Hoja de trabajo #6 – Temario A

Operador módulo (%)	*Devuelve el residuo que queda de dividir el primer operando entre el segundo.* *Ejemplo: 5%2 = 1*
Funciones booleanas	*Devuelve **True** o **False**.*
Operadores	*x == y #x es igual a y* *x != y #x es diferente de y* *x > y #x es mayor que y* *x < y #x es menor que y* *x >= y #x es mayor o igual que y* *x <= y #x es menor o igual que y*
Operadores lógicos	***and**, **or** y **not*** **var1** / **var2** / **var1 and var2** / **var1 or var2** True / False / **False** / **True** **True** / **True** / **True** / **True** **False** / **False** / **False** / **False** False / True / **False** / **True**
Condición múltiple	***if (condicion1):*** *# Instrucciones si se cumple la condicion1* ***elif (condicion2):*** *# Instrucciones si se cumple la condicion2* ***else:*** *# Instrucciones si no se cumple ninguna condicion*

Python nos permite indicar que deseamos que se repita un segmento de código de dos maneras distintas: mediante la sentencia **while** y mediante la sentencia **for**.

En lenguaje natural se pueden ver las repeticiones con **while** como: “Mientras se cumpla esta condición, repita estas acciones.” Y se utilizan cuando de antemano no conocemos el número de veces que las instrucciones deben repetirse; caso contrario con la sentencia **for**, en donde previamente se conoce este número.

Sintaxis de while	***while condicion:*** *# Instrucciones si se cumple la condición*
Sintaxis de for	***for elemento in lista:*** *# Instrucciones a a repetir por cada elemento*
Expresiones booleanas	*Devuelve true o false. Ejemplo: 5==5 devuelve True*

¡Recuerde!

La función **range(start, stop, step)** ES MUY PODEROSA. Es una función que nos permite crear **listas** a partir de una descripción simple (¿desde qué número inicia? ¿hasta qué número termina? ¿de qué tanto en qué tanto contamos?). Esta forma es mucho más limpia de crear una lista que tratar de escribir literalmente todos sus elementos.

Los parámetros que describen a la lista son:

- **start** indica desde qué número empieza la lista a generar. Si omitimos este parámetro, el valor por defecto es 0
- **stop** indica cuál es el primer número que ya no estará en la lista. Este parámetro **no se puede omitir**.
- **step** indica "de cuánto en cuánto" se generarán los números en la lista. Si omitimos este parámetro, el valor por defecto es 1.

Ejemplos:

```
mi_lista = range(7)
print mi_lista
>> [0, 1, 2, 3, 4, 5, 6]

mi_lista = range(2, 7)
print mi_lista
>> [2, 3, 4, 5, 6]

mi_lista = range(4, 15, 3)
print mi_lista
>> [4, 7, 10, 13]
```

El verdadero poder de **range** se desata cuando nos damos cuenta que su valor de retorno es una **lista**. Recordemos que un ciclo **for** nos permite recorrer los elementos de una lista, desde el primero hasta el último.

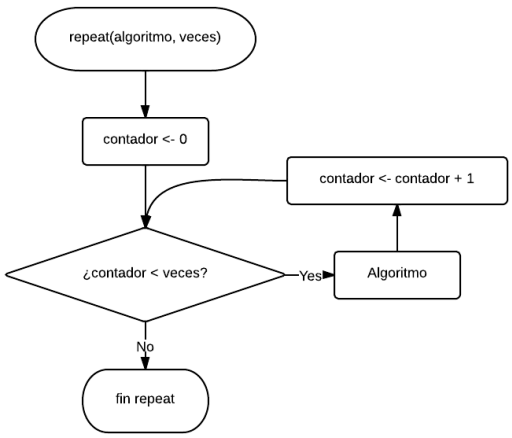
Entonces, esto nos da la pauta que la función **range** puede ser de mucha utilidad para la estructura de nuestros ciclos **for**.

