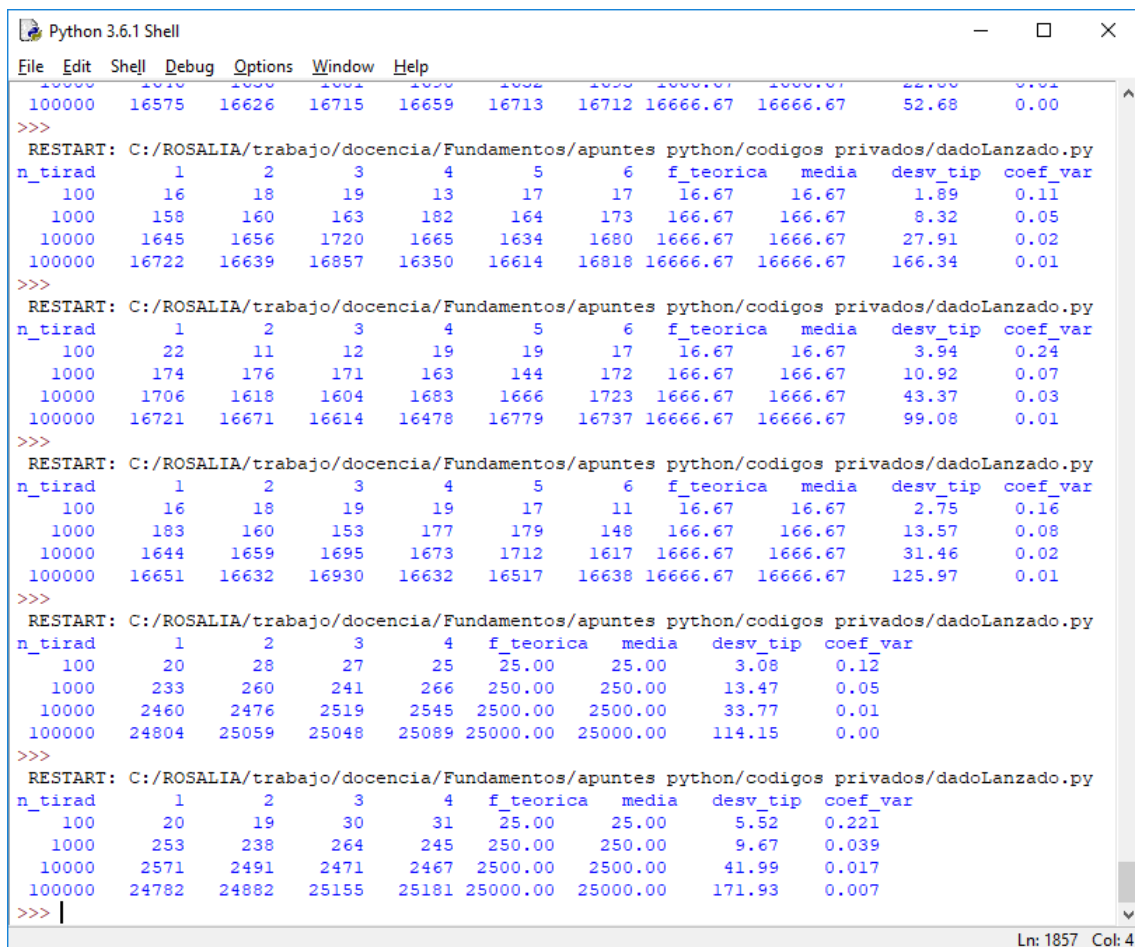


Actividad 2.2: EVALUCIÓN DE LA BONDAD DE random.randin

Enunciado: Por propia definición de aleatoriedad (aleatorio=solo dependencia del azar, sin patrón de comportamiento), es conceptualmente imposible definir un algoritmo (algoritmo=patrón de comportamiento) que genere números aleatorios. Sin embargo, es fundamental disponer de números aleatorios para la resolución de algunos problemas, por ejemplo, simulaciones. La biblioteca `random` de Python dispone de un generador de números pseudoaleatorios (pseudo quiere decir que parece, pero que no es): la función `randint(ini, fin)` devuelve un entero comprendido entre el valor "ini" y el valor "fin" (ambos incluidos), de modo que la instrucción:
`sale=randint(1, 6)`

simula el lanzamiento de un dado de 6 caras.

Con el fin de probar la calidad de este generador de números aleatorios, vamos a construir un programa que simule 100, 1.000, 10.000 y 100.000 tiradas de un dado de n caras, registrando la frecuencia de aparición de cada una de las posibilidades. El programa pintará una tabla con columnas correspondientes a número de tiradas, frecuencia encontrada para cada caso posible, frecuencia teórica, frecuencia media, desviación típica y coeficiente de variación respectivamente; y una fila para cada una de las poblaciones generadas. Utilizaremos la biblioteca de estadística que generamos en la entrega anterior. Os subo mi propuesta para que partamos todos de la misma y corregiré contra ella .



```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help

100000 16575 16626 16715 16659 16713 16712 16666.67 16666.67 52.68 0.00
>>>
RESTART: C:/ROSALIA/trabajo/docencia/Fundamentos/apuntes python/codigos privados/dadoLanzado.py
n_tirad 1 2 3 4 5 6 f_teorica media desv_tip coef_var
100 16 18 19 13 17 17 16.67 16.67 1.89 0.11
1000 158 160 163 182 164 173 166.67 166.67 8.32 0.05
10000 1645 1656 1720 1665 1634 1680 1666.67 1666.67 27.91 0.02
100000 16722 16639 16857 16350 16614 16818 16666.67 16666.67 166.34 0.01
>>>
RESTART: C:/ROSALIA/trabajo/docencia/Fundamentos/apuntes python/codigos privados/dadoLanzado.py
n_tirad 1 2 3 4 5 6 f_teorica media desv_tip coef_var
100 22 11 12 19 19 17 16.67 16.67 3.94 0.24
1000 174 176 171 163 144 172 166.67 166.67 10.92 0.07
10000 1706 1618 1604 1683 1666 1723 1666.67 1666.67 43.37 0.03
100000 16721 16671 16614 16478 16779 16737 16666.67 16666.67 99.08 0.01
>>>
RESTART: C:/ROSALIA/trabajo/docencia/Fundamentos/apuntes python/codigos privados/dadoLanzado.py
n_tirad 1 2 3 4 5 6 f_teorica media desv_tip coef_var
100 16 18 19 19 17 11 16.67 16.67 2.75 0.16
1000 183 160 153 177 179 148 166.67 166.67 13.57 0.08
10000 1644 1659 1695 1673 1712 1617 1666.67 1666.67 31.46 0.02
100000 16651 16632 16930 16632 16517 16638 16666.67 16666.67 125.97 0.01
>>>
RESTART: C:/ROSALIA/trabajo/docencia/Fundamentos/apuntes python/codigos privados/dadoLanzado.py
n_tirad 1 2 3 4 f_teorica media desv_tip coef_var
100 20 28 27 25 25.00 25.00 3.08 0.12
1000 233 260 241 266 250.00 250.00 13.47 0.05
10000 2460 2476 2519 2545 2500.00 2500.00 33.77 0.01
100000 24804 25059 25048 25089 25000.00 25000.00 114.15 0.00
>>>
RESTART: C:/ROSALIA/trabajo/docencia/Fundamentos/apuntes python/codigos privados/dadoLanzado.py
n_tirad 1 2 3 4 f_teorica media desv_tip coef_var
100 20 19 30 31 25.00 25.00 5.52 0.221
1000 253 238 264 245 250.00 250.00 9.67 0.039
10000 2571 2491 2471 2467 2500.00 2500.00 41.99 0.017
100000 24782 24882 25155 25181 25000.00 25000.00 171.93 0.007
>>> |
```

