1. What are some key differences between lists, tuples, and dictionaries in Python, and when would you choose each?

   "A list in Python is ordered and mutable, so I can add, remove, or modify elements in place. A tuple is similar to a list but is immutable—it cannot be changed after creation. It's often used for fixed collections of data or as a key in a dictionary because it's hashable. A dictionary, on the other hand, stores key-value pairs for fast lookups.

   I would use a list if I need to maintain order and possibly alter the collection; a tuple if I need a fixed, unchangeable sequence; and a dictionary if I need to associate a unique key with a value for quick lookups."

2. In Python, a list can store items of different types (e.g., integers, strings, objects) in the same list. Given that C#'s `List<T>` is strongly typed, how would you store heterogeneous data in a single list?

   Answer:
   `List<T>` in C# requires all elements to be of the same type `T`.
   If you need to store different types (heterogeneous data), you can declare a list of a common base type—often `List<object>` or a shared interface/base class if all objects derive from the same hierarchy.

   e.g.
   // A list that can hold any type because it uses 'object'

   List<object> heterogeneousList = new List<object>

```
{
42, // int
"Hello World", // string
3.14159, // double
new Person("Alice") // a custom class
};
```

3. I have 2 two input boxes on a screen, and a submit button. When I click the button, text entered in the box1, will be used to query a db for some row of data. The returned data would fill box 2. We do not want box 2 to allow user manual input:
   A. How would we prevent manual edits of box 2?
   B. We do not want the user to press the button more than once, to prevent multiple queries from a client before a response is received. How would you implement this (higher level design)

4. In dash, Input triggers callbacks, while States are only used when callback is called.

   import dash from dash import html, dcc, Input, Output, State

   app = dash.Dash(__name__)
   app.layout = html.Div([dcc.Input(id='name-input', type='text', placeholder='Enter your name'),
   html.Button('Submit', id='submit-button'),
   html.Div(id='greeting-output')])

```
@app.callback(
    Output('greeting-output', 'children'),
    Input('submit-button', 'n_clicks'),
    State('name-input', 'value')
)
def update_greeting(n_clicks, name):
    if n_clicks is None:
        # Before the button is ever clicked
        return "No greeting yet!"
    else:
        # Update greeting after the button is clicked
        return f"Hello, {name}!"
```

```
@app.callback(
    Output('greeting-output', 'children'),       # We want to update this output
    Input('submit-button', 'n_clicks'),          # The callback fires on button clicks
    Input('name-input', 'value')                 # The current text is read only when button is clicked
)
def update_greeting(n_clicks, name):
    if n_clicks is None:
        # Before the button is ever clicked
        return "No greeting yet!"
    else:
        # Update greeting after the button is clicked
        return f"Hello, {name}!"
```

Which of the two would have unexpected results, why?

5.

You're building a web application (using Plotly Dash) that displays two input boxes and a **Submit** button.

- **Box 1**: Accepts user input (enabled for typing).
- **Box 2**: Displays data fetched from the database (read-only).
- **Submit button**: When clicked, it triggers a Python callback that:
    A. Read the value from **Box 1**.
    B. Fetches corresponding data from an SQL database.
    C. Returns the data to populate **Box 2**.

Because Dash executes callbacks on the server side and does not allow partial/intermediate UI updates within a single callback, you need to ensure that:

The user **cannot click Submit again** while the system is still fetching data.

```
from dash import html, dcc, Input, Output, State, no_update

app.layout = html.Div([ dcc.Input(id='box1', type='text', placeholder='Enter Text'),
dcc.Input(id='box2', type='text', placeholder='Enter Text'),
html.Button('Submit', id='submit-button'),
@app.callback(
    Output('box2', 'value'),       # We want to update this output
    Input('submit-button', 'n_clicks'),          # The callback fires on button clicks
    State('box1', 'value')                  # The current text is read only when button is clicked
)
def update_ui(n_clicks, box1_val):
    if n_clicks is None:
        # Before the button is ever clicked
        return no_update
    else:
        data = get_info(box1_value)
        return data
```

```
dcc.Loading(id="ls-loading-2",children=[html.Div([html.Div(id="
ls-loading-output-2")])],type="circle")
```

6. In python, what is the difference between a list comprehension and a generator expression?

   Answer:
   List Comprehension: Creates a **list** in memory immediately.
   Generator Expression: Creates a **generator** that yields items **on-the-fly**, consuming less memory when dealing with large or infinite sequences.

   B. What is the main syntax difference?

my_list = [x * 2 for x in range(5)]

my_gen = (x * 2 for x in range(5))

C. When would you use one over the other?

7. What is the difference between `git fetch` and `git pull`, and when would you use each command?

   “`git fetch` simply downloads commits, files, and references from the remote repository but doesn’t integrate them into my local branches. It lets me see what changes are available without affecting my working copy. `git pull` does a `fetch` and then merges (or rebases) those changes into my current local branch. I typically use `git fetch` when I want to review the latest commits before deciding if or how to merge, and `git pull` once I’m ready to bring those changes into my local branch.”

8.