

```
In [ ]: # Initialize Otter
import otter
grader = otter.Notebook("bond_polarity.ipynb")
```

Identifying Polar and Non-Polar Bonds

Created and developed by [Suparna Kompalli](https://www.linkedin.com/in/suparna-kompalli-79463b229/) and [Brandon Concepcion](https://www.linkedin.com/in/brandonconcepcion/), with assistance and supervision by [Jonathan Ferrari](https://www.linkedin.com/in/jonathanferrari/), [Professor Darcie McClelland](https://www.linkedin.com/in/darcie-mcclelland-descalzo-56796b1b/), and [Professor Eric Van Dusen](https://www.linkedin.com/in/ericvd/) as part of our work with UC Berkeley's [College of Computing, Data Science and Society](https://cdss.berkeley.edu/) as well as [El Camino College](https://www.elcamino.edu/)

Bond Polarity

Welcome! In this assignment, we will explore the concept of chemical bonding, focusing on how to determine whether a bond is polar or non-polar. Understanding the nature of these bonds is crucial in chemistry as it influences the physical properties and behavior of molecules. We will also delve into identifying the specific bonds that make up different molecules, helping to develop a deeper understanding of molecular structure and its implications on molecular interactions.

By the end of this assignment, you will be equipped with the knowledge to analyze and classify bonds in various compounds based on their polarity and the types of atomic interactions that hold them together.

Start by running the cell below to import our necessary libraries!

```
In [52]: import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw
import ipywidgets as widgets
from IPython.display import display
import io
from ipywidgets import interact
import rdkit.Chem.rdCoordGen as rdCoordGen
import py3Dmol as pm
import rdkit
from rdkit.Chem import rdchem
from utils import *

df = pd.read_csv("properties.csv")

def draw_molecules(lst):
    elements = []
    for molecule in lst:
        m = Chem.MolFromSmiles(molecule)
        img = Draw.MolToImage(m)
        img_byte_array = io.BytesIO()
        img.save(img_byte_array, format='PNG')
        img_byte_array.seek(0)
        elements.append(widgets.Image(value=img_byte_array.read(), format='png'))
    row = widgets.HBox(elements)
    display(row)
```

Section 1: Understanding Molecules

Let's explore some molecules to better understand their composition. In this exercise, we'll see how visualizations can help us gain deeper insights into chemical structures and reactions.

It's incredible to think that just a few hundred years ago, visualizing molecules like [ethanol](https://pubchem.ncbi.nlm.nih.gov/compound/Ethanol) (<https://pubchem.ncbi.nlm.nih.gov/compound/Ethanol>) would have been thought of as impossible without. Thanks to modern computational tools like [RDKit](https://www.rdkit.org/) (<https://www.rdkit.org/>) and [py3Dmol](https://pypi.org/project/py3Dmol/) (<https://pypi.org/project/py3Dmol/>), we can now generate and manipulate 2D and 3D representations of chemical structures efficiently!

Run the cell below. Please do not modify any of the lines—this cell has been pre-written to generate a specific visualization you'll use to answer the following questions.

```
In [53]: draw_molecules(['CCO'])

HBox(children=(Image(value=b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01,\x00\x00\x01,\x08\x02\x00\x00\x00...
```

We can transition Ethanol to [Acetic Acid \(https://pubchem.ncbi.nlm.nih.gov/compound/Acetic-Acid\)](https://pubchem.ncbi.nlm.nih.gov/compound/Acetic-Acid) (the main component of vinegar) by *oxidizing the ethanol*-- a chemical process in which ethanol loses electrons and gains oxygen. This oxidation typically occurs in two steps:

1. First, Ethanol ($\text{C}_2\text{H}_5\text{OH}$) is oxidized to acetaldehyde (CH_3CHO) by an oxidizing agent such as potassium dichromate ($\text{K}_2\text{Cr}_2\text{O}_7$) or any mild oxidizer. In this step, ethanol loses two electrons and one hydrogen atom (oxidized), forming acetaldehyde.
2. Acetaldehyde (CH_3CHO) is further oxidized to acetic acid (CH_3COOH). This is typically achieved by further oxidation using stronger oxidizers.

