|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | M. I. Marco Antonio Martínez Quintana |
| *Asignatura:* | Estructuras de Datos y Algoritmos de programación I |
| *Grupo:* | 17 |
| *No de Práctica(s):* | 10 |
| *Integrante(s):* | Marcos Vega Alvarez |
| *No. de Equipo de cómputo empleado:* | 25 |
| *No. de Lista o Brigada:* | 39 |
| *Semestre:* | 2020-2 |
| *Fecha de entrega:* | 14 de Abril de 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo:**

Aplicar las bases del lenguaje de programación Python en el ambiente de Jupyter notebook.

**Introducción:**

### Estructuras de control de flujo condicionales

"[...] Los condicionales nos permiten comprobar condiciones y hacer que nuestro programa se comporte de una forma u otra, que ejecute un fragmento de código u otro, dependiendo de esta condición [...]"

Cita textual del libro Python para Todos de Raúl González Duque (http://mundogeek.net/tutorial-python/)

Las estructuras de control condicionales, son aquellas que nos permiten evaluar si una o más condiciones se cumplen, para decir qué acción vamos a ejecutar. **La evaluación de condiciones**, solo **puede arrojar** 1 de 2 resultados: **verdadero o falso** (True o False).

En la vida diaria, actuamos de acuerdo a la evaluación de condiciones, de manera mucho más frecuente de lo que en realidad creemos: Si el semáforo está en verde, cruzar la calle. Si no, esperar a que el semáforo se ponga en verde. A veces, también evaluamos más de una condición para ejecutar una determinada acción: Si llega la factura de la luz y tengo dinero, pagar la boleta.

Las estructuras de control de flujo condicionales, se definen mediante el uso de tres palabras claves reservadas, del lenguaje: if (si), elif (sino, si) y else (sino).

Veamos algunos ejemplos:

1) Si semáforo esta en verde, cruzar la calle. Sino, esperar.

if semaforo == verde:

print "Cruzar la calle"

else:

print "Esperar"

2) Si gasto hasta $100, pago con dinero en efectivo. Si no, si gasto más de $100 pero menos de $300, pago con tarjeta de débito. Si no, pago con tarjeta de crédito.

if compra <= 100:

print "Pago en efectivo"

elif compra > 100 and compra < 300:

print "Pago con tarjeta de débito"

else:

print "Pago con tarjeta de crédito"

3) Si la compra es mayor a $100, obtengo un descuento del 10%.

importe\_a\_pagar = total\_compra

if total\_compra > 100:

tasa\_descuento = 10

importe\_descuento = total\_compra \* tasa\_descuento / 100

importe\_a\_pagar = total\_compra – importe\_descuento

### Estructuras de control iterativas

A diferencia de las estructuras de control condicionales, las iterativas (también llamadas cíclicas o bucles), nos permiten ejecutar un mismo código, de manera repetida, mientras se cumpla una condición.

En Python se dispone de dos estructuras cíclicas:

* El bucle while
* El bucle for

Las veremos en detalle a continuación.

#### Bucle while

Este bucle, se encarga de ejecutar una misma acción "mientras que" una determinada condición se cumpla. Ejemplo: Mientras que año sea menor o igual a 2012, imprimir la frase "Informes del Año año".

*# -\*- coding: utf-8 -\*-*

anio = 2001

while anio <= 2012:

print "Informes del Año", str(anio)

anio += 1

#### Bucle for

El bucle for, en Python, es aquel que nos permitirá iterar sobre una variable compleja, del tipo lista o tupla:

1) Por cada nombre en mi\_lista, imprimir nombre

mi\_lista = ['Juan', 'Antonio', 'Pedro', 'Herminio']

for nombre in mi\_lista:

print nombre

2) Por cada color en mi\_tupla, imprimir color:

mi\_tupla = ('rosa', 'verde', 'celeste', 'amarillo')

for color in mi\_tupla:

print color

En los ejemplos anteriores, nombre y color, son dos variables declaradas en tiempo de ejecución (es decir, se declaran dinámicamente durante el bucle), asumiendo como valor, el de cada elemento de la lista (o tupla) en cada iteración.

