

Section: Getting Started

ChatGPT Query: I want to create 100 text files. Each text file should contain a different boy's name. Write me a Python script for that.

```
import os

# list of 100 boy names
boy_names = ["Noah", "Liam", "Ethan", "Oliver", "Aiden", "Lucas", "Mason", "Logan", "Elijah", "Ethan",
"Caden", "Jacob", "Jackson", "Jayden", "Daniel", "Matthew", "William", "James", "Michael", "Benjamin",
"Ryan", "Luke", "Alexander", "Gabriel", "Owen", "Nicholas", "Nathan", "David", "Andrew", "Caleb",
"Dylan", "Samuel", "Christian", "Isaac", "Joshua", "Joseph", "Adam", "Anthony", "Isaiah", "Eli",
"Aaron", "Connor", "Jonathan", "Landon", "Evan", "Thomas", "Avery", "Cameron", "Christopher", "Colton",
"Brandon", "Dominic", "Tyler", "Jaxon", "Zachary", "Nathaniel", "Adrian", "Wyatt", "Julian", "Austin",
"Steven", "Carlos", "Tristan", "Jason", "Ian", "Josiah", "Cole", "Xavier", "Parker", "Chase", "Kevin",
"Levi", "Kayden", "Blake", "Alex", "Vincent", "Maxwell", "Ryder", "Eric", "Miles", "Jace", "Gavin",
"Maddox", "Easton", "Kingston", "Hudson", "Brandon", "Bentley", "Cooper", "Maverick", "Justin", "Greyson",
"Damian", "Leo", "Declan", "Ezra", "Max", "Axel", "Braxton", "Everett", "Silas", "Lincoln"]

# create a directory to store the text files
directory = "boy_names"
if not os.path.exists(directory):
    os.makedirs(directory)

# create 100 text files with boy names
for i in range(1, 101):
    file_name = f"{directory}/boy_name_{i}.txt"
    with open(file_name, "w") as f:
        f.write(boy_names[i-1])
```

Lecture: Modifying Text Files in Bulk with ChatGPT

Generated Code:

```
import os

# Define the directory where your text files are located
directory = 'files/'

# Iterate over each file in the directory
for filename in os.listdir(directory):
    if filename.endswith('.txt'):
        # Read in the contents of the file
        with open(directory + filename, 'r') as file:
            text = file.read()

        # Split the text into sentences
        sentences = text.split('.') )
```

```

# Capitalize the first letter of each sentence
sentences = [s.capitalize() for s in sentences]

# Join the sentences back together
text = '. '.join(sentences)

# Write the modified text back to the file
with open(directory + filename, 'w') as file:
    file.write(text)

```

Lecture: Merge Text Files with ChatGPT

ChatGPT Query: I have multiple text files in a folder named "files". Each text file contains some text. I want to merge those text files into one single text file. Write me a Python script for that.

Generated Code:

```

import os

# Set the directory path where the text files are located
directory_path = "files/"

# Set the name of the output file
output_file = "merged_text.txt"

# Create an empty list to store the text from each file
file_text = []

# Loop through each file in the directory
for filename in os.listdir(directory_path):
    # Make sure we're only looking at text files
    if filename.endswith(".txt"):
        # Open the file and add its text to the list
        with open(directory_path + filename, "r") as file:
            file_text.append(file.read())

# Merge the text from all the files into one string
merged_text = "\n".join(file_text)

# Write the merged text to a new file
with open(output_file, "w") as file:
    file.write(merged_text)

```

Lecture: Streamline Excel Processing with ChatGPT

ChatGPT Query: I have multiple Excel files in a folder named "excel_files". Each excel file contains two columns, an "employee" column and a "monthly salary" column. I want to add a new "annual salary" column to each Excel file. The "annual salary" column should be equal to "monthly salary" x 12. Write me a Python script for that.

