Fundamentos de Programación

Cuaderno de Trabajo 2

Ejercicios resueltos

1. **Programa una función que, recibiendo los 3 coeficientes a, b y c de una ecuación de segundo grado, calcule el valor del discriminante (valor del radicando) de la misma según la fórmula:**

$$raiz = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

SOLUCION PROPUESTA:

```
def discriminante(a, b, c):
""" float,float,float --> float
    OBJ: calcula el discriminante de la ecuación de segundo grado """
    return b**2-4*a*c

#Probador 4,5,1
print(discriminante(4,5,1))
```

2. **Programa una función que determine si un número entero es impar.**

SOLUCION PROPUESTA:

```
def es_impar(n):
   """ int --> bool
       OBJ: Determina si un entero es impar o no
       PRE: n es entero"""
    return (n % 2 != 0)     # alternativa n % 2 == 1

#Probador
print(es_impar(25), 'si ')   #Aprecia que este subp requiere 2 probadores
print(es_impar(24), 'no ')
```

3. **Escribe una función que calcule el impuesto que debe pagar un contribuyente a partir de sus ingresos anuales y el número de hijos. El impuesto a pagar es un tercio del ingreso imponible, siendo este último igual a los ingresos totales menos una deducción personal de 600€, a lo que se ha de añadir otra deducción de 60€ por hijo.**

SOLUCION PROPUESTA:

```
def impuesto (ingresos, num_hijos):
    """ float, int --> float
       OBJ:impuesto que debe pagar un contribuyente a
       partir de sus ingresos anuales y el número de hijos. Existe una
       deducción personal de 600 euros y una por hijo de 60eur/hijo
       PRE: num_hijos >= 0                              """
    DEDUCCION_PERSONAL = 600
    DEDUCCION_POR_HIJO = 60
    ingreso_imponible = ingresos - DEDUCCION_PERSONAL
    return (ingreso_imponible - DEDUCCION_POR_HIJO * num_hijos)/3

#Probador
print(impuesto(25000,2))
```

4. **Escribe un programa modularizado que, utilizando funciones, calcule el perímetro y el área de un círculo. Construye los respectivos probadores.**

SOLUCION PROPUESTA[1]:

```
def area_circulo(radio):
    """ float --> float
        OBJ: Calcula el area de un circulo
        PRE: radio >= 0 """
    import math
    return math.pi * radio**2

# Programa probador
radio=3.0
print ('área = ',area_circulo(radio), 'debe dar 28.274.. ')


def perimetro_circunferencia(radio):
    """ float --> float
        OBJ: Calcula el perímetro de una circunferencia
        PRE: radio >= 0 """
    import math
    return math.pi * radio * 2.0

#Programa probador
radio=3.0
print ('perímetro = ', perimetro_circunferencia(radio), 'dará 18.84... ')
```

Ejercicios propuestos

1. Implementa una función "fuerza" que retorne el valor de la magnitud física fuerza a partir de los valores de masa y aceleración recibidos como parámetros. Construye después un programa probador especificando los casos de prueba necesarios que muestre por pantalla el valor de la fuerza a partir de una masa y aceleración dadas.
2. El Barn es una unidad de superficie muy utilizada en la física de partículas, que equivale a 10^{-28} m². Para ilustrar su tamaño, ten en cuenta que un Barn es, aproximadamente, el área de la sección transversal del núcleo de un átomo de uranio. Programa dos funciones, una que permita convertir unidades en m² a Barns, y su inversa.
3. La *Tasa de Interés Efectivo Anual* (TIEA) es función de la tasa nominal anual (TNA) y del número (entero) de períodos de capitalización (del número de veces en el año que los intereses pasan a formar parte del capital) según la siguiente expresión: $TIEA=(1 + TNA/m)^{m-1}$, siendo m el número de periodos total de un año (es decir, 12 si hablamos de períodos mensuales, 4 si es trimestral, 2 si es semestral, etc.). Escribe una función que calcule el TIEA a partir del TNA y el número de períodos.
4. Escribe un programa modularizado que a partir de una hora en formato [hora, minutos y segundos] y utilizando una función que calcule el número total de segundos transcurridos desde la última medianoche, lo muestre posteriormente por pantalla.
5. Escribe una función que determine si un punto de coordenadas en 2D está o no sobre la circunferencia $x^2+y^2=1000$.

[1] Advierte que se ha importado el valor de pi de dos formas diferentes. La primera importa solamente el valor solicitado, mientras que la segunda importa la biblioteca completa. La primera es preferible.

