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€ y 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.

```
def fuerza (m, a):
    """float*2-->float
    OBJ: fuerza de una masa m con aceleración a
    PRE: masa>0    """ #La a puede ser negativa
    return m*a

#PROBADOR
print(fuerza(3.0,9.8))  #rpr no hay más casos de prueba
                        #es un probador--> no me preocupo del formato
```

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.

[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. El uso de una u otra depende del número de recursos que necesites.

```
def en_Barn (m2):
    """float-->float
    OBJ: en Barn los metros cuadrados m2
    PRE: m2>0"""
    return m2*1e+28
#PROBADOR
print(en_Barn(2.0))

def en_m2 (Barn):
    """float-->float
    OBJ: en metros cuadrados m2 los Barn
    PRE: m2>0"""
    return Barn*1e-28
#PROBADOR
print(en_m2(2.0e-28))
```

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.

```
def TIEA(TNA, m):
    """float,int-->float
    OBJ: Tasa de Interés Efectivo Anual de la Tasa Nominal Anual (TNA)
    y del número de periodosde capitalización m
    PRE: m>0   """   #la tasa TNA puede ser negativa
    return (1+TNA/m)**(m-1)

#PROBADOR
print(TIEA (1.0, 12))
```

4. Escribe un programa modularizado que, a partir de una hora en formato [hora, minutos y segundos] solicitados al usuario, y utilizando una función que calcule el número total de segundos transcurridos desde la última medianoche, lo muestre posteriormente por pantalla.

```
def seg_transcurridos(hh,mm,ss):
    """int*3-->float
    OBJ: segundos total transcurridos a las hh horas, mm minutos, ss segundos
         desde media noche
    PRE:0<=hh<=23 y 0<=mm,ss<=60 o h==24 y mm==ss==0 """
    return (hh*60+mm)*60+ss

'''
#PROBADOR
print(en_seg(1,1,1), 3661)
'''
try:
    horas=int(input('horas: '))
    minutos=int(input('minutos: '))
    segundos= int(input('segundos: '))
except:
    print('introduzca los datos en cifras')
else:
    print('A las',horas,':',minutos,':',segundos,'han transcurrido ',
          seg_transcurridos(horas,minutos, segundos), 'segundos')
```

5. Escribe una función que determine si un punto de coordenadas en 2D está o no sobre la circunferencia $x^2+y^2=1000$.

```
def esta_dentro_circunf(x,y):
    """float*2-->bool
    el punto formado por x,y esta dentro de circunferencia x2+y2=1000"
""
    return x**2+y**2<=1000

#PROBADOR
print (esta_dentro_circunf(1.0,2.0),'si')
print (esta_dentro_circunf(100.0,200.0),'no')
```

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.

```
PINTA= 473.176473  #miliLitros
def en_mL(pintas):
     """float-->float
     OBJ: pintas en mililitros
     PRE: pintas >=0, PINTA definido"""
     return pitas *PINTA

 #PROBADOR
 print(en_mL(1.0))
```

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.

```
def en_centigrados(t_fahrenheit):
    """float --> float
    OBJ: temperatura t_fahrenheit en Fahrenheit en grados centígrados
    PRE: t_fahreneit>=-459,67°F """

    return (t_fahrenheit-32.0)*5.0/9.0

"""PROBADOR"""
print(en_centigrados(100.0)) #Caso de número mayor que 0
print(en_centigrados(0.0)) #Caso de número igual a 0
```

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.

```
def distancia(x1,y1,z1,x2,y2,z2):
    """float*6-->float
    OBJ: distancia entre 2 puntos (x,y,z)"""
    import math
```

```
    return math.sqrt((x2-x1)**2+ (y2-y1)**2+ (z2-z1)**2)
'''
#PROBADOR
print(distancia(0,0,0,1,1,1), 'debe dar')
'''
```

