PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

Laboratorio 4
Funciones y módulos

Segundo semestre, 2015

Objetivo

Este laboratorio es para que practiques con funciones y módulos en Python.

1. Funciones en Python

Una **función** es una sección de un programa que realiza una tarea específica de manera independiente del resto del programa. No se ejecuta por sí sola. Debe ser "llamada" o "invocada" desde el "programa principal" para que se ejecute. Esto permite escribir programas más modulares, agrupando instrucciones en tareas específicas. Al igual que en cualquier programa, en las funciones se distinguen tres partes: entrada, proceso (instrucciones de la función), salida.

- Entrada: argumentos o parámetros de la función.
- Proceso: conjunto de instrucciones de la función.
- Salida: argumento de retorno (opcional).

Ya conoces algunas funciones que haz usado en tus programas, como `print`, `input`, entre otras.

Veamos un ejemplo de una función propia, calculando el área de un cuadrado. Definimos una función llamada `area_cuadrado`, y el argumento de entrada será el `lado` del cuadrado. Dentro de la función se calculará el área del cuadrado (`lado**2`), y el valor de retorno será el área calculada. Todo esto en Python se define de la siguiente forma:

```
def area_cuadrado(lado) :
    area = lado**2
    return area
```

Recuerda que en tus programas en Python, debes respetar la estructura del lenguaje, como:

- Usar dos puntos (:) después de cada condición de un `if` y `elif`, y después de `else`; y después de la condición de `while`.

- Usar dos puntos (:) después de los argumentos/parámetros de entrada de la definición de una función con `def`.
- Indentar. IDLE y pycharm, entre otros, lo hacen automáticamente después de los dos puntos (:). La regla general es que la indentación sean cuatro espacios.
- Colocar el código dentro de cada control de flujo en el mismo nivel de indentación. Recuerda que también puedes anidarlos, es decir, poner un control de flujo dentro de una parte de otro control de flujo.

Las funciones se deben definir antes de ser invocadas (llamadas) por el programa principal.

Las funciones poseen varias características comunes:

- Puedes especificar varios argumentos/parámetros de entrada (separados por comas) o ninguno. Si la función no recibe parámetros de entrada, igual tienes que poner los paréntesis.
- Todas las variables declaradas dentro de la función son sólo visibles por la función. En el ejemplo anterior, la variable `area` no es visible (no está definida) fuera de la función. El valor que retorna con la instrucción `return` hace que se pueda conocer el valor de retorno, pero no el nombre de la variable de retorno.
- Los argumentos de entrada pueden tener un valor predeterminado (ver ejemplo resuelto). Los nombres de los argumento de entrada son usados como variables dentro de la función.
- Las funciones pueden devolver o retornar varios valores con la instrucción `return`. En este caso debes separar los valores con comas, por ejemplo al retornar dos valores mediante la instrucción `return area, perimetro`. Una función puede no devolver/retornar un valor. En este caso la instrucción `return` es opcional. Adicionalmente, la instrucción return (con o sin valores) puede aparecer en varias partes de la función.

Veamos un ejemplo más completo mediante ejercicio resuelto. Fíjate en este ejemplo que el parámetro `unidad` tiene un valor pre-determinado (o "por defecto").

1.1. Ejercicio resuelto

Escribe un programa que permita convertir de millas a kilómetros y viceversa, donde la conversión predeterminada es de kilómetros a millas. Prueba tu programa convirtiendo 1 milla, 2.5 km y 100 km.

Solución

```
def convertir(distancia, unidad="km_a_milla") :
    kms_por_milla = 1.609344
    if unidad == "km_a_milla" :
        dist_convertida = distancia*kms_por_milla
        return dist_convertida
    elif unidad == "milla_a_km" :
        dist_convertida = distancia/kms_por_milla
        return dist_convertida
```

```
    else :
        print("Conversion desconocida")
        return distancia

>>> d = convertir(1, "milla_a_km")
>>> print(d)
1.609344
>>> d = convertir(2.5, "km_a_milla")
>>> print(d)
1.5534279805933349
>>> d = convertir(100)
62.13711922373339
```

2. Módulos en Python

Los módulos en Python (conocidos como "librerías" en otros lenguajes) proveen una gran cantidad de funciones pre-definidas que puedes utilizar en tus programas. Esto incrementa enormemente la capacidad de Python para desarrollos profesionales, pudiendo por ejemplo, entregar toda la funcionalidad de MATLAB y otras herramientas para ciencias e ingeniería. Ya haz visto algunos, como el módulo `random` que se usó en el laboratorio anterior.