Generated Code:

```

import os
import pandas as pd

# Define the folder path where Excel files are stored
folder_path = "excel_files"

# Loop through each Excel file in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.xlsx'): # Check if the file is an Excel file
        # Read the Excel file into a pandas dataframe
        filepath = os.path.join(folder_path, filename)
        df = pd.read_excel(filepath)

```

```
# Add the "annual salary" column to the dataframe
df['annual salary'] = df['monthly salary'] * 12

# Save the updated dataframe back to the Excel file
writer = pd.ExcelWriter(filepath, engine='xlsxwriter')
df.to_excel(writer, index=False)
writer.save()
```

Lecture: Merging Excel Files with ChatGPT

ChatGPT Query: I have multiple Excel files in a directory named "excel_files". They all contain the same columns. I want to merge the excel files into one single Excel file containing the data of all files. Write me a Python script for that.

Generated Code:

```
import os
import pandas as pd

# set the directory where the Excel files are located
directory = "excel_files"

# create an empty list to store the data from all files
all_data = []

# loop through each file in the directory
for filename in os.listdir(directory):
    if filename.endswith(".xlsx"): # only process Excel files
        # read the Excel file into a pandas DataFrame
        filepath = os.path.join(directory, filename)
        data = pd.read_excel(filepath)
        # append the data to the all_data list
        all_data.append(data)

# concatenate all the data into a single DataFrame
merged_data = pd.concat(all_data, ignore_index=True)

# save the merged data to a new Excel file
merged_data.to_excel("merged_file.xlsx", index=False)
```

Lecture: Processing CSV Files in Bulk with ChatGPT

ChatGPT Query: I have a "stockholm.csv" file. The file has a TG column. I want to divide all values of that column by 10. Write me a Python script for that.

Code:

```
import csv

# Open the input and output files
with open('stockholm.csv', 'r') as infile, open('stockholm_updated.csv', 'w', newline='') as outfile:
    reader = csv.DictReader(infile)
    writer = csv.DictWriter(outfile, fieldnames=reader.fieldnames)

    # Write the headers to the output file
    writer.writeheader()

    # Iterate over each row in the input file and write the updated row to the output file
    for row in reader:
        row['TG'] = str(float(row['TG'])/10) # Convert the value to float, divide by 10, and convert back to string
        writer.writerow(row)
```

Lecture: Rule-Based ZIP Generation with ChatGPT

Query: I have 400 text files in a directory named "text_files". The files are named like "TG_STAID000001.txt", "TG_STAID000002.txt", and so on. I want to place files "TG_STAID000001.txt" to "TG_STAID000010.txt" in one separate ZIP file, "TG_STAID000011.txt" to "TG_STAID000020.txt" in another separate ZIP file, and so on. Write me a Python script for that.

Generated Code:

```
import os
import zipfile

# Define the directory where the text files are stored
dir_path = "text_files/"

# Define the size of each batch of files to be zipped
batch_size = 10

# Get a list of all the text files in the directory
files = os.listdir(dir_path)

# Sort the files by their numeric suffix
files.sort(key=lambda x: int(x.split("TG_STAID")[1].split(".")[0]))

# Create a counter variable to keep track of the batch number
batch_count = 1

# Loop through the files, creating a new zip file every batch_size files
for i in range(0, len(files), batch_size):
    batch_files = files[i:i+batch_size]
    zip_file_name = f"batch{batch_count}.zip"
    with zipfile.ZipFile(zip_file_name, "w") as zip_file:
        for file_name in batch_files:
            file_number = int(file_name.split("TG_STAID")[1].split(".")[0])
            if (batch_count-1)*batch_size < file_number <= batch_count*batch_size:
                file_path = os.path.join(dir_path, file_name)
                zip_file.write(file_path, file_name)
    batch_count += 1
```

Section: Using ChatGPT for Data Analysis and Visualization

Lecture: Generate Histogram and Descriptive Statistics with ChatGPT

ChatGPT Query: I have a "stockholm_updated.csv". The file has a "TG" column which contains the temperature observations in degree celsius. I want to create a histogram for that column. Write me a script using Streamlit and Python.

