G1962 - Programación

Grado en Ingeniería Civil

Examen Extraordinario

Instrucciones

- Crear un único fichero mediante JupyterLab con la extensión '. ipynb '.
- Cada ejercicio debe realizarse en una celda por separado dentro del Notebook de Jupyter a excepción de los ejercicios enlazados. Cada ejercicio deberá ser documentado de manera precisa.
- Al finalizar, subir el fichero al apartado Examen Extraordinario alojado en Moodle.
- Se debe entregar esta hoja de examen como acreditación de asistencia al examen con las respuestas teóricas.
- Se dispone de un total de 2 horas de examen, tras las cuales la entrega en Moodle se cerrará, por lo que se aconseja entregar unos minutos antes ya que cualquier entrega fuera de ese periodo será descartada .

Nombre y apellidos:

Ejercicio 1 (1p):

Dado el número **0** **10000000110** **100100010...0** ,representado con el formato de punto flotante (IEEE 754) con 64 bits de precisión, se pide obtener su representación en número decimal, explicando el proceso para su obtención.

Ejercicio 2 (1p):

Indica de los siguientes códigos de Python, cuales son correctos y cuales están mal implementados, indicando en este ultimo caso que errores se cometen en la implementación y una breve idea de como se solucionarían, sin necesidad de corregirlos.

```
def exponenciacion(base, exponente):
    resultado = 1
    for i in range(str(exponente)):
    resultado = resultado * base
    return resultado
```

```
numerador = 100
denominador = '0'
division = numerador/denominador
print(float(division))
```

Ejercicio 3 (1.5p):

Implementa un código en Python que solicite al usuario que escriba por teclado el valor de una fuerza F y un ángulo de aplicación α. Posteriormente haciendo uso de los datos introducidos por teclado, que muestre por pantalla un mensaje similar al presentado a continuación tras calcular las componentes F_x y F_y de las fuerzas:

$$F_x = F \cdot \cos\alpha \tag{1}$$

$$F_y = F \cdot \sin\alpha \tag{2}$$

```
La componente X de la fuerza es igual a 'Valor calculado'.
La componente Y de la fuerza es igual a 'Valor calculado'.
```

Ejercicio 4 (3p):

3.1) (1p) Se desea implementar una **función que retorne la posición y velocidad de un objeto** en caída libre. Para ello será necesario crear una **función** que reciba los siguientes **argumentos**: *altura inicial del objeto (h) y tiempo de caída (t)*. Asumiendo la **aceleración constante (g)**

como la de la gravedad en la tierra ($\approx 9{,}8m/s^2$), se pide que la función **retorne** la posición final (y) y la velocidad final (v) en el instante (t) proporcionado como argumento.

$$y = h - \frac{1}{2} \cdot g \cdot t^2 \tag{3}$$

$$v = -g \cdot t \tag{4}$$

3.2) (0.5p) Posteriormente se pide utilizar dicha función para calcular la posición y velocidad finales tras 1 segundo de caída, de un objeto que en origen estaba suspendido en reposo a 35 metros de altura.

3.3) (1.5p) Haciendo uso de la función anterior, representar las siguientes dos gráficas con la librería que consideres oportuna (*recomendable* MATPLOTLIB):

- Gráfica de altura: evolución de la altura de un objeto desde el instante $t = 0$ donde esta suspendido en reposo a 100 metros de altura hasta que alcanza el suelo ($h = 0$).
- Gráfica de velocidad: evolución de la velocidad de un objeto desde el instante $t = 0$ donde esta suspendido en reposo a 500 metros de altura hasta que alcanza el suelo ($h = 0$).

Ejercicio 5 (3.5p):

Implementar una **clase** llamada **Dado** que tenga como atributo una variable *caras*. También necesitamos implementar su método **constructor (0.5p)** que recibe como argumento el *numero de caras* y lo almacena en el atributo de la clase. Por otro lado, también queremos implementar los siguientes métodos:

- **(1p) tirarDados(self, n)**: Método que mediante la utilización de una librería de generación de números aleatorios (*recomendado* NUMPY), genera n tiradas del dado aleatorias teniendo en cuanta su número de caras. Estas tiradas deberán ser **almacenadas y retornadas** en un **diccionario** donde la clave es el número de la tirada y el valor el resultado.
- **(0.75p) probabilidadExperimento(self, l)**: Método que recibe como **argumento** una lista de posibles resultados del dado y **retorna** la probabilidad de obtener alguno de ellos teniendo en cuenta el número de caras del dado. El método deberá prevenir mediante el método que consideres oportuno posibles excepciones o valores de entrada erróneos.

$$Ley\ de\ Laplace : P(Z) = \frac{Casos\ probables}{Casos\ posibles} \tag{5}$$

- **(0.75p) sumaTiradasRecursivo(self, r, ...)**: Implementación recursiva de un método que recibe una lista de resultados r de tiradas del dado y las suma para posteriormente **retornar** el valor resultante.

Posteriormente crear algunas instancias de la clase para comprobar que la clase y todos sus métodos asociados funcionan como se espera (0.5p), comentando los resultados obtenidos.