Física Computacional II (510240)

Universidad de Concepción
Facultad de Ciencias Físicas y Matemáticas
Departamento de Física

Profesor : Roberto E. Navarro
Ayudantes : Fernanda C. Mella
Amaro A. Díaz

Tarea 1: Python

Fecha de entrega: 8 de octubre de 2025, 23:59 hrs.

Esta evaluación tiene como objetivo que los estudiantes demuestren su comprensión y habilidades para trabajar con listas de `Python` y arreglos de `Numpy`.

Instrucciones generales: Lea atentamente estas instrucciones antes de comenzar.

- Esta tarea puede responderse en grupos de 3 estudiantes (2 en casos debidamente justificados). Los grupos no son fijos y podrán reorganizarse en cada nueva tarea.
- Entregue sus respuestas a esta tarea antes de la fecha declarada arriba de este recuadro.
- Sus respuestas deben ser agregadas a su portafolio personal, el cual se aloja en un repositorio de GitHub dentro de la organización `fiscomp2-UdeC2025`. Para acceder a su portafolio por primera vez, use este enlace: `https://classroom.github.com/a/eYo8qQSw`.
- La solución de cada problema debe ser presentada en un capítulo independiente del portafolio (un problema = un capítulo).
- Aunque la tarea se resuelva en grupos, **cada estudiante es responsable de mantener su propio portafolio**, asegurándose de la integridad del mismo (por ejemplo: realizar los *commits* y *pushes* antes del plazo límite, comprobar que el documento compila correctamente y sin errores, y redactar mensajes de *commits* de forma clara, descriptiva y adecuada). Si un integrante no cumple con esta obligación, su tarea se calificará con nota mínima, sin afectar al resto del grupo.
- Revise la plantilla del portafolio para instrucciones adicionales.

Pregunta 1 En Python existe un módulo llamado `this` que, al importarlo, imprime *El Zen de Python*. Sin embargo, el módulo está implementado de forma especial: guarda el texto en una variable codificada y lo descifra para mostrarlo. El objetivo es investigar este módulo y explicarlo en detalle.

(a) Encuentre en su computador el archivo que define el módulo `this`. Use exclusivamente herramientas de su sistema operativo (por ejemplo, el buscador de archivos, `find` o `grep` en Linux/macOS, `dir` o el buscador en Windows, etc.) y **describa el procedimiento** que siguió para encontrar este módulo. *No basta con indicar solo la ruta: explique los pasos y/o comandos utilizados.* Finalice con la *ruta completa* (*path*) del archivo hallado.

Note que esta parte depende de su sistema operativo y de la forma en que instaló python.

(b) Este módulo contiene el siguiente código:[1]

```python
s = """Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
[... otras 17 líneas ...]
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)

print("".join([d.get(c, c) for c in s]))
```

Explique en sus palabras qué propósito cumple cada *parte* del código. En particular:

I. ¿Qué representan los números 65, 97, 26 y 13?
II. ¿Cuál es la intención de cada ciclo `for`?
III. ¿Qué son `d` y `chr`?

[1] `https://github.com/python/cpython/blob/main/Lib/this.py`

iv. ¿Qué intención tiene la asignación `d[...]=...`?

v. ¿Qué hace el operador `%` y por qué es útil aquí?

vi. ¿Qué hacen `join` y `get` en la última línea?

vii. ¿Cuál es el resultado de `[d.get(c,c) for c in s]` en la última línea?

Pregunta 2 Averigüe y explique qué hacen las siguientes funciones de `numpy`: `arange`, `array`, `linspace`, `geomspace`, `ones`, `zeros`, `random.random`, `random.normal`, `random.randint`. Presente su respuesta en una tabla de tres columnas: (i) el nombre de la función, (ii) la explicación y (iii) un ejemplo con su resultado.