GeneratedCode:

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv("stockholm_updated.csv")

# Define the histogram function
def histogram(data):
    fig, ax = plt.subplots()
    ax.hist(data['TG'], bins=20)
    ax.set_xlabel('Temperature (°C)')
    ax.set_ylabel('Frequency')
    ax.set_title('Temperature Distribution')
    st.pyplot(fig)

# Create the Streamlit app
st.title('Temperature Histogram')
```

```
st.write('Histogram of temperature observations in Stockholm')

# Show the histogram
histogram(data)
```

Lecture: Aggregate and Plot Timeseries Data with ChatGPT

ChatGPT Query: The "stockholm_updated.csv" file also has a "DATE" column. That column contains the temperature observation date in the format "YYYYMMDD". I want to aggregate the temperatures by year, and then plot the yearly temperatures across time. Write me a script using Streamlit and Python.

Generated Code:

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv("stockholm_updated.csv")

# Convert the DATE column to datetime format and extract the year
data["YEAR"] = pd.to_datetime(data["DATE"], format="%Y%m%d").dt.year

# Group the data by year and calculate the mean temperature
yearly_data = data.groupby("YEAR")["TG"].mean().reset_index()

# Create the Streamlit app
st.title('Yearly Temperature Trends')
st.write('Average yearly temperatures in Stockholm from 1756 to 2021')

# Create the plot
fig, ax = plt.subplots()
ax.plot(yearly_data["YEAR"], yearly_data["TG"])
ax.set_xlabel("Year")
ax.set_ylabel("Temperature (°C)")
ax.set_title("Yearly Temperature Trends")
st.pyplot(fig)
```

ChatGPT Query Part 2: I want a bar graph. Rewrite the script.

Generated Code Part 2:

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv("stockholm_updated.csv")

# Convert the DATE column to datetime format and extract the year
data["YEAR"] = pd.to_datetime(data["DATE"], format="%Y%m%d").dt.year

# Group the data by year and calculate the mean temperature
yearly_data = data.groupby("YEAR")["TG"].mean().reset_index()

# Create the Streamlit app
st.title('Yearly Temperature Trends')
st.write('Average yearly temperatures in Stockholm from 1756 to 2021')

# Create the plot
fig, ax = plt.subplots()
ax.bar(yearly_data["YEAR"], yearly_data["TG"])
ax.set_xlabel("Year")
ax.set_ylabel("Temperature (°C)")
ax.set_title("Yearly Temperature Trends")
st.pyplot(fig)
```

Lecture: Build Interactive Data Dashboards with ChatGPT

ChatGPT Query: The "stockholm_updated.csv" file also has a "DATE" column. That column contains the temperature observation date in the format "YYYYMMDD". I want to aggregate the temperatures by year, and then plot the yearly temperatures across time using a bar graph. I also want a drop down list widget to let users choose between celsius and fahrenheit and plot the data based on the user selection. Write me a script using Streamlit and Python.

Generated Code:

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv("stockholm_updated.csv")

# Convert the DATE column to datetime format and extract the year
data["YEAR"] = pd.to_datetime(data["DATE"], format="%Y%m%d").dt.year

# Group the data by year and calculate the mean temperature in Celsius
yearly_data_celsius = data.groupby("YEAR")["TG"].mean().reset_index()

# Convert Celsius to Fahrenheit
yearly_data_fahrenheit = yearly_data_celsius.copy()
yearly_data_fahrenheit["TG"] = yearly_data_fahrenheit["TG"] * 9/5 + 32

# Create the Streamlit app
st.title('Yearly Temperature Trends')
st.write('Average yearly temperatures in Stockholm from 1756 to 2021')

# Create the drop down list
units = ["Celsius", "Fahrenheit"]
unit = st.sidebar.selectbox("Select unit", units)

# Choose the data based on the user selection
if unit == "Celsius":
    yearly_data = yearly_data_celsius
    ylabel = "Temperature (°C)"
else:
    yearly_data = yearly_data_fahrenheit
    ylabel = "Temperature (°F)"

# Create the plot
fig, ax = plt.subplots()
ax.bar(yearly_data["YEAR"], yearly_data["TG"])
ax.set_xlabel("Year")
ax.set_ylabel(ylabel)
ax.set_title("Yearly Temperature Trends")
st.pyplot(fig)
```

Section: Using ChatGPT to Build Web Apps

Lecture: Build Web Apps with ChatGPT

ChatGPT Query: I want to create a word counter web app. Write me a web app using Flask and Python.

