de $y = \log_{10}(x)$, que cumplan $y > \frac{1}{2}$, guarde los valores que cumplen la condición en un vector (use un ciclo for).
- Haga un gráfico de la función en estos valores guardados anteriormente.
- Genere un código para esta tarea en Python y otro en R.