Quiz 2 - Computational Physics II

NAME: Juan Daniel Vásconez Vela **SCORE:** 15/20

Date: Thursday 20 March 2025 **Duration:** 45 minutes

Credits: 20 points (4 questions) **Type of evaluation:** LAB

Provide short and concise answers to the following items. Code syntax should be clear.

1. **(5 points) Python classes**
   (a) Provide a simple code snippet of a Python class showing how it is defined and its components.
   (b) Explain the difference between an instance attribute and a class attribute in a Python class.

-2.2

a)

```
class Student:
    """ Docstring
    def __init__(self, name, age)
        self.name = name
        self.age = age.
```

↳ What about cls attributes & methods?

b)

Class atribute: Related to self, are the variables that refers directly and explicat to the class in general

Instance Atribute: → self parameters inside the class methods and not explicatly related to the class factory

2. **(5 points) Python decorators**
   (a) What is a decorator in Python, and what is its purpose?
   (b) Provide a simple code example of a decorator and explain what it does.

(b)

```
def mydecorator(funct)
    def wrapper(fino)
        funct()
        print("Extra funcionality")
    return wrapper()

@ my decorator
def function(*args)
    print("function itself")
```

Docstrings are missing

(a)

A decorator is a function that adds functionality to another class without changing the inital ~~class~~ code.

function

3. **(5 points) Python packages**

(a) Describe the typical directory structure of a well-designed Python package.

(b) Describe the primary purpose of the `argparse` module.

a)

- package
  - module.py
  - __init__.py
- package_other
  - module.py
  - __init__.py
- readme.md
- setup.py
- requirements.txt
- usage_example.ipynb
- test.py

b)

argparse module is a package used to give the python script atributes to be interacted with on the comand lines, modifying or giving parameters from the exuton or called flags. depends on

__name__ = __main__

4. **(5 points) Testing Python modules**

(a) Why is it important to add testing classes to Python modules?

-2.8

(b) Write set-up and tear-down test classes using `pytest` for the class below. Your test class should check if the `time_of_flight` method returns an expected value.

```
1  import numpy as np
2  class ParabolicMotion:
3      """ A class to calculate the flight time of a projectile."""
4      def __init__(self, v0, angle):
5          """Initial velocity (v0 in m/s) and launch angle (degrees)."""
6          self.v0 = v0
7          self.angle = np.radians(angle)
8          self.g = 9.81  # Gravity (m/s^2)
9      def time_of_flight(self):
10         """Returns the total time the projectile stays in the air."""
11         return (2 * self.v0 * np.sin(self.angle)) / self.g
```

a) To prevent the code to break if the user misuses it giving it wrong parameters or variables

Bug detection.
Refactoring code.
Validating code

b)

```
def time_of_flight_test(t_o_f):
    if t_o_f < 0 ?
        raise ValueError as e:
            print('e', "Inital velocity must be a positive number")
```

X X ✓