Hay dos formas de importar las funciones de un módulo:

1. Usar la instrucción

   ```
   import <modulo>
   ```

   reemplazando `<modulo>` con el nombre del módulo a importar. En este caso, las funciones son accesibles mediante el nombre del módulo seguido por un punto y el nombre del la función, por ejemplo en `random.randint(i,j)` del módulo `random`.

2. Importar funciones particulares mediante la instrucción

   ```
   from <modulo> import <funcion1>[, <funcion2>,...]
   ```

   donde los corchetes (`[]`) indican argumentos opcionales. Por ejemplo, si queremos sólo importar la función `randint` desde el módulo `random`, podemos ejecutar la instrucción `from random import randint`. En este caso, la función se usa simplemente como `randint(i,j)`.

En este laboratorio usaremos funciones de los módulos `random` y `math`.

3. Ejercicios

1. Escribe un programa que permita dibujar un árbol navideño. El follaje del árbol debe ser representado por `"*"`, mientras que para adornar el árbol puedes usar los símbolos `"$"`, `"#"`, `"@"` y el número `0`, entregando al programa el ancho del árbol, que debe ser un número impar. Tu programa debe realizar lo siguiente:

   - Consultar al usuario cuál será ancho. El valor pre-determinado del ancho debe ser 9, el que debe usarse si el usuario ingresa un ancho inválido.
   - Programar una función que imprima una línea del dibujo, dándole como entrada el ancho total y la cantidad de caracteres que desea que sean visibles al centro de la línea. Por ejemplo, para un árbol de ancho 5, la segunda línea debe estar compuesta por un espacio al inicio, tres caracteres al medio, y un espacio final. Para los caracteres distintos de espacio, la función debe imprimir 6 de cada 10 símbolos como follaje `"*"`, y 4 de cada 10 veces un símbolo entre (`$, #, @, 0`) en forma aleatoria.
   - La base del árbol debe ser de ancho 3, representada por dos líneas con la letra `I`.

   Un ejemplo de ejecución de tu programa:

```
>>>
*** Arbol de navidad ***
Ingresa el ancho: 11
     *
    0**
   *$**#
  0@***$*
 **$**#**0
@**0****#**
    III
    III
```

2. En el Laboratorio 2, encontramos qué tipo de soluciones tiene la ecuación cuadrática $ax^2 + bx + c = 0$:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

   Cuando la cantidad $\Delta = b^2 - 4ac$ es mayor que cero, x_1 y x_2 son números reales; si $\Delta = 0$, entonces $x_1 = x_2$, y corresponde a un número real; mientras que si $\Delta < 0$, tanto x_1 como x_2 son números complejos.

   a) Crea una función que te indique si los valores de `(a, b, c)` producen dos soluciones reales, una única solución o dos soluciones complejas. Si lo tienes disponible, puedes usar el código del Laboratorio 2, pero esta vez definirás una función.

   b) Crea una función que calcule las raíces de la ecuación cuadrática. Puedes importar el módulo `math`, y usar la función `math.sqrt` que calcula la raíz cuadrada.

   c) Usando $a > 0$, escribe una función que calcule, en forma iterativa, el valor de x que corresponde al mínimo de la función cuadrática. Para esto, debes ingresar como entrada los valores de `a`, `b` y `c`, más un cuarto valor que será la tolerancia de tu solución, que debe tener un valor predeterminado de 0.01.

3. Quieres organizar una fiesta en tu casa y necesitas determinar si tienes en tu refrigerador toda la comida necesaria para que la fiesta sea exitosa. Para esto, debes generar una función, que reciba el número de invitados, la cantidad de pizzas de 6 trozos cada una, y cantidad de latas de bebidas que tienes. Las condiciones que supones para preparar una fiesta exitosa y que todos los invitados no pasen hambre, es que por cada invitado debes tener al menos 4 latas de bebidas y más de 2 trozos de pizza. Programa lo siguiente:

   - Una función `refrigerador`, que de acuerdo al número de invitados, y la cantidad de pizza y bebidas, te pronostique si tu fiesta será exitosa o no, retornando `True` o `False`. En caso de que retorne `False`, debes imprimir en pantalla qué es lo que te falta para que la fiesta sea exitosa.
   - Usando la función `refrigerador`, encuentra una cantidad de bebidas y pizzas adecuada para una fiesta de 100 personas. Luego, simula tu fiesta, agregando un invitado a la vez, quien consume una cantidad aleatoria entre 2 a 5 bebidas, y entre 0 a 5 trozos de pizza. Compara el total de bebidas y pizza consumida por 100 invitados, y ve si tu pronóstico fue acertado (si la bebida y pizzas no se acabaron antes de que llegara el último invitado).