Converting ethanol to acetic acid requires two stages of oxidation: the first transforms ethanol into acetaldehyde, and the second further oxidizes acetaldehyde into acetic acid. We can further describe the process by using the table below:

Step	Molecule	Formula	Description
1	Ethanol	$\text{C}_2\text{H}_5\text{OH}$	Starting alcohol
↓	Oxidation		(loss of H, gain of O)
2	Acetaldehyde	CH_3CHO	Intermediate aldehyde
↓	Oxidation		(further oxidation)
3	Acetic Acid	CH_3COOH	Final carboxylic acid product

Let's try and visualize this transition!

Run the cell below. Please do not modify any of the lines—this cell has been pre-written to generate a specific visualization you'll use to answer the following questions.

```
In [54]: draw_molecules(['CCO', 'C=O', 'CC(=O)O'])  
  
HBox(children=(Image(value=b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01,\x00\x00\x01,\x08\x02\x00\x00\x00...
```

Question 1.1: How many hydrogen bonds are in Ethanol?

Type your answer here, replacing this text.

SOLUTION: Ethanol can form 1 hydrogen bond per molecule (with other molecules), using its $-\text{OH}$ group.

Question 1.2: How many hydrogen bonds are in Acetic Acid?

Type your answer here, replacing this text.

SOLUTION: Acetic acid can form up to 2 hydrogen bonds per molecule.

Section 2: Building Molecules

We'll begin our journey by learning how to construct the water molecule (H_2O). We'll promptly move to other examples later.

The first step is to create an instance of a molecule. This is something we'll need to do **each time** we want to build a new molecule from scratch.

Run the cell below to initialize the structure of the water molecule.

```
In [55]: h2o_molecule = Chem.RWMol()
```

Now it's time to add atoms to our molecule! To do this, we first create an instance of the atom we want to include, and then we add it to our molecule.

Let's start by adding an oxygen atom

```
In [56]: oxygen = Chem.Atom(8)
oxygen_index = h2o_molecule.AddAtom(oxygen)
```

We begin by creating an element instance using the `Chem.Atom()` method, which takes the atomic number (i.e., number of protons/electrons) as an argument. Since oxygen has an atomic number of 8, we pass in the number 8 to represent it. After creating the atom, we add it to the molecule we initialized earlier.

Question 2.1: Use the cell below to add the rest of the atoms needed to build H_2O .

Hint: Follow the example in the cell above.

```
In [57]: # BEGIN SOLUTION NO PROMPT
hydrogen1 = Chem.Atom(1)
hydrogen1_index = h2o_molecule.AddAtom(hydrogen1)

hydrogen2 = Chem.Atom(1)
hydrogen2_index = h2o_molecule.AddAtom(hydrogen2)
# END SOLUTION
""" # BEGIN PROMPT
hydrogen1 = ...
hydrogen1_index = ...

hydrogen2 = ...
hydrogen2_index = ...
""" # END PROMPT
```

```
Out[57]: ' # BEGIN PROMPT\nhydrogen1 = ...\nhydrogen1_index = ...\n\nhydrogen2 = ...\nhydrogen2_index = ...\n'
```

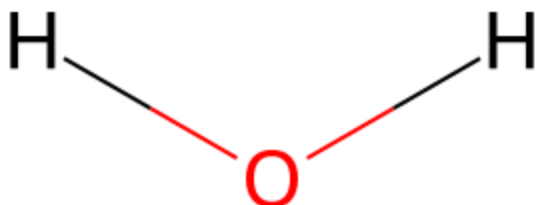
```
In [ ]: grader.check("q21")
```

We have our atoms, but they aren't connected yet! Let's go ahead and add the necessary bonds to complete the structure of our molecule.