Ejemplo: *Imprima en pantalla todos los múltiplos positivos de 3 menores que 20, e indique cuáles son pares.*

```
for multiplo in range(0, 20, 3):
     if multiplo % 2 == 0:
          print multiplo, "par"
     else:
          print multiplo, "impar"

>> 0 par
>> 3 impar
>> 6 par
>> 9 impar
>> 12 par
>> 15 impar
>> 18 par
```

Una última nota … ¿se recuerdan del comando **repeat** en RURPLE? Este comando repetía otro comando una cantidad determinada de veces. En Python podemos emular ese comportamiento a través de un **for** con la siguiente estructura:



Ejemplo:
De acuerdo con lo anterior, si quisiéramos hacer un algoritmo que imprima "Gool!" **3** veces, implementaríamos lo que sigue:

```
for i in range(3):
      print "Gool!"

>> Gool!
>> Gool!
>> Gool!
```

Todos sus programas deben tener un **encabezado** de comentarios con los siguientes datos:

- Nombre del archivo del script.
- Fecha de creación.
- Nombre de los autores y sus carnets.
- Propósito del programa.

PARA ENTREGAR EN CLASE

Ejercicio 1 (30 puntos)

La aerolínea Flay ha determinado que si más del 50% de sus meteorólogos de confianza pronostican tormenta, su próximo vuelo deberá ser cancelado. Flay le ha contratado para realizar un programa que facilite saber si es necesaria la cancelación de un vuelo, y lo primero que usted entregará será un prototipo que funcione con los pronósticos de exactamente 5 meteorólogos.

Cree una función llamada **habraTormenta** en un módulo llamado **pronosticos.py**. La función debe recibir una probabilidad numérica, y devolverá True si es mayor o igual a 80%, y False de lo contrario. Su función NO debe verificar que la probabilidad esté dentro de los límites adecuados (0 o 100).

Cree otra función en su módulo de pronósticos llamada **cancelarVuelo** que reciba las estimaciones de cinco meteorólogos. Use su función **habraTormenta** *dentro* de la función **cancelarVuelo** para determinar cuántos meteorólogos pronostican tormenta de los 5 recibidos. Si la cantidad de pronósticos de tormenta es mayor o igual a 3 su función devuelve 1; de lo contrario devuelve -1.

Finalmente cree un programa principal llamado **planificacionDeVuelo.py** que importe su módulo **pronosticos.py**. En este módulo ud. solicitará al usuario que ingrese las 5 estimaciones de los meteorólogos, verificando (aquí sí) que estén dentro de los límites adecuados (0 o 100). Si recibe alguna estimación que no esté dentro de los límites deberá interpretarla como 0. Con las estimaciones almacenadas, use su función **cancelarVuelo** para desplegar al usuario un mensaje que indique si será necesario hacer cancelar el vuelo o no. A continuación se presentan dos ejemplos de cómo se podría ver la ejecución de su trabajo:

```
Bienvenido al planificador de vuelos
Por favor ingrese las estimaciones de los 5 meteorólogos
Meteorólogo 1: 88
Meteorólogo 2: 76
Meteorólogo 3: 40
Meteorólogo 4: 91
Meteorólogo 5: 33
Cantidad de meteorólogos que pronostican tormenta: 2 de 5
No se cancela el vuelo
Bienvenido al planificador de vuelos
Por favor ingrese las estimaciones de los 5 meteorólogos
Meteorólogo 1: 88
Meteorólogo 2: 76
Meteorólogo 3: 40
Meteorólogo 4: 102
Meteorólogo 5: 33
Cantidad de meteorólogos que pronostican tormenta: 1 de 5
No se cancela el vuelo
```

Debe entregar en Blackboard:

- Módulo **pronosticos.py** con sus funciones: 15 puntos
  - Documentación: 1 punto
  - Función **habraTormenta**: 4 puntos
    - Documentación: 2 puntos
    - Ejecución correcta: 2 puntos
  - Función **cancelarVuelo**: 10 puntos
    - Documentación: 2.5 puntos
    - Ejecución correcta: 7.5 puntos
- Programa **planificacionDeVuelo.py:** 15 puntos
  - Documentación: 3 puntos
  - Interfaz amigable e intuitiva: 2 puntos
  - Validación de notas: 5 puntos
  - Ejecución correcta: 5 puntos

Universidad del Valle de Guatemala
CC2005-Algoritmos y Programación Básica
Semestre 2, 2016

PARA EMPEZAR EN CLASE Y ENTREGAR DE TAREA

Ejercicio 2 (20 puntos)

Escriba una función dentro de un módulo llamado **modulo1.py,** que reciba como parámetro una altura, y que **retorne** caracteres dibujando un triángulo de dicha altura. Luego escriba el programa **triangulo.py** que le pregunta al usuario por una altura, e imprime en pantalla el triángulo de esa altura regresado por su función en modulo1.py.

A continuación se muestra un ejemplo de cómo se podría ver su programa en ejecución:

```
>> Bienvenido a triangulo.py, empecemos!
>> Cual es la altura del triangulo?
>> 5
>> Ok, aca esta su triangulo!
>>
>>     *
>>    ***
>>   *****
>>  *******
>> *********
>>
>> Hasta luego!
```

Debe entregar en Blackboard:

- **modulo1.py**: 10 puntos
  - Documentación: 2 puntos
  - Función para dibujar triángulo: 8 puntos
    - Documentación: 4 puntos
    - Ejecución correcta: 4 puntos
- **triangulo.py**: 10 puntos
  - Documentación: 2 puntos
  - Programación defensiva: 2 puntos
  - Interfaz amigable e intuitiva: 2 puntos
  - Ejecución correcta: 4 puntos

Ejercicio 3 (40 puntos)

Se le ha contratado para programar una calculadora de tres operaciones: multiplicación, división y potenciación. Programe un módulo **funciones.py** que contenga funciones para cada una de estas operaciones. Estas funciones deben recibir dos operandos, y devolver un resultado numérico.

Luego diseñe un programa principal en un diagrama de flujo. Este diagrama debe pedir dos números al usuario, y luego desplegar un menú que permita elegir la operación a realizarse. Al concluir la operación el programa permitir al usuario realizar una operación nueva si así lo desea.

Su calculadora debe indicar en un mensaje cuando se ha elegido una opción inexistente. Finalmente programe lo que diseñó en su diagrama de flujo en un archivo llamado **calculadora.py** que importe su módulo de funciones, y que las use para implementar la calculadora

Su programa puede verse como el ejemplo a continuación:

```
Bienvenido a la calculadora
Por favor ingrese un número: 2
Por favor ingrese otro número: 5
¿Qué operación desea realizar?
1. Multiplicación
2. División
3. Potenciación
Ingrese opción: 3
El resultado es: 32
¿Desea realizar otra operación (s/n)? s
Bienvenido a la calculadora
Por favor ingrese un número: 10
Por favor ingrese otro número: 2
¿Qué operación desea realizar?
1. Multiplicación
2. División
3. Potenciación
Ingrese opción:9
Esa opción es inválida. Ingrese opción: 2
El resultado es: 5
¿Desea realizar otra operación (s/n)? n
```

Debe entregar en Blackboard:

- Diagrama de Flujo del programa principal: 5 puntos
- Módulo **funciones.py** con sus funciones: 10 puntos
    - Documentación: 5 puntos
    - Funciones correctamente implementadas: 5 puntos
- Módulo **calculadora.py**: 25 puntos

- Documentación: 5 puntos
- Interfaz amigable e intuitiva: 5 puntos
- Menú funcional: 10 puntos
- Ejecución correcta: 5 puntos

Reflexión semanal (10 puntos)

Responda las preguntas de su reflexión semana en Blackboard antes del siguiente día de clase.