Ln: 1857 Col: 4

Figura 1: Ejemplos de ejecución

La Figura 1 recoge la salida de tres ejecuciones del programa con un supuesto dado de 6 caras y 2 salidas con un dado de 4 caras.

Análisis de los resultados de la evaluación de `randint`:

1) Puesto que `randint` intenta parecer aleatorio, a diferencia de lo que suele ocurrir en programación, la salida de dos ejecuciones con los mismos datos no son idénticas ¿Inhabilita esta falta de reproductibilidad de las simulaciones aleatorias usarlas como experimentos válidos en el método científico de investigación? o por el contrario, ¿aunque los datos no sean idénticos, la reproductibilidad de las tendencias de los datos en sucesivas ejecuciones, nos permitirán extraer conocimientos científicos?

2) La frecuencia media de aparición de caras es coincidente con la frecuencia teórica, pero eso es una consecuencia directa de las fórmulas matemáticas empleadas y no nos aporta conocimientos sobre la bondad de `randint`.

3) En todas las ejecuciones, el coeficiente de variación va disminuyendo cuando aumenta el tamaño de la muestra, lo cual también ocurriría con poblaciones muestrales realizadas con un dado real.

4) De hecho, el error estadístico (error debido al tamaño de la muestra) se comporta como $1/\sqrt{n}$ donde n es el tamaño de la muestra ¿quieres comprobar cómo se comporta en esta simulación?

5) En este ejercicio se conocían los resultados teóricos esperados, por lo que podemos evaluar la bondad de `randint` como generador de números aleatorios, pero adicionalmente, cuando desee usar este generador para extraer conclusiones sobre fenómenos cuya solución desconozca, este ejercicio puede orientarnos en la determinación del tamaño de la muestra adecuado, en función de la precisión (tanto_por_uno de error aceptable) con que se necesite extraer las conclusiones (una muestra de 1000, permitiría una aproximación hasta la primera cifra decimal, ya que los coeficientes de variación, en todas las ejecuciones son menores de 0,0X para ese tamaño de muestra).

Recomendaciones para el desarrollo de la práctica:

Este ejercicio tiene una componente de diseño de datos que puede que te cueste ver, al ser tu primer diseño en solitario. Pero piensa en los ejemplos que hemos visto en clase y busca paralelismos. La codificación de los procesos, por supuesto depende íntimamente de la estructura que hayas dado a los datos. De modo, que no empieces a codificar hasta que no hayas acordado firmemente el diseño de los datos.

Como siempre aconsejamos: “toma tierra”, poniéndote un ejemplo concreto y resolviéndolo a mano, para no hacer algoritmos más ineficientes que a mano. Realizar el recuento de unas elecciones es algo equivalente. Sacar una papeleta de la urna es equivalente a tirar un dado. Piensa que eres el presidente de una mesa electoral y hay tantos partidos como caras de dados.

Cuando hayas decidido la estructura adecuada para los datos, inicializa los contadores de frecuencia. Como ya sabes “lanzar un dado”, ahora, no debe resultarte muy complejo calcular la frecuencia de aparición de cada cara.

En cuanto a generar la tabla de salida, como siempre, sugiero que resuelvas los problemas de forma escalonada, por ejemplo, primero haz “una tabla” con solo 2 de las columnas y una sola fila (para muestra =100). Añade columnas, una a una, hasta obtener la primera fila completa. Solo entonces, plantéate cómo generalizar para que la muestra de esa primera columna sea de tamaño “n” en vez de 100.

Ahora ya sabes escribir una cualquiera de las líneas ¿Cómo consigues generar 1000 a partir de 100? Y ¿10000 a partir de 1000?, etc. Ahora centraos en encontrar una fórmula que generalice el tamaño de la muestra de una fila al de la siguiente. ¿Cómo consigues que se repita 4 veces cualquier algoritmo que sabes hacer una vez?

Estimación de tiempos:

Mi propuesta de solución tiene menos de 40 líneas incluidos comentarios. El proceso no es muy complejo, ni de largo desarrollo. El tiempo en escribir las líneas, corregir los errores de sintaxis y probar no serían más de dos horas, pero, teniendo en cuenta que es la primera vez que te enfrentas a ello, es posible que te atranques en varios puntos y el tiempo en salir de un atranque es muy variable. Pártete la cara con el atasco una o dos horas, a lo sumo una tarde, pero pide ayuda transcurrido un tiempo prudencial.

Entrega y calificación:

El ejercicio se realizará en equipos de exactamente **2 personas**, se entregará en el enlace de la actividad, en la plataforma aula virtual de la uah, subiendo un único archivo con la extensión “.py”. Cada miembro del equipo deberá subir una copia idéntica a la plataforma, firmando así su autoría. El equipo debe reunirse para discutir el diseño de datos y la descomposición funcional, a continuación, lo realizarán cada uno de forma individual, reuniéndose nuevamente, para acordar, de forma conjunta, la versión definitiva a subir.