Question 2.2: Fill in the cell below to visualize our final H₂O molecule.

```
In [62]: # BEGIN SOLUTION NO PROMPT
h2o_molecule.AddBond(oxygen_index, hydrogen1_index, Chem.BondType.SINGLE)
h2o_molecule.AddBond(oxygen_index, hydrogen2_index, Chem.BondType.SINGLE)
# END SOLUTION
""" # BEGIN PROMPT
h2o_molecule.AddBond(oxygen_index, hydrogen1_index, Chem.BondType.SINGLE)
h2o_molecule.AddBond(oxygen_index, ..., ...)
""" # END PROMPT
Chem.Draw.MolToImage(h2o_molecule)
```

```
Out[62]:
```



```
In [ ]: grader.check("q21")
```

Question 2.3: Using the example above, fill in the cell below to create a molecule of your choice! In the answer box below, describe the molecule you selected, why you chose it, and how you used RDKit to visualize it. Be sure to explain your reasoning and approach!

```
In [66]: molecule = Chem.RWMol()  
  
# YOUR WORK HERE  
  
Draw.MolToImage(molecule)
```

Out[66]:

SOLUTION: Students may have used a similar approach as demonstrated above to generate their molecule. Any clear description and justification of their process/reasoning will receive full credit.

Section 3: Polar and Nonpolar Bonds

Chemical bonds (<https://www.britannica.com/science/chemical-bonding>) form the foundation of molecules, determining their shape, interactions, and function in biological and chemical systems. In this section of the notebook, we will explore two fundamental types of covalent bonds: **polar** bonds and **non-polar** bonds. To understand this process a bit better, let's refer to the following analogy

Tug of War

Imagine it's a scorching summer afternoon and you and your friend are locked in an intense game of tug-of-war. The rope stretches taut between you, perfectly balanced. Neither side is able to overpower the other. Since you both have the same strength and the same determination, for now it remains a fair fight.

If we swap out the rope for electrons, and the players for atoms—suddenly, you're looking at the fundamental nature of **covalent** (<https://www.britannica.com/science/covalent-bond>) chemical bonding.

Playing Fair

Now let's consider the case where you and your friend are **equally** strong. We wouldn't expect the rope to move much if at all. This is exactly what happens in **non-polar** covalent bonds. When two atoms have equal or nearly equal electronegativities, they share electrons relatively evenly.

Examples of this include:

- O₂ (oxygen gas)
- N₂ (nitrogen gas)
- CH₄ (methane)

Since non-polar molecules have no distinct positive or negative sides, they tend to be hydrophobic (water-fearing) and don't dissolve well in water. Think of how oil and water don't mix—oil is made of non-polar molecules that refuse to interact with water.

"Hitting" the Gym

Now imagine that this friend of yours has been hitting the gym, where their **forearms** (<https://www.youtube.com/shorts/OG131ahlxEs>) and **back** (<https://www.youtube.com/shorts/7yN-o28CKtc>) have grown **massive**. When y'all play tug of war again, they *easily* pull the rope toward themselves, making the game unfair. This kind of scenario happens in **polar covalent bonds** (<https://www.sciencedirect.com/topics/chemistry/polar-covalent-bond>), where for specific molecules, one atom is significantly more electronegative than the other. Thus, it yanks more on the shared electrons, bringing them closer.

This *uneven* sharing creates partial charges:

- The stronger atom (which pulls electrons closer) gets a partial negative charge (δ^-)
- The weaker atom (which loses some electron density) gets a partial positive charge (δ^+)
- This is why molecules like water (H₂O) are polar. Oxygen is electronegative and pulls the electrons away from hydrogen, making it slightly negative (δ^-) and leaving the hydrogens slightly positive (δ^+).

Some other examples of polar molecules include:

- H₂O (water)
- NH₃ (ammonia)
- CO (carbon monoxide)

Since polar molecules have distinct positive and negative sides, they are hydrophilic (water-loving) and dissolve easily in water. This is why salt and sugar dissolve in water, but oil does not.

