

# Clase 11

Diego Armando Santillán Arriaga

21/01/2019

## 1 Notas:

Para que una variable pueda ser reconocida como un número flotante debe de escribirse de la siguiente manera:

### `float("variable")`

Los archivos .pyc son una versión compilada de un módulo de Python. Son más rápidos.

**.gitignore** : Este archivo se crea en gituhub. En él se listan las terminaciones de los archivos que no nos interesan como por ejemplo .pyc o .aux. Para escribir las terminaciones se utiliza el siguiente formato:

"terminación" (el asterisco sirve para indicar que todos los archivos que tienen esa terminación serán ignorados)

Para importar varias funciones de un módulo en un solo comando utilizamos

```
from "nombre del módulo" import *
```

## 2 Listas (continuación):

Se pueden escribir variables dentro de un a lista. Al dar valores a estas variabes, automáticamente se asignan a su lugar correspondiente en la lista.

Para crear una lista que tiene n elementos contados desde el cero se utiliza:

```
for i in range("número"):
```

Esta lista va desde el 0 hasta el número n-1

```
for i in range(len("lista"))
```

Este ciclo toma cada elemento de una lista que tiene los índices de la lista "lista"

La expresión que está más anidada es la primera que se ejecuta

```
>>>L1
[0, 10, 15, 20]
>>> for i in range(len(L1)):
L1[i] += 5
```

```
>>>L1
[5, 15, 20, 25]
```

"nombre de la lista"[i (índice de algún elemento de la lista)] regresa el iésimo elemento de la lista.

Para utilizar en un ciclo for un índice y su valor correspondiente en la liste se utiliza enumerate de la siguiente forma:

for "índice", "variable que representa el valor correspondiente" in enumerate("lista"):

Una forma condensada de crear listas es:

"nombre de la lista"= ["fórmula o variable" + ciclo for]

Ejemplos

```
>>> n=12; gradosC = [-5 + i*0.5 for i in range(n) ]
>>> gradosC
[-5.0, -4.5, -4.0, -3.5, -3.0, -2.5, -2.0, -1.5, -1.0, -0.5, 0.0, 0.5]
>>> C_mas_7 = [ C+7 for C in gradosC]
>>> C_mas_7
[2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5]
```

zip("lista1", "lista2") toma los valores de ambas listas correspondientes al mismo índice (también aplica para más de dos listas)

¿Para que sirven los paréntesis, llaves y corchetes en Python? paréntesis = en una función permite agregar los argumentos. corchetes = sirven por acceder a un elemento de una lista o definirla. s