Otra forma de iterar con el bucle for, puede emular a while:

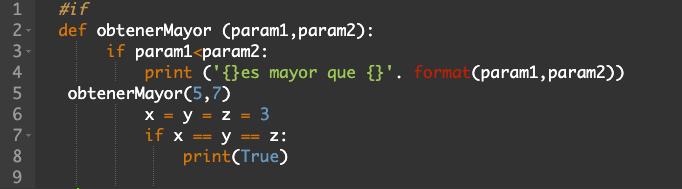
3) Por cada año en el rango 2001 a 2013, imprimir la frase "Informes del Año año":

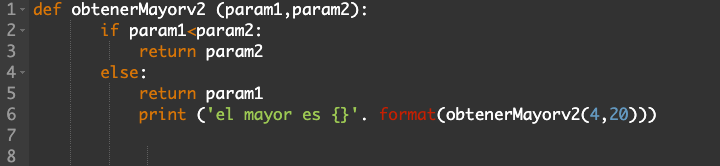
*# -\*- coding: utf-8 -\*-*

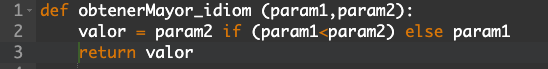
for anio in range(2001, 2013):

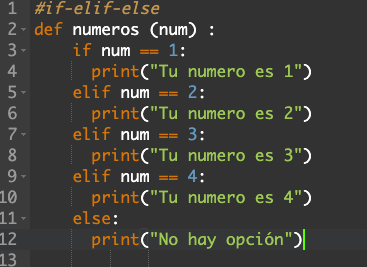
print "Informes del Año", str(anio)

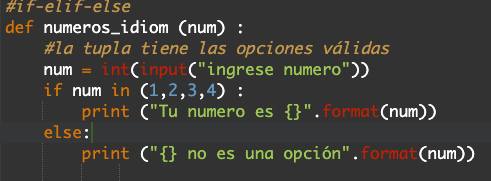
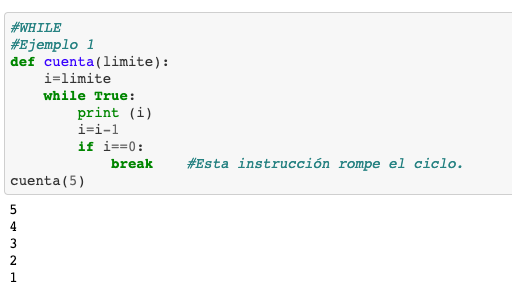
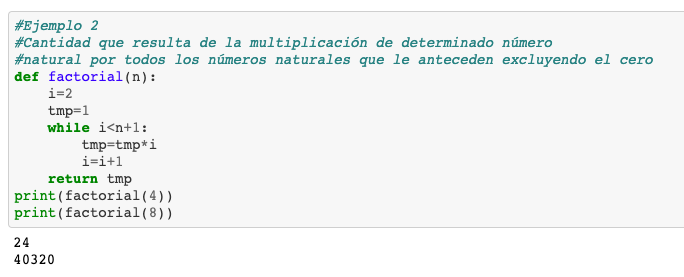
**Desarrollo:**

