**Clase 11 de Octubre 2016**

-Python 2.7 tiene módulos de datos de Análisis Numérico que no posee Python 3.5

-EOL End of Line

-raw\_input: transforma datos de texto o traduce (no hace falta poner comillas para texto, solo versión 2.7)

-4 tipos de números: enteros, reales complejos y flotantes

**type(8)**

=> <type 'int'>

**type(8.09)**

=> <type 'float'>

**type(9.0)**

=> <type 'float'>

**type(8+7j)**

=> <type 'complex'>

**type(337738484940022939033)**

=> <type 'long'>

**from sys import maxint**

=> None

**maxint**

=> 9223372036854775807

**8/5**

=> 1

**8/5.0**

=> 1.6000000000000001

**9.0//2**

=> 4.0

**9.0/2**

=> 4.5

**4!=9**

=> True

-El “not” devuelve el valor contrario a la comparación.

**2>3**

=> False **not(2>3)**

=> True

* Tipo boleano: es verdadero o falso

**type(4>5)**

=> <type 'bool'>

-la interpretación o precisión en sumas no siempre es exacta en Python

**b=0.1+0.5**

=> None

**b**

=> 0.59999999999999998

**a=0.1+0.2**

=> None **round(a,2)**

=> 0.29999999999999999

-En python una variable puede manejar cualquier tipo de dato (no hay que definir lo que contendrá)

**g=7**

=> None

**g='Ana'**

=> None

Tratar de definer variables con nombres significativas

**Name = "Manuel"**

=> None **Lastname = "Torres"**

=> None **print Name, Lastname**

Manuel Torres

=> None

-Estandar de definición de variables con nombres significativos:

**primer\_apellido='Gonzales'**

=> None

**interes\_compuesto=0.3**

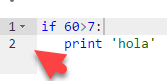
=> None

-Mayúscula para variables numéricas

**MAX\_ALUMNOS = 90**

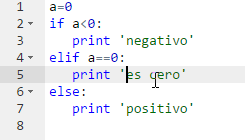
**IMPORTANTE**

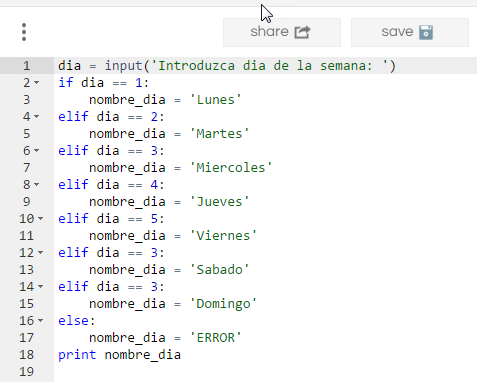
**-para reglas de condicionales “if” deben tener tres espacio por regla o obligatorio**

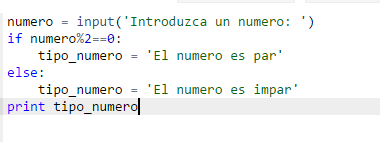


-Encontrar errores o “bug” y depurar “debug” o “debug

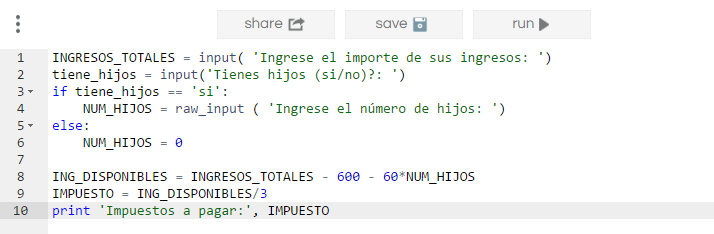
-if

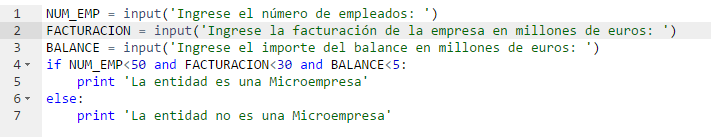




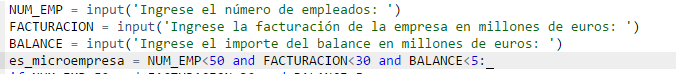


-Impuesto a partir del número de hijos:

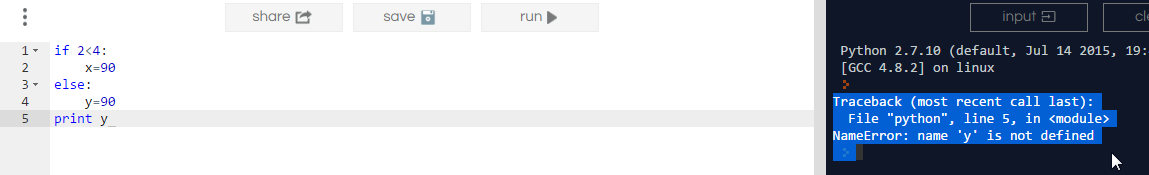




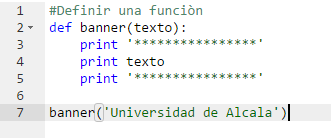
Devolver resultados booleano



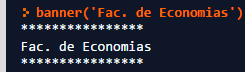
Da error porque se genera el condicional sin definir una variable:

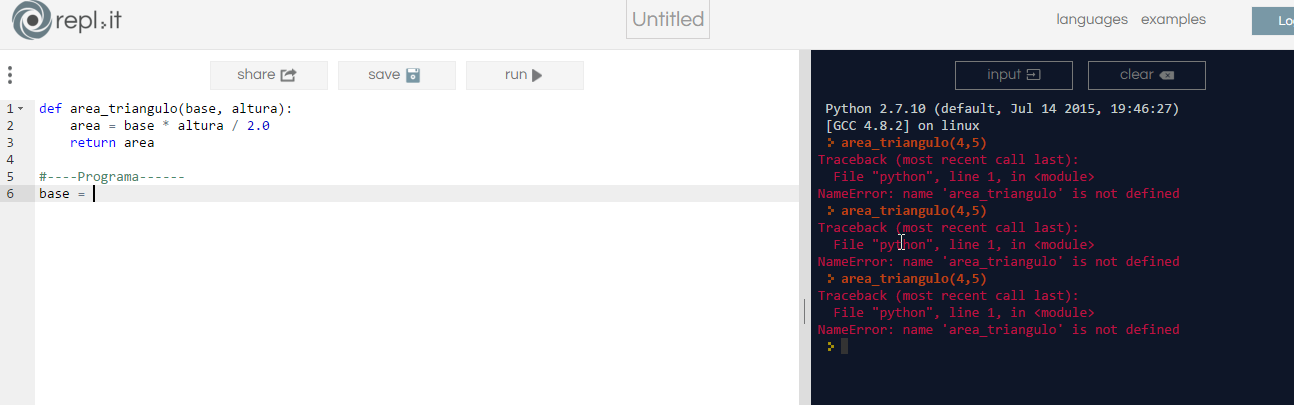


**FUNCIONES:** Instrucciones que se pueden usar varias veces

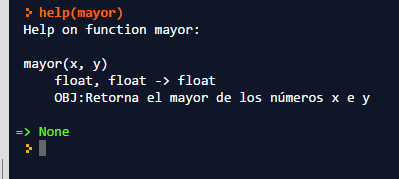








**Clase 17 de Octubre 2016**



De los ejercicios anteriores:

def mayor(x,y):

""" float, float -> float

OBJ:Retorna el mayor de los números x e y

"""

if x>y:

return x

elif x == y:

return 0

else:

return y

y = función(y)

print(texto +variable texto)

**Bibliotecas (library)**

Import xxx.py, math viene en Python (import math).

Solo algo especifico:

From math import pi

**Instrucciones selectivas:**

Selectives (if)

**Instrucciones repetitivas o bucles (Loop)**

**-“for”:**

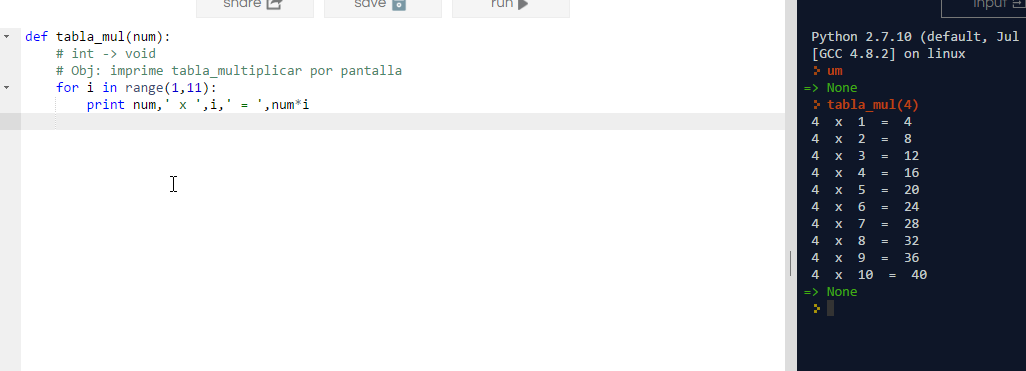
For x in lista:

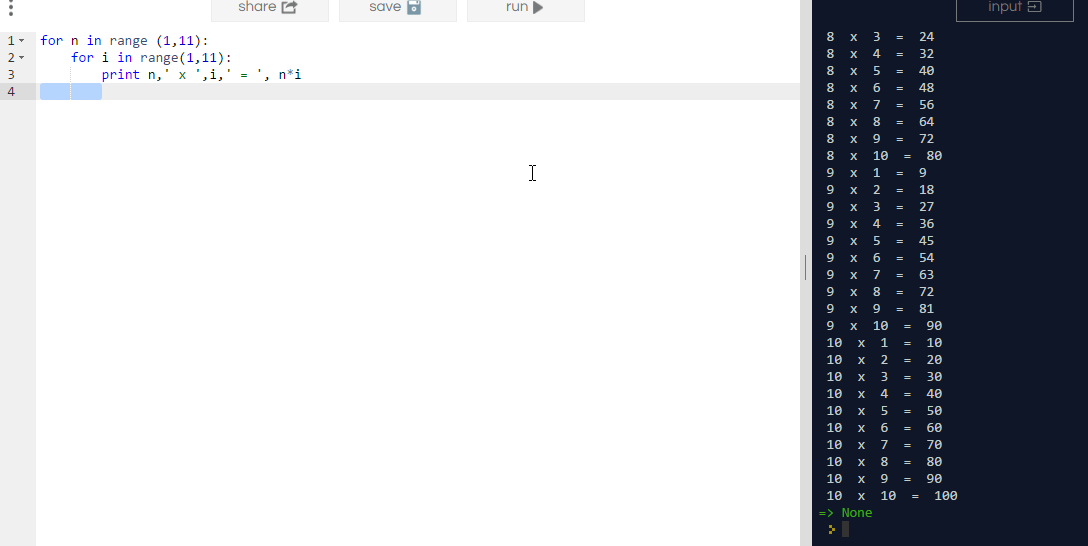
Instrucciones

For x range (2,10) => en un rango menos 1 o del 2 al 9

For x range (2,10,2) => rango de 2 en 2 es decir 2,4,6,

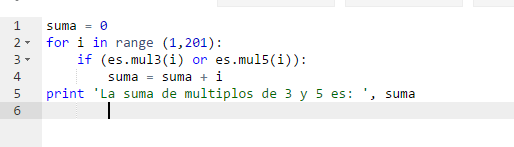
Xrange?



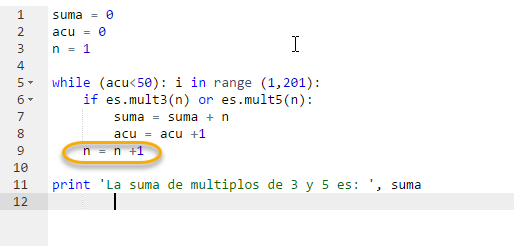


**Instrucción “While”:**

**Ejercicios “For”**



**Ejercicio “while”**



**Instrucciones “break” o “continue” usadas en bucles**

**Listas:**

Estructura de datos con varios elementos separados por coma y mutables, podemos tener números, textos y otros tipos en la misma lista, siempre representados por **corchetes**.

Index: posición valor, in: si esta un valor, count: cuenta un valor, length: longitud, append: agrga valor , extend: extiende lista, suma o “+”, remove: quita valor, sort: ordena, reverse: lista al revés y pop

Pueden existir listas de listas (ej: matriz) , o listas de listas de listas (ej: cubos o 3D). Y sucesivamente.

**TUPLAS:**

Se crean con paréntesis y no son modificables

**STRING:**

Ejemplo:

a=’Hola’

print a[1] = o

Funciones “upper” y “lower”

b = a.upper(), para sobrescribir “a” a mayúscula a= a.upper()

a=’Hola q tal’

b=a.split()

[‘Hola’,’q’,’tal’]

Se puede hacer b=a.split(a) y separ por la letra “a”

‘Hola’.find(‘l’)

2

**Claves:**

Diccionario?

Las claves son uniformes y únicas

A = {1:’Uno’, 2:’Dos’} => A[1] => ‘Uno’

A = {8:’Uno’, 12:’Dos’} => A[8] => ‘Uno’

12 in A => True

For x,y in diccionario.items():

**Archivos:**

f = open(‘nombre.txt’,’r’) el r significa lectura

for línea in f:

f.close()