Electronegativity

Below we reference [Dr. Scott's \(https://learnwithdrscott.com/\)](https://learnwithdrscott.com/) simplified table of electronegativity values for the elements in the periodic table. You'll notice that the noble gases typically have no listed electronegativity values. This is because they already possess a full valence shell (octet) and, in most cases, do not need to form bonds with other elements.



No description has been provided for this image

Question 3.1 Use the table above to determine the electronegativity value of gold (Au). Assign this value to it to the variable gold_EN .

```
In [67]: # BEGIN SOLUTION NO PROMPT
gold_EN = 2.4
# END SOLUTION
""" # BEGIN PROMPT
gold_EN = ...
""" # END PROMPT
print(f"Based on the table above, gold's electronegativity value is roughly {gold_EN}")
```

Based on the table above, gold's electronegativity value is roughly 2.4

```
In [ ]: grader.check("q31")
```

Electronegativity Difference

The difference in electronegativity (ΔEN) between two bonded atoms determines whether the bond is:

- **Non-Polar Covalent:** Electrons are shared almost equally.
- **Polar Covalent:** Electrons are unequally shared, creating partial charges.
- **Ionic:** One atom completely transfers electrons to another.

The table below summarizes these bond types based on **electronegativity difference ranges** and provides an example for each type.

Electronegativity Difference (ΔEN)	Bond Type	Example Bond	Explanation
≤ 0.4	Non-Polar Covalent	C-H ($\Delta EN = 0.4$)	Electrons are shared almost equally, no significant charge separation.
> 0.4 and ≤ 1.7	Polar Covalent	O-H ($\Delta EN = 1.4$)	Electrons are unequally shared, creating partial positive and negative charges.
> 1.7	Ionic	Na-Cl ($\Delta EN = 2.1$)	One atom completely transfers electrons to another, forming charged ions.

Question 3.2 For each of the following bonds, use [Dr. Scott's \(https://learnwithdrscott.com/\)](https://learnwithdrscott.com/) table to determine the electronegativity difference between the two elements involved. Then, assign that value to the corresponding variable as shown below:

Hint: Since electronegativity values must always be nonzero, you may find the `abs` function useful!

- MgCl_2
- TiO
- S-F
- N-O
- C-S
- Cl-O

```
In [71]: # BEGIN SOLUTION NO PROMPT
Mg_Cl = abs(3.0 - 1.2)
Ti_O = abs(3.5 - 1.5)
S_F = abs(4.0 - 2.5)
N_O = abs(3.5 - 3.0)
C_S = abs(2.5 - 2.5)
Cl_O = abs(3.5 - 3.0)
# END SOLUTION
""" # BEGIN PROMPT
Mg_Cl = ...
Ti_O = ...
S_F = ...
N_O = ...
C_S = ...
Cl_O = ...
""" # END PROMPT
```

```
Out[71]: ' # BEGIN PROMPT\nMg_Cl = ...\nTi_O = ...\nS_F = ...\nN_O = ...\nC_S = ...\nCl_O = ...\n'
```

```
In [ ]: grader.check("q32")
```

Glycine $\text{C}_2\text{H}_5\text{NO}_2$

In this next part of the notebook, we'll apply what we've learned about molecular structure and electronegativity differences to determine whether the Glycine molecule is polar or nonpolar.

To begin, let's calculate the electronegativity differences for each of the bonds in the molecule!