6. El antiguo sistema anglosajón de unidades sigue en vigor en muchos lugares y su uso es frecuente en algunos contextos. Programa una función que determine el número de pintas que contiene una cierta cantidad de líquido expresada en mililitros, sabiendo que 1 pinta (pt) = 473,176473 ml.

7. La temperatura expresada en grados centígrados TC, se puede convertir a grados Fahrenheit (TF) mediante la siguiente fórmula: TF = 9*TC/5 + 32. Igualmente, es sabido que −273,15 °C corresponden con el 0 Kelvin. Escribe una función devuelva la temperatura en grados Farenheit y otra en Kelvin a partir de la temperatura en grados centígrados. Escribe un programa para probarlas que a partir de una temperatura en grados centígrados obtenga e imprima por pantalla la temperatura en Kelvin y Farenheit.

8. Escribe una función que a partir de las coordenadas 3D de dos puntos en el espacio en formato (x, y, z) calcule la distancia que hay entre dichos puntos. Prueba tu función.

9. Un número complejo es un número de la forma a+bi, donde a y b son números reales y el valor de i es $\sqrt{-1}$. Las cuatro operaciones aritméticas básicas sobre números complejos se definen como:
    - Suma: (a+bi)+(c+di)=(a+c)+(b+d)i
    - Resta: (a+bi)-(c+di)=(a-c)+(b-d)i
    - Producto: (a+bi)*(c+di)=(ac-bd)+(ad+bc)i
    - División: (a+bi)/(c+di) = ((ac+bd)/(c^2+d^2)) + ((bc-ad)/(c^2+d^2))i, suponiendo $c^2+d^2<>0$

   Programa funciones para cada una de las operaciones descritas con sus respectivos probadores y prepara una biblioteca con ellos. Construye los probadores que muestren por pantalla el resultado de las operaciones reseñadas. No utilices datos de tipo complejo, sino que deberás simularlos programando los complejos utilizando como coeficientes (a,b,c y d) valores de tipo real.

10. Una aplicación leerá de teclado los valores necesarios para mostrar en pantalla el área de un círculo, un cuadrado y un triángulo. No queremos implementar la aplicación completa aún, sólo preparar los subprogramas de que necesitaremos disponer para en un futuro construir la aplicación (funciones que leen de teclado números y los validan, por una parte, y funciones que hacen los cálculos de las distintas áreas, por otra). Nota: si lo necesitas, utiliza como aproximación de Π (pi) el valor de que proporciona la biblioteca "math" de Python.

11. El calendario gregoriano estableció (1582) que un año es bisiesto si es divisible entre 4, a menos que sea divisible entre 100. Sin embargo, si un año es divisible entre 100 y además es divisible entre 400, también resulta bisiesto. Implementa una función que determine si un año es bisiesto o no y escribe un probador para todos los posibles casos. Este probador debe mostrar en pantalla, por ejemplo, si has puesto una variable anno= 2015, una salida como la siguiente: "el año 2015 ¿es bisiesto?: False".

    Resuelve los problemas poco a poco, prueba primero si es divisible por 4. Cuando lo tengas ve añadiendo el resto de las condiciones.

    Error frecuente: considerar que necesitas una instrucción if. Los booleanos se pueden operar como el resto de los tipos.

12. Los profesores solemos utilizar dos decimales cuando trabajamos con las notas parciales de un alumno, pero en las actas se desea calificar con puntos enteros o medios (es decir: 0 / 0.5 / 1 / 1.5 /… / 9.5 /10). Combinando operaciones y funciones enteras y en coma flotante, encuentra una fórmula general que convierta una nota con dos decimales a la calificación correspondiente en el acta e implementa una función que la utilice para

convertir cualquier nota. Pista (aunque te parezca obvio, esta es la pista): 0.5 (que es lo que quiero conseguir) es la mitad de 1.

Tu propuesta debe funcionar para todos los casos de prueba que adjuntamos.

Nota original	**En acta**
8,89	9,0
8,50	8,5
8,45	8,5
8,24	8,0

Error frecuente: considerar que necesitas una instrucción if. No es necesario utilizarlo y te pedimos de hecho que no lo utilices.

13. Haz un subprograma que determine los días, horas, minutos y segundos que hay en una cantidad de segundos introducida por el usuario (desprecia las fracciones de segundo). Escribe un probador para el mismo.