El nombre del archivo será:

“A2.2_grupo-apellidos1_nombre1_y_apellidos2_nombre_2” sustituyendo “apellidos” y nombre por los de los alumnos y grupo con IC o II.

Un archivo que dé errores de tiempo de traducción o de tiempo de ejecución obtendrá la calificación de 0 directamente. **No se corregirán** archivos de más de 2 personas, y a lo sumo 1 grupo de 1 persona. Pero, si se entregaran más de un grupo con menos de 2 personas **no se corregirá** ninguno de ellos. Tenéis el foro y otros medios para buscar alianzas.

La puntuación máxima de esta fase de la actividad es 6.1. Los 3.9 puntos restantes corresponden a la primera fase.

ADELANTE CON VUESTRO PROGRESO

Conducción más detallada hacia la solución:

El hilo de dudas de la actividad 2.2 me sugiera que quizá algunos andáis despistados. El trabajo a realizar es sencillo, por si os sirve de pista, **mi propuesta de solución tiene 22 líneas de código**, incluida documentación.

Ya conocéis el ZEN de Python: Si es complejo, está mal. → si te está resultando difícil es que no vas por buen camino. Párate y dale otra pensada, abordando las complejidades de una en una, hasta simplificarlas

1) Organiza los datos que debe manejar el programa

A ver..., si respondiendo a estas preguntas concretas lo que necesitas

1A) ¿En qué tipo de datos puedo almacenar 1 frecuencia de aparición? (vamos poco a poco, te he preguntado solo 1 frecuencia de aparición... ponte ejemplos del mundo real... dilo en voz alta, y escuchate... Ej: " la cara 4 ha salido X veces". El tipo de X es el que buscas, digamos que has decidido que es T).

1B) Ya sabes de que tipo es 1 contador ¿cuantos contadores necesitas?

¿En qué tipo de datos almacenarías varios objetos del mismo tipo? ¿En qué orden los colocarías?

1C) Para evaluar la bondad de randint ¿necesito saber qué cara salió en cada tirada en concreto (=100.000 enteros en la más larga de las pruebas)? o por el contrario, tengo suficiente con saber la frecuencia de cada una de las caras? si hay n caras (digamos n=9, por poner un ejemplo inusual), quiere decir que solo necesitas n posiciones de tipo T. ¿vas a gastar 100000 enteros si solo necesitas 9T? Si aún no ha quedado claro, más abajo hay un ejercicio de recuento de urnas.

y... Como suele ocurrir **cuando organizas adecuadamente los datos ... el proceso es sencillo:**

2) Aun no has lanzado ningún dado... ¿cuánto valen ahora cada uno de los contadores de frecuencia? Consigue una variable que contenga los num-caras contadores inicializados a cero.

3) Ahora tira 1 dado (solo una vez, vamos de poco en poco). Supongamos que te ha salido la cara C (digamos C=5). ¿recuerdas dónde has puesto el contador de la cara 5 en la variable que has creado en el paso 2? Añádele 1 al contador correspondiente, tuviera lo que tuviera, eso es fácil ¿verdad?

4) Ahora ya sabes manejar los contadores de frecuencia cuando lanzas dado 1 vez... ¿sabrás hacer lo mismo (tirar el dado y aumentar el contador) 100 veces? (te he preguntado 100, en concreto, pótelo de momento fácil) ¿sabes hacerlo para 100? ¿a que sí?

5) Ahora quiero pintar la salida. La tabla tendrá solo 2 filas, una del rótulo y otra de la evaluación de 100 lanzamientos del dado... y me encuentro con que el número de columnas que quiero escribir es variable.... ¡¡¡¡Ni siquiera se escribir la cabecera de la tabla!!!! Porque no sé cuántas columnas tiene. Aparca aquí tu problema general, y resuelve este pequeño

problema. Defino el subproblema: "Quiero escribir 1 (ocupando 8 caracteres, ajustado a la derecha), coma después 2 (idem), 3(idem)num_caras.

5A) ¿sabes escribir un solo número (cualquiera de ellos, digamos el $i=3$, pero el i , no el 3) ocupando 8 caracteres, ajustado a la derecha? Seguro que sabes. Hazlo con patrones de formato.

5B) Y... ¿Cómo podrías repetir este proceso, sin que salte de línea, un número variable de veces, digamos num_caras veces?

5C) Ahora, dedícate a hacer florituras: subraya cada uno de esas num_caras cabeceras de columna que has creado ¿sabrás? Seguro que sí. Recuerda que todos los tipos secuencia de Python (tuplas, listas y cadenas) disponen del operador `sec*int-->sec` que repite int veces la sec.

5D) Ahora, componer tu rótulo añadiendo las columnas que eran fijas, por delante y por detrás, está chupado. Es el momento de hacerlo.

5E) Y llegados a este punto, escribir la primera fila de datos a esta tabla también está chupado. Hazlo ahora.

6) En el paso 4 has aprendido a sacar los datos de frecuencia para una fila de la tabla con un número concreto de tiradas.... y en el paso 5 a pintar la fila... pero cada una de las filas tiene un número de tiradas diferente ¿Cómo generalizar este proceso?

6A) Primer paso es fácil... sustituye el 100, por una variable que iguale a 100, ¿por ejemplo "t" de tiradas?

6B) El segundo no lo es tanto, por ello, olvídate del mundo mundial.... tu único problema ahora es conseguir un programa que genere (y pinte, para que compruebes que lo has hecho bien), los números 100,1000,10.000,100.000 ¿qué has de hacerle al núm 100, para que se convierta en 1000? ¿y al 1000 para que llegue a 10.000?. Generaliza, es decir, dame una expresión que calcule el siguiente número, a partir del anterior ... ¿de qué valor debes partir para que la primera vez ejecute con 100? ¡Tampoco era tan difícil ¿no?!

6C) En 6A) aprendiste a escribir la línea con "t" lanzamientos. Quieres escribir 5 líneas, y en 6B aprendiste a generar la siguiente t en función de la anterior ¡¡¡Quien sabe escribir una línea sabe escribir un número variable de ellas!!! ¿no?