Question 3.4. For each of the following bonds in glycine, calculate the electronegativity difference using Dr. Scott's table. Remember to subtract the smaller electronegativity from the larger one so the result is positive.

```
In [78]: # BEGIN SOLUTION NO PROMPT
en_diff_C_H = abs(2.5 - 2.1) # Carbon-Hydrogen bond
en_diff_C_O = abs(3.5 - 2.5) # Carbon-Oxygen bond
en_diff_N_H = abs(3.0 - 2.1) # Nitrogen-Hydrogen bond
en_diff_C_N = abs(3.0 - 2.5) # Carbon-Nitrogen bond
# END SOLUTION
""" # BEGIN PROMPT
en_diff_C_H = 2.5 - 2.1 #Carbon-Hydrogen bond
en_diff_C_O = 3.5 - 2.5 #Carbon-Oxygen bond
en_diff_N_H = 3.0 - 2.1 #Nitrogen-Hydrogen bond
en_diff_C_N = 3.0 - 2.5 #Carbon-Nitrogen bond
""" # END PROMPT

print(f"Electronegativity Differences in Glycine Bonds:")
print(f"  C-H bond: {en_diff_C_H}")
print(f"  C-O bond: {en_diff_C_O}")
print(f"  N-H bond: {en_diff_N_H}")
print(f"  C-N bond: {en_diff_C_N}")
```

```
Electronegativity Differences in Glycine Bonds:
  C-H bond: 0.3999999999999999
  C-O bond: 1.0
  N-H bond: 0.8999999999999999
  C-N bond: 0.5
```

```
In [ ]: grader.check("q34")
```

Question 3.5 Based on the electronegativity differences you calculated, which of the following bonds in glycine are polar?

1. Only the C-H bonds
2. Only the C-O and N-H bonds
3. All of the listed bonds (C-H, C-O, N-H, and C-N)
4. The C-O, N-H, and C-N Bond
5. None of them

Set the variable `answer` to the **number** corresponding to your chosen option. For example, if you believe options 1 is correct, set `answer = 1`.

```
In [84]: # BEGIN SOLUTION NO PROMPT
answer = 4
# END SOLUTION
""" # BEGIN PROMPT
answer = ...
""" # END PROMPT
```

```
Out[84]: ' # BEGIN PROMPT\nanswer = ...\n'
```

```
In [ ]: grader.check("q35")
```

Now that we have identified which of the bonds are polar, let's try and determine the molecular shape of glycine! We will first use Rdkit to accomplish this.

Question 3.6. Fill in the ellipses below with the correct atomic numbers for each element in glycine. For example, in the carbon assignment statement, we used 6 because carbon is the sixth element on the periodic table! Use the comments to help guide your choices.

```
In [88]: # BEGIN SOLUTION NO PROMPT
glycine = Chem.RWMol()
# Add atoms to the molecule
carbon = glycine.AddAtom(Chem.Atom(6))      # Carbon atom
hydrogen = glycine.AddAtom(Chem.Atom(1))    # Hydrogen atom
oxygen = glycine.AddAtom(Chem.Atom(8))      # Oxygen atom
nitrogen = glycine.AddAtom(Chem.Atom(7))    # Nitrogen atom
# END SOLUTION
""" # BEGIN PROMPT
glycine = Chem.RWMol()
# Add atoms to the molecule
carbon = glycine.AddAtom(Chem.Atom(6))      # Carbon atom
hydrogen = ....AddAtom(Chem.Atom(1))      # Hydrogen atom
oxygen = glycine...(...)                   # Oxygen atom
nitrogen = glycine.AddAtom(...)           # Nitrogen atom
""" # END PROMPT
```

```
Out[88]: ' # BEGIN PROMPT\nglycine = Chem.RWMol()\n# Add atoms to the molecule\ncarbon = glycine.AddAtom(Chem.Atom(6))      # Carbon atom \nhydrogen = ....AddAtom(Chem.Atom(1))      # Hydrogen atom \noxygen = glycine...(...)                   # Oxygen atom \nnitrogen = glycine.AddAtom(...)           # Nitrogen atom \n'
```

```
In [ ]: grader.check("q36")
```

