Assignment 3

Problem 1

Write Python code in a Jupyter notebook to evaluate the following expressions for $x = 3.0$ and $y = 4.0$. *Note* that variable names are case-sensitive, and lowercase variable names are generally preferred.
Hint: Import NumPy using `import numpy as np`. You can use the functions `np.exp()`, `np.sqrt()`, `np.sin()`, `np.log10()`, and `np.log()`. The function `np.log()` computes the natural logarithm. Use the constant `np.pi` for π.

(a) $F(x,y) = yA\exp\left[-\frac{(x-x_0)^2}{2\sigma_x^2}\right]$ where $A = 1.5,\ x_0 = 0.5,\ \sigma_x = 3.0$.

(b) $g(x,y) = \frac{\sqrt{x+y}}{\sqrt{x}+\sqrt{y}}$

(c) $f(x,y) = \exp\left(\frac{-\sin(\pi x)}{(x+y)^2+xy^6}\right)$

(d) $G(x,y) = \log_{10}(x) + \ln(y)$

Problem 2

Write Python code in a Jupyter notebook to calculate the mean molar mass of dry air using the simplified four-component model in the following table. The composition is given as mole percentage.

Component	**Molar mass (g·mol^{-1})**	**Mole percentage (%)**
N_2	28.013	78.084
O_2	31.999	20.946
Ar	39.948	0.9290
CO_2	44.010	0.041

You can store the molar masses and mole percentages in separate dictionaries, using the component names as keys in both dictionaries. Use a `for` loop to calculate the weighted average:

$$\langle M\rangle_{\text{air}} = \sum_i x_i M_i$$

where x_i is the mole fraction and M_i is the molar mass of the ith component. Check that the mole fractions sum to 1.

Problem 3

Write Python code in a Jupyter notebook to write an `XYZ` file for a water molecule with the following atomic Cartesian coordinates. Store the atomic symbols and coordinates in Python lists, and use a loop to write the file.

Atom	x (bohr)	y (bohr)	z (bohr)
O	0.00000000	0.00000000	0.00000000
H	1.80941647	0.00000000	0.00000000
H	-0.45334744	1.75170319	0.00000000

Use an `if`/`elif`/`else` structure to handle units. If the coordinates are given in bohr, convert all coordinates to ångström using 1 bohr = 0.5292 Å. For units different from bohr or ångström, display an error message and do not write the file.
Write the XYZ file `water.xyz` with the standard format:

1. First line: number of atoms.
2. Second line: a short comment, e.g. `H2O coordinates in Å`.
3. Then one line per atom: `symbol x y z`, with all coordinates in ångström and six digits after the decimal point.

Hint: Below is an example of Python code to write to a text file. The character `\n` starts a new line.

```
with open('example.txt', 'w') as f:
    f.write('This is an example, line 1' + '\n')
    f.write('This is an example, line 2' + '\n')
```

To format an atom record inside your file-writing loop, you can use:

```
f.write(f"{symbol} {x:.6f} {y:.6f} {z:.6f}\n")
```

Here, `symbol` is the atomic symbol and `x`, `y`, and `z` are its coordinates in ångström. The `f` prefix inserts the values in braces, and `:.6f` formats a number with six digits after the decimal point.

Problem 4

Use the original Cartesian coordinates in bohr from the table in Problem 3 to compute:

(a) The distance (in bohr) between the oxygen atom and each hydrogen atom.

(b) The H–O–H bond angle (in degrees).

Hints:

1. The distance between two points in Cartesian coordinates, $\mathbf{r}_1 = (x_1, y_1, z_1)$ and $\mathbf{r}_2 = (x_2, y_2, z_2)$, is given by:
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$
2. To compute the bond angle, define two vectors $\mathbf{a}$ and $\mathbf{b}$ along the O–H bonds, both pointing away from the oxygen atom:
$$\mathbf{a} = \mathbf{r}_{\mathrm{H1}} - \mathbf{r}_{\mathrm{O}}, \qquad \mathbf{b} = \mathbf{r}_{\mathrm{H2}} - \mathbf{r}_{\mathrm{O}}$$
The bond angle θ can then be obtained from the dot product:
$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}|\,|\mathbf{b}|}$$
where $|\mathbf{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ is the norm (or magnitude) of vector $\mathbf{a}$.
3. You can represent each atomic position as a NumPy array using `np.array([x, y, z])`. In the calculations, you can use `np.dot`, `np.linalg.norm`, and `np.arccos`.
4. The result of `np.arccos` is in radians. Convert it to degrees using `np.degrees`, or by multiplying by `180 / np.pi`.