... ¿A QUE SI QUE LO SABIAS HACER?... Los procesos son sencillos cuando las estructuras de datos son adecuadas y ... si aun así son complejos, abordo cada una de las complejidades de forma individual y voy componiendo, poco a poco la solución, integrando las subsoluciones.

Un pasito cada vez se anda el Camino ¡¡¡ADELANTE PEREGRINOS!!!

Actividad 1.

12 de febrero 2018

Empezamos muy suavcito y con algo muy parecido a lo que ya hemos trabajado. Espero que absolutamente todos tengamos éxito en esta actividad.

La distancia entre dos puntos cualesquiera del espacio tridimensional viene dada por la raíz cuadrada de la suma de los cuadrados de la diferencia entre cada una de sus tres coordenadas. Puesto que la distancia puede ser un valor relevante en muy diversas aplicaciones lo vamos a subprogramar.

1) (2,5 puntos) Haz un subprograma que recibe las 6 coordenadas y devuelve la distancia. Sigue el dodecálogo de la subprogramación (en las transparencias de clase tienes un resumen). Todo es importante en programación (y en la vida real), elije adecuadamente los nombres tanto del subprograma como de los argumentos, elije cuidadosamente el orden de éstos, documenta, codifica, comprueba.

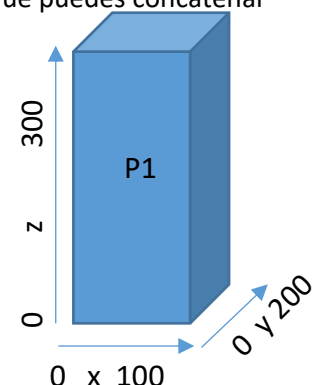
Recuerda que la biblioteca `math` de Python contiene un subprograma que calcula la raíz cuadrada. Reúsalo.

2) (2 puntos) Es muy frecuente trabajar en superficie, lo cual es algo más sencillo que trabajar en el espacio. Una de las grandes ventajas de la subprogramación es el reuso. ¡¡¡No olvides reusar tu esfuerzo!!! Tu nuevo subprograma tendrá una línea de código ejecutable y ni un solo operador. Escribe el probador, con los casos de prueba adecuados.

3) (2,5 puntos) Haz un subprograma que pida a un usuario las coordenadas en centímetros de un punto confinado dentro de una habitación de 1*2*3 metros (como se muestra en la figura).

El subprograma mostrará al usuario el nombre del punto del que está pidiendo la coordenada, para lo cual, tiene un argumento de entrada. Para codificar este subprograma reusa la función `realPedido` que se encuentra en la biblioteca `libInterfaz` y que pide un valor real comprendido entre un máximo y un mínimo hasta que el usuario introduzca un valor adecuado. Usa la función de la biblioteca, no la copies en tu programa. Recuerda el orden de las secciones de un subprograma → importa en el sitio adecuado. Recuerda también que puedes concatenar cadenas de caracteres con el operador `+`. Prueba tu subprograma.

```
EJEMPLO DE SALIDA DEL PROBADOR
P1.x =50.0
P1.y =100.0
P1. =me equivoco
debe ser un real. P1.z =150.0
el punto es (50.0, 100.0, 150.0)
```



4) (3 puntos) Añade ahora un cuerpo ejecutivo que pida al usuario dos puntos en el espacio e imprima los dos puntos (redondeando a 2 cifras decimales cada coordenada, usando la escritura

con formato tipo c de Python) y la distancia entre ellos **también redondeada**. Un ejemplo de salida sería:

```
la distancia entre (0.00,0.00,0.00) y (1.00,1.00,1.00) es 1,73 metros
```

5) En la documentación de tu archivo indica el tiempo consumido.

Objetivo: En el proceso de aprendizaje es más importante la autoevaluación que la calificación. En este contexto, es importante tener retroalimentación temprana que nos brinde la oportunidad de tomar medidas correctoras. Por ello, pretendo sugeriros mejoras personalizadas sobre vuestro código, y sois muchos. Recuerda respetar el tiempo del revisor (en este caso yo). No copies pues haría inútil este esfuerzo. Si te atrancas, pide ayuda bien al profesor, a compañeros, a un profesional, o en bibliografía. La idea no es “que me resuelvan el problema”, sino “entender qué me faltaba” y en tu documento, comenta brevemente la dificultad encontrada y la ayuda recibida y agradece explícitamente a la persona (con nombre y apellidos) que te ha ayudado o la fuente que has empleado, como debe hacerse en un ejercicio profesional deontológico (esto se llama “reconocimiento o **créditos de autoría**”).

Estimación de tiempos: Si habéis estudiado al día, y realizado los ejercicios solicitados las semanas anteriores, este ejercicio debería consumiros entre 1 y 8 horas (personas muy válidas tienen ritmos muy diferentes, pero principalmente depende de lo que hayáis practicado estas semanas).

Entrega y calificación: El ejercicio se entregará en la actividad, en la plataforma aula virtual de la uah. Subiendo un único archivo con la extensión “.py”. El nombre del archivo será: “A1_apellidos_nombre” sustituyendo “apellidos” y nombre por los del alumno.

Un archivo que de errores de tiempo de traducción o de tiempo de ejecución obtendrá la calificación de 0 directamente. La calificación de cada apartado está indicada entre paréntesis.

¡¡¡Deseo BUENA CAZA a todos!!!