Question 3.7. We haven't added all the remaining atoms needed to complete the glycine molecule yet! Let's go ahead and add the additional hydrogen, oxygen, and nitrogen atoms to our molecule below. Feel free to reference the code above for guidance on how to add atoms using `Chem.Atom()`. Remember, glycine contains multiple atoms of some elements!

```
In [94]: # BEGIN SOLUTION NO PROMPT
# Add a second carbon atom
carbon2 = glycine.AddAtom(Chem.Atom(6))      # Carbon atom

# Add remaining hydrogen atoms (we've already added 1, so we add 4 more)
hydrogen2 = glycine.AddAtom(Chem.Atom(1))
hydrogen3 = glycine.AddAtom(Chem.Atom(1))
hydrogen4 = glycine.AddAtom(Chem.Atom(1))
hydrogen5 = glycine.AddAtom(Chem.Atom(1))

# Add a second oxygen atom (we've already added 1)
oxygen2 = glycine.AddAtom(Chem.Atom(8))
# END SOLUTION
""" # BEGIN PROMPT
# Add a second carbon atom
carbon2 = glycine.AddAtom(Chem.Atom(6))      # Carbon atom

# Add remaining hydrogen atoms (we've already added 1, so we add 4 more)
hydrogen2 = glycine.AddAtom(Chem.Atom(...))
hydrogen3 = ...
hydrogen4 = ...
hydrogen5 = ...

# Add a second oxygen atom (we've already added 1)
oxygen2 = ...
""" # END PROMPT
```

```
Out[94]: " # BEGIN PROMPT\n# Add a second carbon atom\ncarbon2 = glycine.AddAtom(Chem.Atom(6))      # Carbon atom\n\n# Add remaining hydrogen atoms (we've already added 1, so we add 4 more)\nhydrogen2 = glycine.AddAtom(Chem.Atom(...))\nhydrogen3 = ...\nhydrogen4 = ...\nhydrogen5 = ...\n\n# Add a second oxygen atom (we've already added 1)\noxygen2 = ...\n"
```

```
In [ ]: grader.check("q37")
```

Question 3.8. We've created the basic scaffolding for our glycine molecule—now it's time to start adding the bonds between the atoms!

To help you get started, we've added the first carbon–nitrogen bond as an example. Using this format, add the remaining bonds to correctly connect all the atoms in glycine based on its structure.

Hint: Think about how the atoms are arranged in glycine's structure ($\text{C}_2\text{H}_5\text{NO}_2$), including the backbone and side groups!

```

In [99]: # BEGIN SOLUTION NO PROMPT
# Define bond type for single bonds
SINGLE = rdchem.BondType.SINGLE
DOUBLE = rdchem.BondType.DOUBLE

# Add bonds to complete the glycine structure

# Backbone: NH2-CH2-COOH
glycine.AddBond(nitrogen, hydrogen, SINGLE)      # N-H
glycine.AddBond(nitrogen, hydrogen2, SINGLE)     # N-H
glycine.AddBond(nitrogen, carbon, SINGLE)         # N-C (first carbon)

glycine.AddBond(carbon, hydrogen3, SINGLE)        # C-H
glycine.AddBond(carbon, hydrogen4, SINGLE)        # C-H
glycine.AddBond(carbon, carbon2, SINGLE)          # C-C (connect to second carbon)

glycine.AddBond(carbon2, oxygen, DOUBLE)          # C=O
glycine.AddBond(carbon2, oxygen2, SINGLE)         # C-O
glycine.AddBond(oxygen2, hydrogen5, SINGLE)       # O-H
# END SOLUTION
""" # BEGIN PROMPT
# Define bond type for single bonds
SINGLE = rdchem.BondType.SINGLE
DOUBLE = rdchem.BondType.DOUBLE

# Add bonds to complete the glycine structure

# Backbone: NH2-CH2-COOH
glycine.AddBond(nitrogen, hydrogen, SINGLE)      # N-H
glycine.AddBond(nitrogen, hydrogen2, SINGLE)     # N-H
glycine.AddBond(nitrogen, carbon, SINGLE)         # N-C (first carbon)

glycine.AddBond(..., hydrogen3, ...)            # C-H
glycine.AddBond(..., ..., ...)                  # C-H
glycine.AddBond(..., carbon2, ...)               # C-C (connect to second carbon)

glycine.AddBond(..., ..., ...)                   # C=O
glycine.AddBond(..., ..., ...)                   # C-O
glycine.AddBond(..., ..., ...)                   # O-H
""" # END PROMPT