Generated Python Code:

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/', methods=['POST'])
def count_words():
```

```

text = request.form['text']
word_count = len(text.split())
return render_template('index.html', word_count=word_count)

if __name__ == '__main__':
    app.run(debug=True)

```

Generated HTML Code:

```

<!DOCTYPE html>
<html>
<head>
  <title>Word Counter</title>
</head>
<body>
  <h1>Word Counter</h1>
  <form action="/" method="POST">
    <label for="text">Enter text:</label>
    <textarea id="text" name="text"></textarea>
    <br>
    <input type="submit" value="Count words">
  </form>
  {% if word_count %}
    <p>{{ word_count }} words</p>
  {% endif %}
</body>
</html>

```

Improved ChatGPT Query:

The app looks very basic on the frontend. Add some bootstrap to make it look more modern.

Improved Generated HTML code:

```

/* Set background color and font family */
body {
  background-color: #F8F9FA;
  font-family: Arial, sans-serif;
}

/* Center the form and the word count */
.form-container {
  margin: 50px auto;
  max-width: 600px;
}

/* Add some spacing between elements */
.form-group {
  margin-bottom: 20px;
}

/* Set the width of the text area */
.form-control {
  width: 100%;
}

/* Set the color of the button text and background */
.btn-primary {
  color: #FFFFFF;
  background-color: #007BFF;
  border-color: #007BFF;
}

.btn-primary:hover {
  color: #FFFFFF;
  background-color: #0062CC;
  border-color: #005CBF;
}

/* Set the color of the word count */
.word-count {
  font-size: 20px;
  font-weight: bold;
  color: #007BFF;
}

```

Lecture: Build a Quote Generation Web App with ChatGPT

ChatGPT Query:

I have a quotes.txt file. That file contains quotes. The file structure looks like this:

"The best way to predict the future is to invent it." - Alan Kay

"Life is 10% what happens to you and 90% how you react to it." - Charles R. Swindoll

"Believe you can and you're halfway there." - Theodore Roosevelt

"In the end, we will remember not the words of our enemies, but the silence of our friends." - Martin Luther King Jr.

"Success is not final, failure is not fatal: It is the courage to continue that counts." - Winston Churchill

I want to create a web app that displays one of the quotes of the "quotes.txt" file whenever the web app is visited. Write a web app using Flask and Python.

Generated Code:

```
from flask import Flask, render_template
import random

app = Flask(__name__)

def get_random_quote():
    with open("quotes.txt", "r") as f:
        quotes = f.readlines()
    return random.choice(quotes).strip()

@app.route('/')
def index():
    quote = get_random_quote()
    return render_template('index.html', quote=quote)

if __name__ == '__main__':
    app.run(debug=True)
```

Section: Using ChatGPT to Build Desktop GUI Apps

Lecture: Your First Desktop GUI App

ChatGPT Query: I want to create a temperature converter desktop GUI app. In the app, users can convert temperatures from celsius to fahrenheit and vice-versa. Build me a program for that using PyQt and Python.

Generated Code:

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton

class TemperatureConverter(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Temperature Converter')
        self.setGeometry(100, 100, 300, 150)

        self.celsius_label = QLabel(self)
        self.celsius_label.setText('Celsius')
        self.celsius_label.move(20, 20)

        self.fahrenheit_label = QLabel(self)
```



```

self.fahrenheit_label.setText('Fahrenheit')
self.fahrenheit_label.move(20, 60)

self.celsius_input = QLineEdit(self)
self.celsius_input.move(100, 20)

self.fahrenheit_input = QLineEdit(self)
self.fahrenheit_input.move(100, 60)

self.celsius_to_fahrenheit_button = QPushButton(self)
self.celsius_to_fahrenheit_button.setText('Convert to Fahrenheit')
self.celsius_to_fahrenheit_button.move(20, 100)
self.celsius_to_fahrenheit_button.clicked.connect(self.celsius_to_fahrenheit)

self.fahrenheit_to_celsius_button = QPushButton(self)
self.fahrenheit_to_celsius_button.setText('Convert to Celsius')
self.fahrenheit_to_celsius_button.move(160, 100)
self.fahrenheit_to_celsius_button.clicked.connect(self.fahrenheit_to_celsius)

def celsius_to_fahrenheit(self):
    celsius = float(self.celsius_input.text())
    fahrenheit = (celsius * 9/5) + 32
    self.fahrenheit_input.setText(str(fahrenheit))

def fahrenheit_to_celsius(self):
    fahrenheit = float(self.fahrenheit_input.text())
    celsius = (fahrenheit - 32) * 5/9
    self.celsius_input.setText(str(celsius))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    converter = TemperatureConverter()
    converter.show()
    sys.exit(app.exec_())

