Lazo de repeticion For y While

Lazo For

Los lazos de repeticion (o iteración) permiten construír un código que se ejecutara un número de veces. En Python los ciclos de repetición descansan sobre el concepto de un *iterador*. Tomemos la lista de los números primos:

In [ ]:

```
primos = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 9
7]  #es una lista
```

Podems utiliar la instrucción *range(n)* que genera un *iterable* entre *0* y *n-1*. Un iterable es un elemento que puede formar parte de un lazo *for*. Este lazo va extrayendo cada dato del iterable y lo va guardando en una variable. En este caso, se genera un rango de las posiciones de una lista (las posiciones de una lista empiezan en 0):

In [ ]:

```
for i in range(len(primos)):
    print (primos[i])
```

La instrucción *rag(len(primos)* obtiene el numero de elementos de la lista *primos*. Como el primer elemento de la lista esta asociado al indice 0, y la instrucción *range(n)* genera valores entre *0* y *n-1*, el iterable tendrá los valores de todos los indices de una lista.

Sin embargo, ¡la lista *primos* en si misma es un iterable! Por lo tanto la lista misma se puede usar dentro de un lazo:

In [ ]:

```
for each_number in primos:
    print (each_number)
```

Esta construcción es mucho mas *"pythonica"*, ya que el codigo es mucho mas legible. Sin embargo, la función *range* puede servir para generar un rango de valores para casos especiales:

In [ ]:

```
# Genera un rango entre 10 y 20
for i in range(10,21):
    print(i)
```

In [ ]:

```
# Genera un rango entre 0 y 20 en pasos de 5
for i in range(0,21,5):
    print(i)
```

In [ ]:

```
# Si voy a generar un rango de valores decrecientes, necesito especificar el paso como negativo
for i in range(10, 0, -1):
    print(i)
```

Lazo While

Otra forma de establecer un lazo de iteración es con la instrucción *while* que requiere de una condición, de tal forma que mientras esta condición retorne un valor *True* las instrucciones dentro del bloque de repetición continuarán repitiendose.

In [ ]:

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Las lazos *while* son ideales para establecer un lazo del que no se conoce el número de iteraciones que se van a suceder, pero se conoce la condición que hara que este proceso se mantenga. Por ejemplo, si se tiene un robot que se desplaza hacia adelante y tiene un sensor que mide la distancia hacia una pared, se puede establecer un codigo como el siguiente:

```
distancia = sensor()      # Hay una instrucción "sensor" que mide la distancia del robot contr
a un obstaculo en cm
if distancia > 10:
    avanzar()
```

Este ejemplo hara que el robot, sin importar a que distancia se encuentre de una pared, se desplazará hacia esta hasta que este a 10 cm de esta (se asume que *distancia* se va reduciendo con el tiempo).

Por ejemplo, supongamos que queremos un programa que sume las notas ingresadas hasta que se ingrese un *valor centinela*, es decir, un valor que indique que ya no se desean seguir ingresando notas.

In [ ]:

```
suma = 0
num_notas = 0
nota = 0


while nota != -1:
    nota = int(input("Ingrese una nota [-1 para salir]: "))
    suma = suma + nota
    num_notas += 1

promedio = (suma + 1) / (num_notas - 1)   # Hay que restar la nota con valor -1
print("El promedio de las notas ingresadas es {:.1f}".format(promedio))
```

Lazo Do... While (o como contruirlo utilizando *break*)

El ejemplo anterior tiene un problema: hay que ajustar la suma de las notas parra evitar que a nota -1 se agregue al calculo, tanto en valor como en la cantidad de notas ingresadas. Esto se podria evitar utilizando un lazo do...while que evalua la condición luego de la ejecución del bloque de iteracion y no antes como en el caso anterior.

Sin embargo, Python no tiene implementado este tipo de lazo, por lo que hay que construirlo utilizando la instruccion *break*, que detiene un lazo:

In [ ]:

```
suma = 0
num_notas = 0
nota = 0


while True:
    nota = int(input("Ingrese una nota [-1 para salir]: "))
    if nota == -1:
        break
    suma = suma + nota
    num_notas += 1

promedio = suma / num_notas
print("El promedio de las notas ingresadas es {:.1f}".format(promedio))
```

En este caso, se crea un *lazo infinito* ya que la condición sera siempre True. Sin embargo, antes de realizar algun calculo, se verifica el valor de entrada y si este es igual a valor centinela, se detiene el lazo

while en los lazos

En Python tanto un lazo *for* como un lazo *while* soportan la instrucción *else*. Esto con la intención que el bloque de teración sea más legible. Los dos siguientes codigos realizan lo mismo, pero el segundo es mas legible

In [ ]:

```
num = 0
# Mientras el numero a imprimir sea menor que 10 se imprime un valor que se va incrementado
while num < 10:
    print(num)
    num += 1
# Se imprime final de lista
print("Fin de lista")
```

In [ ]:

```
num = 0
# Mientras el lazo se sostega se imprime un numero que va aumentando...
while num < 10:
    print(num)
    num += 1
else:
    # ...de lo contario, si el lazo termino (por una condición False y no por un break) se realiza lo s
iguiente
    print("Fin de lista")
```

La fiilosofia detras de Python es que el código sea legible, y esto se consigue agragando la instrucción *else* a los lazos