```

```

Out[99]: ' # BEGIN PROMPT\n# Define bond type for single bonds\nSINGLE = rdchem.BondType.SINGLE\nDOUBLE = rdchem.BondType.DOUBLE\n\n# Add bonds to complete the glycine structure\n\n# Backbone: NH2-CH2-COOH\nglycine.AddBond(nitrogen, hydrogen, SINGLE)      # N-H\nglycine.AddBond(nitrogen, hydrogen2, SINGLE)     # N-H\nglycine.AddBond(nitrogen, carbon, SINGLE)         # N-C (first carbon)\nglycine.AddBond(..., hydrogen3, ...)            # C-H\nglycine.AddBond(..., ..., ...)                  # C-H\nglycine.AddBond(..., carbon2, ...)               # C-C (connect to second carbon)\nglycine.AddBond(..., ..., ...)                   # C=O\nglycine.AddBond(..., ..., ...)                   # C-O\nglycine.AddBond(..., ..., ...)                   # O-H\n'

```

```

In [ ]: grader.check("q38")

```

Now that we have added all of our molecules, let's try using RDKit to go about visualizing this in 2 dimensions.

Run the cell below.

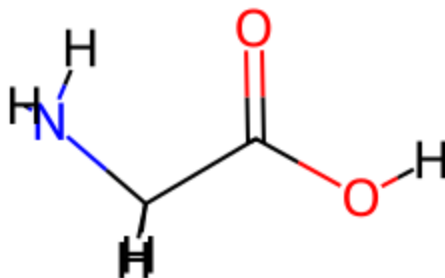
```
In [105]: # Convert to a normal molecule object
glycine = glycine.GetMol()

# Explicitly compute valence & add hydrogens
Chem.SanitizeMol(glycine)
glycine = Chem.AddHs(glycine)

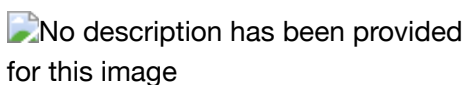
# Generate 3D coordinates
AllChem.EmbedMolecule(glycine)

# Display the molecule
glycine
```

Out[105]:



We've provided a visualization of glycine below — it should look similar to the one you generated above. Try to match the structure as closely as possible!



Since it's a bit difficult to see the full shape of a molecule in 2D, let's take things a step further and explore it in **3D**!

Run the cell below. Please do not modify any of the lines—this cell has been pre-written to generate a specific visualization you'll use to answer the following questions.

```
In [106]: # Function to convert RDKit molecule to 3D (assuming it's defined)
glycine_3d = molecule_to_3d(glycine)

# Create a 3D viewer
view = pm.view(width=400, height=400)

# Load the molecule data
view.addModel(Chem.MolToMolBlock(glycine_3d), "mol")

# Apply enhanced styles
view.setStyle({
    "stick": {"radius": 0.2, "colorscheme": "Jmol"}, # Stick representation with colors
    "sphere": {"scale": 0.5, "colorscheme": "Jmol"}, # Spheres for atoms
})

# Add title label
view.addLabel(
    "Glycine (C2H5NO2)", # Title text
    {"position": {"x": 0, "y": 2, "z": 0}, "backgroundColor": "white", "fontColor": "black", "fontSize": 16}
)

# Adjust background and zoom
view.setBackgroundColor("white")
view.zoomTo()

# Show the visualization
view.show()
```

Perfect! We can now see a 3d representation of our molecule!

In the 3D model above, each atom is represented by a specific color using the **Jmol** color scheme:

- **Carbon (C)** – Gray or black
- **Hydrogen (H)** – White
- **Oxygen (O)** – Red
- **Nitrogen (N)** – Blue

Based on the visualization above, answer the following questions

Question 3.9. Glycine has multiple atoms and functional groups, so instead of a single overall molecular shape, we'll focus on local geometries around key atoms.

- What is the approximate shape around the central carbon atom (the one bonded to both the amine and carboxyl groups)?
- What is the bond angle between the atoms attached to that central carbon?