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 cada uno de 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.

```
def es_bisiesto(anno):
    """int-->bool
    OBJ: True si es bisiesto anno
    PRE: año>=1582                                    """
    return anno%4==0 and anno%100!=0 or anno%400==0
'''
#PROBADOR
print(es_bisiesto(2001), 'no, 2001 no es multiplo de 4')
print(es_bisiesto(2004), 'si, 2004 es multiplo de 4, pero no de 100')
print(es_bisiesto(2100), 'no, 2100 es multiplo de 100, pero no de 400'
)
print(es_bisiesto(2000), 'si,2000 es multiplo de 400')
'''
```

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.

```
def nota_acta (nota):
    """float-->float
    OBJ: Nota para  acta, redondeada a 0,5
    PRE: 0<=nota<=10"""
```

```
    return(nota*2+0.5)//1/2

#PROBADOR
print(nota acta(8.89), 9.0)
print(nota_acta(8.5), 8.0)
print(nota_acta(8.45), 8.5)
print(nota_acta(8.24), 8.0)
```

Solución por pasos:

```
def nota_acta (nota):
    """float-->float
    OBJ: Nota para  acta, redondeada a 0,5
    PRE: 0<=nota<=10"""
    nota*=2
    nota+=0.5
    nota//=1
    return nota/2
```

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.

```
def en_ddhhmmss(seg):
    """float--> int*4
    OBJ: dias,horas, minutos y segundos que hay en seg
    PRE: seg>=0"""
    dd,seg=divmod(int(seg),24*3600)
    hh,seg=divmod(seg,3600)
    mm,seg=divmod(seg,60)
    return dd,hh,mm,seg

#PROBADOR
print(en_ddhhmmss((2*24+1)*3600+3*60+2.5), 'dará 2d, 1h,3m,2s')
```

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.*

```
def centrarRotulo (rotulo, signo,tam_p):
    """string--> None
    OBJ: centra rótulo, subrayado con signo,
         en pantalla de tamaño tam_p
     +línea encima y debajo
     """
    tam=len(rotulo)
    num_blancos=((tam_p-tam)//2)-1
    print()
    print(' '*num_blancos,rotulo)
    print(' '*num_blancos, 'signo*tam , end='\n')

#Probador
rotulo = 'El famoso hidalgo don Quijote de la Mancha' # mismo nombre
centrarRotulo(rotulo)
```

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.

```
#16
"""**********      Biblioteca Gases Ideales      **********"""

R = 0.082   #en atm·litro/Kmol
P_CN=1.0 #atm
T-CN=273.15  #Kelvin


def vol(n,T,P):
    """float,float,float --> float
    OBJ: volumen de n moles de un gas ideal a una temperatura T y a un
a presión P.
    PRE: n>0, T>0, P>0, n en moles, T en Kelvin, P en atmósferas. R de
finida"""
    return n*R*T/P

'''
"""PROBADOR"""
print('V=', vol(1,230,2), 'litros')
'''
#17
def esta_CN(T,P):
    """float,float --> bool
        OBJ: comprobar si un gas se encuentra en condiciones ideales.
        PRE:T>0, P>0, T en Kelvin, P en atm. P_CN y T_CN definidas"""
    return P == P_CN and T == T_CN

'''
"""PROBADOR"""
print('¿Condiciones ideales?', esta_CN(273.15, 1)) #Probador de true
print('¿Condiciones ideales?', esta_CN(1, 1))   #Probador de false
'''
#18
def vol_CN(n):
    """float --> float
    OBJ: volumen de n moles de un gas ideal en condiciones normales.
    PRE: n>0, n en moles."""
    return vol(n,T_CN, P_NC)
'''

"""Probador"""
print('Volumen=', vol_CN(1))

'''
def vol(n,T=T_NC,P=P_NC):
    """float,float,float --> float
    OBJ: volumen de n moles de un gas ideal a una temperatura T y a un
a presión P.
```

```
    PRE: n>0, T>0, P>0, n en moles, T en Kelvin, P en atmósferas. R de
finida"""
    return n*R*T/P
'''
"""PROBADOR"""
print('V=', vol(1,230,2), 'litros')
print('V=', vol(1,230), 'litros')
print('V=', vol(1), 'litros')
'''
```

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.

    Incluida en biblioteca (ejerc anterior)

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.

    Incluida en biblioteca (ejerc anterior)

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.*

    Incluida en biblioteca (ejerc anterior)