14. Cuando construyamos programas para su comercialización mimaremos el aspecto de la entrada y la salida (lo que en conjunto se llama **interfaz de usuario**). Posiblemente pondremos al principio un rótulo, indicando al usuario qué aplicación está usando, rótulo que irá centrado en la pantalla y subrayado, dejando una línea en blanco por encima y otra por debajo. Así:

```
Aplicación de combinatoria
==========
```

Escribe el siguiente subprograma:

```
def centrarRotulo (rotulo):
    """string--> None
    OBJ: centra rótulo, subrayado con signos =, +línea encima y debajo
                    """
    tam=len(rotulo)
    lado=((72-tam)//2)-1  # 72 columnas suele ser ancho de la pantalla
    print()
    print(' '*lado,rotulo)
    print(' '*lado , '='*tam , end='\n')

#Probador
frase = 'Don Quijote de la Mancha'
centrarRotulo(frase)                          # distinto nombre
centrarRotulo('Cervantes')                    # constante
rotulo = 'El famoso hidalgo don Quijote de la Mancha' # mismo nombre
centrarRotulo(rotulo)
```

Ejecútalo en Pythontutor (http://www.pythontutor.com/visualize.html)

Nota: Este ejercicio pretende que visualices:

- Que la declaración de un subprograma no es ejecutable
- Que los argumentos reales no tienen por qué llamarse igual, ni diferente que los argumentos formales.
- De hecho, los argumentos reales de entrada pueden ser una constante
- Aprecia que cada pieza de código tiene su propia tabla de símbolos (*frames* en Python tutor). En un momento de la ejecución hay dos variables con el mismo nombre y distinto contenido.

15. Flexibiliza el subprograma **centrarRotulo** de modo que el signo con el que se subraye y el tamaño de la ventana puedan ser distintos en diferentes ejecuciones. *Nota: Este ejercicio pretende enseñarte que la forma de flexibilizar un subprograma es pasarle parámetros.*

16. La ley de los gases ideales indica que el volumen V (en litros) ocupado por n moles de un gas a presión P (en atmósferas) y temperatura T (en grados Kelvin), responde a la siguiente fórmula:

$$V = \mathrm{nRT/P}$$

siendo R la constante universal de los gases cuyo valor es 0,082 atmósferas litro/Kmol. Hagamos un subprograma que calcule el volumen de un gas a partir de su presión y temperatura y probémoslo.

17. Se dice que un gas está en condiciones ideales cuando está a 1 atmosfera de presión y 0ºC, es decir 273, 15ºK. Escribe una función que determine si un cierto gas se encuentra en condiciones ideales.

18. Siguiendo con lo anterior, implementa ahora un subprograma que calcule el volumen de un gas en condiciones ideales. ¡No olvides reutilizar tu esfuerzo no reescribiendo código que ya tienes de ejercicios anteriores, sino invocándolo! Por cierto, que tu nuevo subprograma debería tener tan solo una línea de código, tenlo en cuenta. Y no olvides probarlo.

19. Python permite dar valores por defecto en la definición de un subprograma. Documéntate y estudia cómo se hace esto. Para ello, ten en cuenta que la presión es mucho más estable que la temperatura. *Nota: Este ejercicio pretende mostrarte que con la implementación de valores por defecto ya no te harían falta los dos ejercicios anteriores.*

20. Una biblioteca es un archivo que agrupa funciones y otros elementos de código con una lógica o temática común. Así, existen bibliotecas gráficas, matemáticas, etc. Son importantes pues, al agrupar los subprogramas en bibliotecas, se facilita y fomenta su reutilización. Algunas vienen con una distribución del lenguaje puesto que son de uso común para muchas aplicaciones (piensa en la biblioteca "math" que ya habrás seguramente utilizado) pero también es posible construirnos nuestras propias bibliotecas. En este ejercicio queremos mostrar lo sencillo que resulta reutilizar código, creando las siguientes bibliotecas para tu uso personal:

    a. Una biblioteca **bib_interfaz** de funciones de interfaz con la función **centrarRotulo** del ejercicio 14, así como los subprogramas que leen datos numéricos de teclado y que permiten construir aplicaciones como la del ejercicio 10. Organiza todo de modo que resulte fácil reutilizarlo en el futuro, pues esta biblioteca la iremos ampliando en adelante.
    b. Otra biblioteca **bib_conversiones** con funciones de conversión de unidades que contenga todas las funciones implementadas en los ejercicios 2, 6 y 7.
    c. Una tercera **bib_geometria** que agrupe funciones relacionadas con la geometría, por ejemplo las de los ejercicios 5, 8 y 10 (al menos).