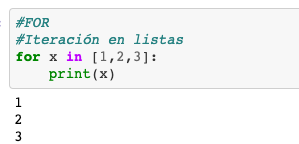


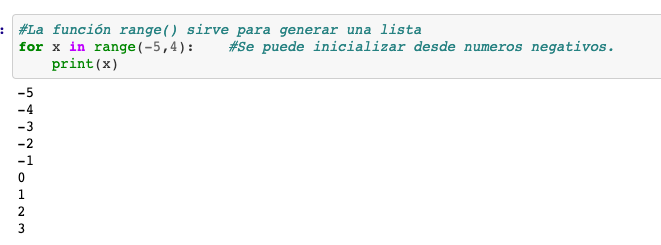


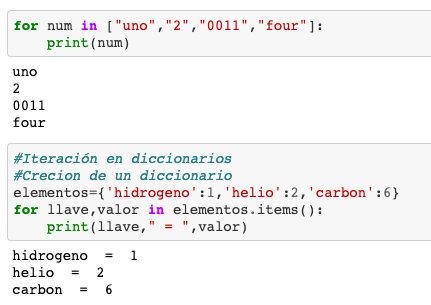






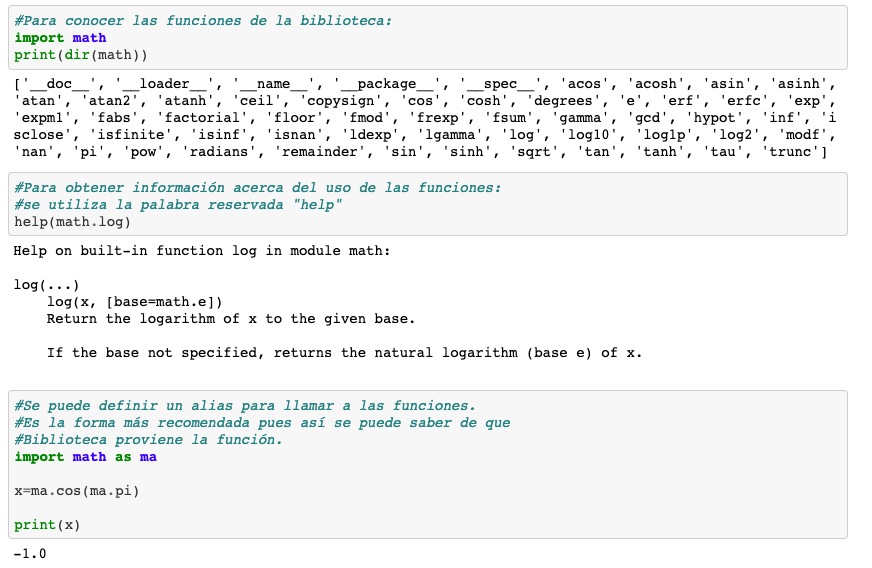
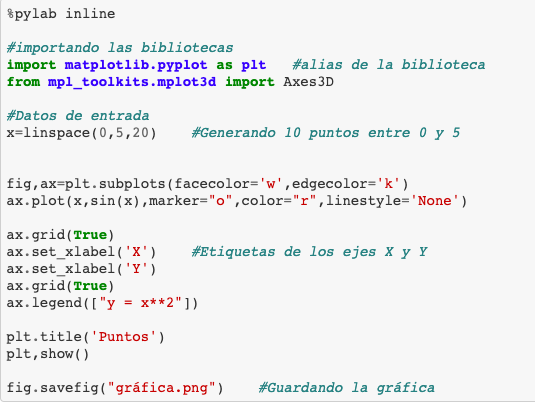












**Conclusión:**

Las [características](https://es.wikipedia.org/wiki/Python#Caracter%C3%ADsticas_y_paradigmas) del lenguaje de programación Python se resumen a continuación:

* Es un [lenguaje interpretado](https://es.wikipedia.org/wiki/Lenguaje_interpretado), **no compilado**, usa [tipado dinámico](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/caracteristicas.html#python-tipado-dinamico), [fuertemente tipado](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/caracteristicas.html#python-fuertemente-tipado).
* Es [multiplataforma](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/caracteristicas.html#python-multiplataforma), lo cual es ventajoso para hacer ejecutable su código fuente entre varios sistema operativos.
* Es un lenguaje de programación [multiparadigma](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_multiparadigma), el cual soporta varios paradigma de programación como [orientación a objetos](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion9/poo.html#python-poo), [estructurada](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion5/programacion_estructurada.html#python-programacion-estructurada), [programación imperativa](https://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa) y, en menor medida, [programación funcional](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional).
* En Python, el formato del código (p. ej., la indentación) es estructural.

## **Fuertemente tipado**

El [fuertemente tipado](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_fuertemente_tipado) significa que el tipo de valor no cambia repentinamente. Un [string](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/tipo_cadenas.html#python-str) que contiene solo dígitos no se convierte mágicamente en un número. Cada cambio de tipo requiere una conversión explícita. A continuación un ejemplo de este concepto:

*# varible "valor1" es integer, varible "valor2" es string*

valor1, valor2 = 2, "5"

*# el metodo int() es para convertir a integer*

total = valor1 + int(valor2)

*# el metodo str() es para convertir a string*

**print** "El total es: " + str(total)

## **Tipado dinámico**

El [tipado dinámico](https://es.wikipedia.org/wiki/Tipado_din%C3%A1mico) significa que los objetos en tiempo de ejecución (valores) tienen un tipo, a diferencia del tipado estático donde las variables tienen un tipo. A continuación un ejemplo de este concepto:

*# "variable" guarda un valor integer*

variable = 11

**print** variable, type(variable)

*# "variable" guarda un valor string*

variable = "activo"

**print** (variable), type(variable)

**Referencias:**

<https://uniwebsidad.com/libros/python/capitulo-2/estructuras-de-control-de-flujo>

<https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/>