Contrasta tu respuesta con la propuesta atendiendo a tus errores,

PROPUESTA DE SOLUCIÓN

```
"""Geometria:  Propuesta_solución_rpr para Actividad_1"""
#45 líneas de código, que caben en pantalla de 72 caracteres

from libInterfaz import realPedido          #la sección de interfaz va aqui

def distancia_3D(x1,y1,z1,x2,y2,z2):  #identificadores cortos y significativos
    """ float,float,float,float,float,float-->float
    OBJ: Distancia entre 2 puntos(x,y,z) """
    from math import sqrt              #donde necesito y solo lo que necesito
    return sqrt((x2-x1)**2 + (y2-y1)**2 + (z2-z1)**2)

'''#PROBADOR
print('dist', distancia_3D(0.0,0.0,0.0,    1.0,-1.0,1.1),'debe dar 1,79')
```

```

# 1 solo caso de prueba, xq no hay nada especial. Y sin florituras'''

def distancia_2D(x1,y1,x2,y2):          #nombre coherente con anterior fun _3D
    """ float, float,float,float-->float
    OBJ: Distancia entre 2 puntos(x,y) """
    return distancia_3D(x1,y1,0.0,x2,y2,0.0) #operaciones internas FLOAT

'''#PROBADOR
print('dist', distancia_2D(0.0,0.0, 1.0,-1.1),'debe dar 1,49')
# algunos alumnos han vuelto a probar el 3D con z=0'''

def punto_pedido(msg):                  #igualmente podíanos haber flexibilizado limites
    """--> float,float,float,
    OBJ: punto pedido a usuario
    PRE: realPedido definido"""
    x=realPedido(0.0,100.0,msg+'.x =')
    y=realPedido(0.0,200.0,msg+'.y =')
    z=realPedido(0.0,300.0,msg+'.z =')

    return x,y,z                       #no agrupo, pues luego lo necesito
    #si agrupas punto=[x,y,z]o(x,y,z)□distancia_nD(p1,p2) no 6 coord,coherencia
'''
#PROBADOR
print(punto_pedido('p.primer'))        #ejecutar una vez bien y una mal
# cuerpo ejecutivo no es subprograma
'''
print('\n\n                GEOMETRIA \n')    #no pedido, pero razonable
x1,y1,z1=punto_pedido('P1')
x2,y2,z2=punto_pedido('P2')
print('La distancia entre (%.2f,%.2f,%.2f) y (%.2f,%.2f,%.2f) es: %.2f metros'
      %(x1,y1,z1,x2,y2,z2,distancia_3D(x1,y1,z1,x2,y2,z2)))
#escritura con patrones=compongo el texto completo, no requiere round

```

ERRORES FRECUENTES

0.- En cuanto a la entrega

0.1 Solicitado un archivo.py, enviado un rar o varios ficheros.py—no cumple las condiciones→ evaluación =0

Tenéis un foro a vuestra disposición para hacer todo tipo de consultas

La respuesta al problema que estabais teniendo era fácil, y probablemente os la hubiera respondido cualquier compañero. Posiblemente habéis dejado en trabajo para demasiado tarde.

0.2 Orden equivocado de secciones ver transparencia S3.4

0.3 nombre de archivo inadecuado

0.4 La primera línea de todo archivo.py es un comentario indicando el OBJ general del archivo. Ver transparencias S3.4 y S3.5

0.5 separa las diferentes secciones con una línea en blanco

0.6 No importes más de lo necesario, ni más global de lo que necesites. Si solo necesitas sqrt ¿por qué importar toda la biblioteca?

0.7 Desactiva los probadores al terminar la prueba de cada subprograma

0.8 La necesidad de documentar subprogramas ha sido indiscutible, como mínimo desde 1973. Clásicamente se documentaba inmediatamente antes del subprograma, encerrado en una banderola (que en inglés se llama banner). Digamos algo parecido al recuadro que inserto tomado de una de vuestras prácticas:

```
"""*****
*          SUBPROGRAMA: Distancia entre puntos en el plano          *
*INPUT ax,ay,bx,by float    #coordenadas de puntos a y b          *
*OUTPUT dist float          #distancia en las mismas unidades
* PRE:   (aquí las precondiciones)
* POST:  (aquí las postcondiciones )
*****"""
def dist(ax,ay,bx,by) :
```

Siempre he pensado que era un sitio inadecuado, que la documentación, aunque es transparente para el ordenador “forma parte” primordial del subprograma que debería ir dentro y... ¿os he contado que me enamoré de Python? Python, da un valor añadido de la documentación y, como sabéis la muestra al programador cuando está escribiendo `def dist()`. Pero, premeditadamente, la ubica dentro de la función.

Esto es lo que estamos haciendo nosotros con las tres líneas de documentación (Hemos suprimido la POST porque en la mayoría de las situaciones, es idéntica a OBJ).

De modo que en Python (y yo opino que en cualquier lenguaje) **DEBE IR DENTRO**. pero, no lo ponemos dentro y fuera, **cualquier redundancia en código dificulta el mantenimiento** (aunque sea documentación).

Banderola o no, en otros lenguajes sería a gusto del equipo de programadores, aumenta la legibilidad, de modo que es, en principio positiva. En Python, como ha de mostrarse al programador, cuantas menos florituras tenga mejor.

0.9 por convenio, los nombres de variables y subprogramas empiezan con minúscula, las constantes en mayúscula todas las letras y se reserva la primera en mayúsculas y el resto en minúsculas para las clases (en orientación a objetos, que no es nuestro caso)

0.10 El IDLE está preparado para trabajar con “media pantalla” unos 79 caracteres. Parte las líneas de forma que no sea necesario usar las barras de desplazamiento

Apartado 1:puntuación 2,5

- 1.1 El enunciado indica que “RECIBE” las coordenadas, no que las pide.
Error muy grave. Cada subprograma debe hacer 1 y solo una cosa: Si calcula no pide, si pide no calcula.
- 1.2 Nombre inadecuado o mejorable del subprograma
- 1.3 Nombre inadecuado o mejorable de los argumentos
- 1.4 Mal las precondiciones: Solo se pueden poner PRE a las entradas, o avisando consulta global de constantes o subprogramas.
- 1.5 Por término general, y en este caso en concreto, la función que calcula el valor no debe redondear, puesto que aportaría errores irreversibles en sucesivos usos. Es la interfaz de usuario la responsable de esto.
- 1.6 Tienes que documentar cada arg de entrada y salida, Si recibe 6 arg, dirás, float, float, float.... Realmente 6 es mucho, recuerda el 7 ± 3 , está pidiendo a gritos que agrupemos x, y y z como un “punto”, y que distancia reciba 2 arg solo, pero eso no lo veremos hasta mas adelante.
- 1.7 Es recomendable evitar la conversión automática de int a float. Las constantes numéricas float han de llevar “punto”, a diccionablemente, Pep8 (recomendaciones de estilo de python indican que conviene poner al menos 1 decimal, para facilitar la lectura al humano. Asi, en el probador, debieras poner “0.0” y no “0” para las coordenadas
- 1.8 Si no hay PRE: No se pone, o se pone NONE, False quiere decir error.
NONE quiere decir no la hay
- 1.9 Los casos de prueba seleccionados parecen aleatorios.
- 1.10 Documentar los casos de prueba. El probador no debe ser un input, pues no permitiría dejarlos documentados

2 APARTADO 2 puntuación 2

- 2.1 El enunciado pide “un nuevo subprograma”, no que pruebes el anterior con $z=0$
- 2.2 El enunciado pide reuso, no que vuelvas a escribir la fórmula

3 APARTADO 3 puntuación 2,5

3.1 Identificador inadecuado: las funciones que piden “algo” al usuario, SIEMPRE, se llaman “algoPedido”.

3.2 la importación de realPedido, o va en la sección de importación del programa, o del subprograma. Nunca importación en el programa principal, mezclada con otras secciones.