Assign `mol_shape` to one of the following integers based on your shape selection

1. Linear
2. Trigonal planar
3. Tetrahedral

and assign `angle` to your estimated bond angle (in degrees)

```
In [ ]: view.show()
```



```
In [ ]: mol_shape = 3 #SOLUTION
        angle = 109.5 #SOLUTION
```

```
In [ ]: # BEGIN SOLUTION NO PROMPT
mol_shape = 3 #SOLUTION
angle = 109.5 #SOLUTION
# END SOLUTION
""" # BEGIN PROMPT
mol_shape = ...
angle = ... #SOLUTION
""" # END PROMPT
```

```
Out[ ]: ' # BEGIN PROMPT\nmol_shape = ... \nangle = ... \n'
```

Now that we've identified the local geometry of glycine, let's consider the molecule's overall polarity.

Question 3.10. Based on the electronegativity differences and the 3D structure you visualized, do you think glycine is a polar molecule? Assign the variable `polar` to `True` if you believe glycine is **polar**, or `False` if you think it is **nonpolar**

Hint: Consider whether the polar bonds in glycine are arranged symmetrically or asymmetrically

```
In [209]: # BEGIN SOLUTION NO PROMPT
polar = True
# END SOLUTION
""" # BEGIN PROMPT
polar = ...
""" # END PROMPT
```

```
Out[209]: ' # BEGIN PROMPT\npolar = ... \n'
```

```
In [ ]: grader.check("q310")
```

Question 3.11. Explain your reasoning in the box below

Type your answer here, replacing this text.

SOLUTION: Glycine is a polar molecule because it contains multiple polar bonds and a molecular structure that does not allow the dipoles to cancel out. The carboxyl group ($-\text{COOH}$) and amino group ($-\text{NH}_2$) both contain atoms with significant electronegativity differences (e.g., $\text{C}-\text{O} = 1.0$, $\text{N}-\text{H} = 0.9$), creating strong dipoles. These groups are positioned on opposite sides of the central carbon, resulting in an asymmetric charge distribution across the molecule. As a result, glycine has a net dipole moment, making it overall polar.

Question 3.12. If you place glycine in water (H_2O) and hexane (C_6H_{14}), in which solvent will it dissolve better? Assign the number of your answer choice to the variable `soluble`.

1. Water, because water is polar.
2. Hexane, because hexane is nonpolar.
3. Neither, because glycine has both polar and nonpolar bonds.
4. Both, because polarity doesn't affect solubility.

```
In [157]: # BEGIN SOLUTION NO PROMPT
soluble = 1
# END SOLUTION
""" # BEGIN PROMPT
soluble = ...
""" # END PROMPT
```

```
Out[157]: ' # BEGIN PROMPT\nsoluble = ... \n'
```


```
In [ ]: grader.check("q312")
```

Question 3.10 Explain your reasoning in the box below

SOLUTION: We determined that Glycine is a polar molecule in the previous questions. Water is also a polar solvent, while hexane is nonpolar. According to the principle of “like dissolves like,” polar solutes dissolve better in polar solvents. Therefore, glycine will dissolve better in water than in hexane.

Congratulations!

Teddy 🧸 congratulates you on finishing the Bond Polarity notebook!

 No description has been provided for this image

In this notebook, we

- Learned how to build molecules like water, methane, and glycine using RDKit and explored their 2D and 3D structures
- Investigated electronegativity differences between atoms to identify polar bonds
- Used molecular geometry and visualizations to determine whether a molecule is polar or nonpolar
- Applied concepts like bond angle and molecular symmetry to real-world examples

We hope you enjoyed learning how chemistry and computation come together to help us understand molecular behavior! If you're curious to explore further, consider diving into how molecular polarity affects properties like solubility, boiling points, and biological function. You could even build larger biomolecules or simulate chemical reactions using similar tools. This is just the beginning of what's possible when we combine computation with molecular science!

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

These are some submission instructions.

```
In [ ]: # Save your notebook first, then run this cell to export your submission.  
grader.export(run_tests=True)
```