Con esta información, responda a los siguientes problemas:

1. Genere un arreglo $N > 200$ números **reales aleatorios** $\{a_n\}_{n=0,\cdots,N-1}$ siguiendo una **distribución normal**. En una misma figura, haga un gráfico de a_n como función de n, y un histograma de $\{a_n\}$.
2. Defina una matriz de 3×5 números complejos. Luego, imprima en pantalla la primera fila completa, la última columna completa y el elemento que se encuentra en la segunda fila y primera columna.
3. Genere una serie $\{x_i\}_{i=0,\ldots,N}$ de números **enteros aleatorios**. Considere el caso $N = 10$. Luego, evalúe una nueva serie definida por $y_i = x_{i+1} - x_i$ que representa la diferencia de elementos adyacentes de la serie $\{x_i\}$. Para esto, considere (a) usar explícitamente un ciclo `for`, y (b) operaciones elementales de arreglos de `numpy` sin usar un ciclo `for` (a esto se conoce como una operación vectorial).
4. Defina la variable x como un arreglo de 20 elementos en orden creciente en el intervalo $-2\pi \leq x \leq 2\pi$. Luego, grafique $\cos x$. Si observa una función que no es suave, interprete sus resultados.

Pregunta 3 Los números de Catalán se definen, para $n \geq 0$, como:

$$C_n = \frac{(2n)!}{(n+1)!\, n!} \,. \tag{1}$$

Como referencia, los primeros números de Catalán son:

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
1	1	2	5	14	42	132	429	1430	4862	16796	58786

(a) Calcule los primeros $N = 15$ números de Catalán según (1), usando las dos funciones de factorial dadas a continuación. La primera usa instrucciones nativas de `Python`, mientras que la segunda usa funciones de `Numpy` y permite controlar la precisión. **Use exclusivamente estas implementaciones** y no otras funciones para calcular factoriales. Luego, grafique C_n como función de n en un gráfico de puntos con eje vertical semi-logarítmico. Pruebe con números enteros de 16, 32 y 64 bits. Comente y explique los resultados obtenidos (por ejemplo, sobre posibles desbordamientos u errores de redondeo).

```python
import numpy as np

def factorial(N):
    """Retorna una lista de factoriales desde 0! hasta N!. Por ejemplo:
    factorial(5) -> [1, 1, 2, 6, 24, 120]
    """

    f = 1       # note el uso del operador walrus := a continuación
    return [1 if n==0 else (f:=f*n) for n in range(N+1)]


def factorial_numpy(N, dtype=np.int32):
```

```
    """Retorna un arreglo de numpy de factoriales desde 0! hasta N!.
    El argumento ‘dtype‘ controla la precisión. Ejemplos:
    factorial_numpy(5)                  -> array([1, 1, 2, 6, 24, 120], dtype=int32)
    factorial_numpy(5, dtype=np.int64) -> array([1, 1, 2, 6, 24, 120], dtype=int64)
    """
    f = np.arange(N+1, dtype=dtype)
    f[0] = 1
    return f.cumprod(dtype=dtype)
```

Note que estas dos funciones retornan un arreglo completo de factoriales. Por lo tanto, basta llamarlas una sola vez y reutilizar el resultado. Por ejemplo, `f=factorial(5)` crea una lista `f` con elementos desde `f[0]==1` hasta `f[5]==120`, de modo que en general `f[n]==n!`.

(b) Demuestre, analíticamente, que (1) se puede reescribir como una relación de recurrencia de la forma:

$$C_0 = 1\,, \qquad\qquad C_{n+1} = \frac{4n+2}{n+2} C_n\,. \tag{2}$$

Luego, implemente esta relación de recurrencia, asegurándose de trabajar siempre con precisión de 16 o 32 bits (`c=np.int16(5)` fuerza a que `c` sea un entero de 16 bits). Compare el resultado para los primeros $N = 15$ números de Catalán con respecto a sus resultados usando `factorial_numpy`.

(c) Verifique gráficamente que, para $n \leq 15$, la serie presenta un comportamiento asintótico de la forma

$$C_n \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}\,. \tag{3}$$