3.3 No hacer reuso de la función de la biblioteca (es el principal objetivo a evaluar de este ejercicio)

3.4 si va a pedir al usuario un valor, no lo recibe como argumento. Error muy grave, implica que no se ha entendido el paso por parámetros.

3.5 NUNCA `int(input...)` o `float(input...)` sin `try except` (aunque, realmente, haciendo reuso, esto lo necesitabas

3.6 Es una función, no un procedimiento, si imprime, el programa principal no podrá reusarlo. Error Grave. Como consecuencia, en el apartado 4, redundas código

3.7. No reproduces la salida que solicita el enunciado. Otro objetivo del ejercicio es concatenar cadenas. Flexibiliza pasando parte del msg a imprimir al usuario como argumento

3.8 flexibiliza el subprograma, pasándole el mensaje, a imprimir al usuario, y/ o, los límites de la habitación como argumento

4 APARTADO (3 puntos)

4.1 Realmente, no hace falta `round`. Es simplemente un problema de formato de escritura. Mira la transparencia S3.7 y el apartado “escritura con formato” en el libro

4.2 feísimas 6 líneas casi iguales, por culpa de 3.6. Incrementa tu pensamiento en reuso.

4.3 el cuerpo ejecutivo no es un subprograma. Es un código que coordina la tarea del resto de subprogramas

Actividad 2.1: Biblioteca de estadística

La actividad 2 la vamos a dividir en dos entregables. En la primera fase vamos, simplemente, a crear una biblioteca. En la segunda fase creamos una aplicación que tome ventaja de esta biblioteca.

#COMENTARIOS

#Python ofrece la función `sum()` para secuencias numéricas, en el ejercicio profesional, lo razonable sería usar las fun proporcionadas por el lenguaje, puede que estén optimizadas. Pero, ahora lo que queremos es aprender a programarlo, de modo que

No uses la función `range` cuando ya tienes la secuencia a recorrer (ten especial cuidado si vienes de otro lenguaje)

La biblioteca dispondrá de los siguientes estadísticos 1,8:

- 1) (0,3) Total de una población muestral
- 2) (0,3) Media aritmética
- 3) (0,6) Varianza
- 4) (0,3) Desviación típica
- 5) (0,3) Coeficiente de variación

Simuladores

1,2

- 6) (0,6) Lanzar un dado de n caras

La biblioteca `random` de Python dispone de generadores de números pseudoaleatorios. Entre ellos, la función `randint(ini,fin)` devuelve un entero comprendido entre el valor "`ini`" y el valor "`fin`", de modo que `randint(1,6)` simula el lanzamiento de un dado de 6 caras.

- 7) (0,6) Monta un probador que lance el dado un número determinado de veces (¿20 por ejemplo?, indicado en una constante), de modo que puedas observar el resultado de las tiradas en una sola línea.

- 8) En la documentación de tu archivo indica el tiempo consumido, referencias y reconocimientos.

A continuación, se muestra un ejemplo de la ejecución de los casos de prueba para algunos de ellos. El último, muestra el formato del caso de prueba, pero diferentes ejecuciones proporcionarán diferentes salidas.

```
Variación( 4 , 2 )= 12 #ojo que es entero
```

```
caso de prueba (1, 2, 3) Total= 6
caso de prueba (1, 2, 3) Media= 2.0
caso de prueba (1, 2, 3) Varianza= 0.6666666666666666
caso de prueba (1, 2, 3) desviación típica=
0.816496580927726
caso de prueba (1, 2, 3) coeficiente variación=
0.408248290463863
```

lanzado 20 veces: 1, 5, 6, 3, 6, 6, 5, 4, 1, 5, 5, 6, 5, 1, 4, 1, 6, 1, 2, 1,

Estimación de tiempos: Si habéis estudiado al día, y realizado los ejercicios solicitados las semanas anteriores, este ejercicio debería consumir entre 45min y 1 hora 15 minutos

Entrega y calificación: El ejercicio se realizará en grupos de exactamente **2 personas**¹, se entregará en la actividad, en la plataforma aula virtual de la uah. Subiendo **un único archivo con la extensión “.py”**. Cada alumno deberá subir una copia idéntica a la plataforma, firmando así su autoría. El equipo debe reunirse para discutir el reparto de trabajo. Los ejercicios 1, 3, 7 y 8, deberán realizarlos todos los componentes de forma individual, acordando de forma conjunta la versión definitiva. El resto de apartados, puede hacerlos un solo alumno y revisarlos el otro. El nombre del archivo será:

“A2.1_apellidos1_nombre1_y_apellidos2_nombre_2” sustituyendo “apellidos” y nombre por los de los alumnos.

Un archivo que dé errores de tiempo de traducción o de tiempo de ejecución obtendrá la calificación de 0 directamente. **No se corregirán** archivos de más de 2 personas, **y a lo sumo 1** grupo de 2 o 1 persona². Pero, si se entregaran más de un grupo con menos de 2 personas **no se corregirá** ninguno de ellos. Tenéis el foro y otros medios para buscar alianzas con alumnos, incluso de otras titulaciones.

La calificación de cada apartado está indicada entre paréntesis. La puntuación máxima de esta fase de la actividad es 3. El resto (7) corresponderá a la segunda fase.

ADELANTE CON VUESTRO PROGRESO

¹ Ahora que ya os he corregido a cada uno, vuestra A1 y “ponernos de acuerdo” en los objetivos de programación. Deseo que vayamos practicando habilidades de trabajo en grupo. LOS POCOS ALUMNOS QUE HICIERON A1 con compañero **DEBERÁN CAMBIAR DE COMPAÑERO**

². Entre las habilidades de trabajo en equipo está la de buscar miembros del mismo.

2.1 Envío de correo electrónico desde *Python*

Hoy en día se hacen muchas gestiones por internet. El lenguaje *Python* pone a nuestra disposición un paquete con servicios para enviar email.

Escena motivadora. Un trato personalizado requiere hoy en día que a los alumnos del master del proyecto anterior se les comunique por email su nota final. El centro puede comunicarse con alumnos individuales a través de la aplicación, vía email, por otros motivos. Modifica la aplicación anterior para que cubra estos dos servicios.

Ten cuidado con el solapamiento de nombres. El intérprete busca los archivos primero en el directorio actual y luego en la biblioteca estándar. Si creamos un archivo con el mismo nombre que un módulo de la biblioteca, este no será localizable. Si hemos llamado `email.py` a nuestro programa probador de email, al importar el módulo obtendremos un error del tipo:

```
>>> import email
ImportError: No module named 'email.utils'; 'email' is not a package
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    import email
  File "C:\Python34\mis códigos\email.py", line 3, in <module>
    import smtplib
  File "C:\Python34\lib\smtplib.py", line 47, in <module>
    import email.utils
ImportError: No module named 'email.utils'; 'email' is not a package
```

Es un error frecuente y bien conocido. Lee, por ejemplo, en [error email].

En primer lugar, hemos creado una cuenta de Gmail para hacer las pruebas: CorreoPythonCurso@gmail.com cuya contraseña es correo1Python2¹.

	miEmail.py
...	#!/usr/bin/python
21	# -*- coding: utf-8 -*-
22	# Enviar correo Gmail con Python
23	# www.pythondiario.com
24	
25	import smtplib
26	
27	desde = 'correoPythonCurso@gmail.com'
28	hacia = 'correoPythonCurso@gmail.com' #aquí puedes poner cualquiera
29	msg = 'Correo enviado utilizando Python+smtplib en www.pythondiario.com'
30	

¹ Nos pareció cómodo que el lector dispusiera de una dirección de correo y su contraseña a efectos de estas pruebas. Si pese a estar publicada la respetamos, estará disponible, si se pierde el acceso, el lector necesitará crear una nueva.

31	# Datos
32	username = 'correoPythonCurso@gmail.com'
33	password = 'correo2015'
34	
35	# Enviando el correo
36	servidor = smtplib.SMTP('smtp.gmail.com:587')
37	servidor.starttls()
38	servidor.login(username,password)
39	servidor.sendmail(desde, hacia, msg)
40	servidor.quit()

Puedes ver la gestión automatizada de correo electrónico desde el lenguaje *Python* explicado paso a paso en [Davila 13]. En enero de 2016, Gmail aumentó la seguridad de sus cuentas, la opción por defecto no permite abrir tu cuenta de correo desde una aplicación, como se hace en la línea 38, por ello, la cuenta origen ha de estar desbloqueada, mientras que la de destino no tiene requisitos especiales.

La cuenta que hemos creado para el ejemplo ya está desbloqueada. Como ocurre con todos los estándares, es adecuado mantener las medidas de seguridad recomendadas, excepto, que tengas fuertes motivos para saltártelas. Tienes varias opciones:

- Utilizas mecanismos de envío más seguros (en los que no entramos pues escapa de los objetivos de nuestro tema, aunque puede ser la mejor opción).
- Mantienes una cuenta personal para este tipo de gestiones, separada del resto de tus cuentas.
- Desactivas la protección para enviar el correo y vuelves a activarla inmediatamente. Las Figuras 8.5 a 8.7 muestran como desbloquear una cuenta. Ve a “Mi cuenta” (Figura 8.5).

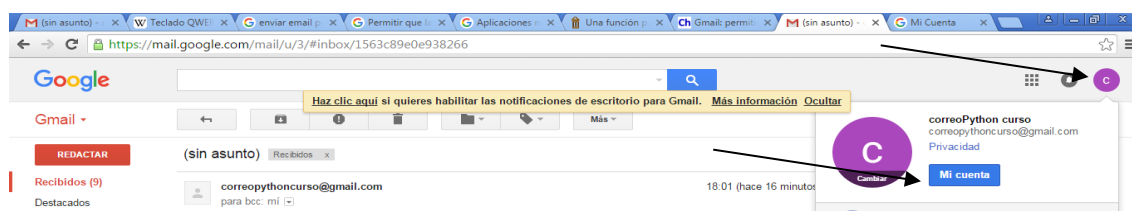


Figura 8.5. Acceso a configuración de la cuenta

Ve a “inicio de sesión y seguridad” (Figura 8.6).

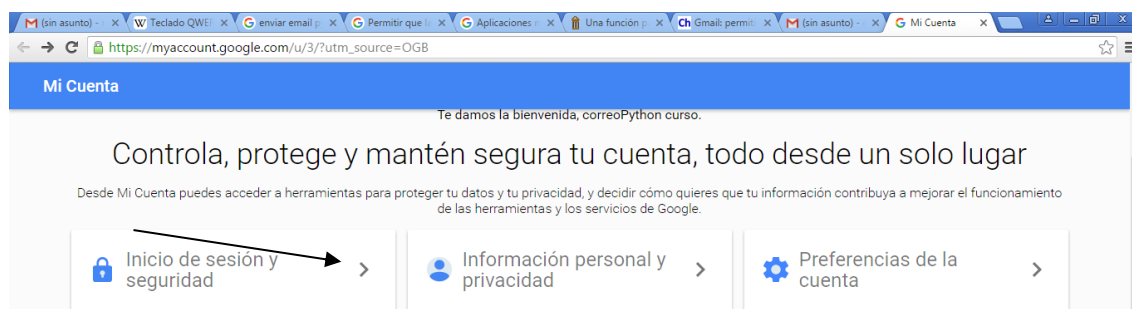


Figura 8.6. Acceso a inicio de sesión y seguridad

Y activa “Permitir el acceso a aplicaciones menos seguras” (está al fondo del todo, como puedes ver en la Figura 8.7).

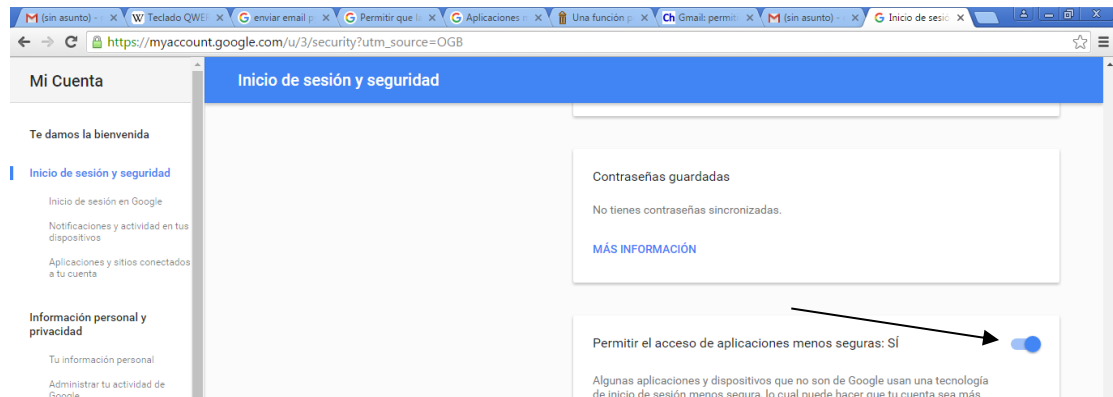


Figura 8.7. Permitir acceso a aplicaciones menos seguras

Trabajo personal 8.26: Ahora puedes ampliar la funcionalidad de tu aplicación del máster. Cuando se acaben de introducir las calificaciones, pregunta si se desea enviar calificaciones por email y obra en consecuencia.

2.1.1 Más sobre deontología profesional

El envío de correo no deseado está regulado en territorio español por la “Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico (LSSI-CE)” [LSSI 14], por la cual se prohíbe el envío de correos comerciales que no hayan sido solicitados expresamente. Incluso solicitados, son ilegales los mensajes con contenido comercial que no incorporen la palabra “publicidad” en su encabezamiento. Las sanciones económicas que contempla esta ley oscilan entre 30 mil y los 150 mil euros.

En diferentes países la legislación puede ser ligeramente distinta, has de estudiar la legislación que es aplicable a tu entorno. Adicionalmente, por encima y antes de que existe una ley, el profesional debe moverse siempre en unos criterios éticos y deontológicos.

La tecnología no tiene un valor moral de por sí. Depende del uso que se haga de ella. Un spam es un correo no deseado. Este hace perder tiempo al receptor, puede saturar su buzón de correos, impidiendo la entrada de correos realmente deseados. El envío automatizado de emails es una herramienta muy potente. Haz un uso responsable de la herramienta que acabas de aprender.

Actividad 4

Las empresas están obligadas por ley a conservar información económica de sus empleados. Nuestro cliente es una empresa pequeña, y realiza las nóminas y la contabilidad de forma manual pero conserva todos los pagos realizados a cada empleado. Vamos a empezar por una pequeña aplicación que facilita al administrativo hacer las siguientes gestiones, de manera continua:

- 1- Da de alta un empleado (dejándolo ordenado por DNI)
- 2- Apunta el pago a un empleado (dado su dni, si ya está dado de alta)
- 3- Muestra una lista ordenada por nombre de los empleados
- 4- Indica el DNI del empleado que ha acumulado el menor importe en el año
- 5- Muestra todos los datos de un empleado sabido su DNI
- 6- Envía un email a todos los empleados indicando el total de sus cobros hasta la actualidad.
- 7- (opcional) Amigo invisible

Con “de manera continua” nos referimos a que el programa no acaba al terminar de hacer una acción, por el contrario, ofrece hacer la misma o diferente gestión.

Ejemplos de ejecución son los siguientes

- 1- DNI: 22918759H
Nombre: Martínez López José Carlos
Email: jcml@gmail.com

El campo nombre debe contener, al menos nombre y primer apellido, solo puede tener caracteres alfabéticos. La aplicación lo formateará de modo que la primera letra de cada palabra letra de cada palabra vaya en mayúsculas y el resto en minúsculas, eliminando blancos innecesarios. Si alguno de los campos es imposible, la aplicación lo indicará, y volverá a pedirlo hasta que sea posible. Respecto al campo nombre.

No puede haber dos empleados con el mismo DNI (el resto de los campos si puede ser compartido por 2 empleados)

Respecto al email, debe de tener al menos un carácter entre los permitidos (A..Z, a..z, 0..9, . _) delante de una “@”, seguida por entre 2 y 5 caracteres, seguidos por “.” Y entre 2 y 3 caracteres.

Igualmente formateará la letra del DNI, una vez confirmada a mayúsculas

- 2- Indique empleado: 2291759H
Importe:509,5

Advierte que el usuario introduce “.” y no “,” para los céntimos de Euro. El importe puede ser negativo, porque el usuario deba devolver algún pago indebido, pero se mantendrá siempre en valores “razonables” para el contexto español de una pequeña empresa.

3- RESUMEN ACTUAL
Fecha 09/12/2018

Nombre	DNI	email	Total
Álvarez Juan	99999999X	miEmail@yahoo.es	2999,27
Martínez López Ana	8888888Y	elSuyo@ministerio.org	3333,98
Martínez López José Carlos	22918759H	jcml@gmail.com	0

Emplee para ello el algoritmo de la burbuja, que encontrará

4- Menores ingresos: 22918759H

5- Introduzca DNI: 22918759H

Martínez López José Carlos 22918759H jcml@gmail.com (500.0, -400.0 -100.0)

6- Hecho

7- Regala Recibe

DNI	DNI
DNI	DNI
DNI	DNI
DNI	DNI

Diseñe los datos necesarios. Puede reusar la biblioteca libInterfaz que se adjunta y cualquier recurso de listas , tuplas, cadenas y records (excepto el método sort) . Puede usar today de la biblioteca datetetime.

Comente las dificultades encontradas, recursos utilizados y tiempo empleado.

Grupos de 2 alumnos

Entrega aula virtual, archivo .Py