```

ChatGPT Query Part 2:

The program you built has the two buttons too close to each other. Rewrite the program fixing that issue.

Improved Generated Code:

```

import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QHBoxLayout, QVBoxLayout

class TemperatureConverter(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Temperature Converter')
        self.setGeometry(100, 100, 300, 150)

        self.celsius_label = QLabel(self)
        self.celsius_label.setText('Celsius')
        self.celsius_label.move(20, 20)

        self.fahrenheit_label = QLabel(self)
        self.fahrenheit_label.setText('Fahrenheit')
        self.fahrenheit_label.move(20, 60)

        self.celsius_input = QLineEdit(self)
        self.celsius_input.move(100, 20)

        self.fahrenheit_input = QLineEdit(self)
        self.fahrenheit_input.move(100, 60)

        self.celsius_to_fahrenheit_button = QPushButton(self)
        self.celsius_to_fahrenheit_button.setText('Convert to Fahrenheit')
        self.celsius_to_fahrenheit_button.clicked.connect(self.celsius_to_fahrenheit)

        self.fahrenheit_to_celsius_button = QPushButton(self)
        self.fahrenheit_to_celsius_button.setText('Convert to Celsius')
        self.fahrenheit_to_celsius_button.clicked.connect(self.fahrenheit_to_celsius)

        button_layout = QHBoxLayout()

```

```

        button_layout.addWidget(self.celsius_to_fahrenheit_button)
        button_layout.addWidget(self.fahrenheit_to_celsius_button)

    main_layout = QVBoxLayout()
    main_layout.addWidget(self.celsius_label)
    main_layout.addWidget(self.celsius_input)
    main_layout.addWidget(self.fahrenheit_label)
    main_layout.addWidget(self.fahrenheit_input)
    main_layout.addLayout(button_layout)

    self.setLayout(main_layout)

def celsius_to_fahrenheit(self):
    celsius = float(self.celsius_input.text())
    fahrenheit = (celsius * 9/5) + 32
    self.fahrenheit_input.setText(str(fahrenheit))

def fahrenheit_to_celsius(self):
    fahrenheit = float(self.fahrenheit_input.text())
    celsius = (fahrenheit - 32) * 5/9
    self.celsius_input.setText(str(celsius))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    converter = TemperatureConverter()
    converter.show()
    sys.exit(app.exec_())

```

Lecture: Excel Viewer GUI App

ChatGPT Query: I want to create an Excel file viewer desktop GUI app. In the app, users can browse an Excel file in their file system, and press a "View" button. The "View" button should display the data of the Excel file in a table view widget. Build me a program for that using PyQt and Python.

Generated Code:

```

import sys
import pandas as pd
from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QMessageBox, QTableWidgetItem, QTableWidgetItem, QVBoxLayout, QWidget, QHBoxLayout

class ExcelViewer(QMainWindow):
    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):
        # Create the main window
        self.setWindowTitle("Excel Viewer")
        self.setGeometry(100, 100, 800, 600)

        # Create a label to display the selected file name
        self.selected_file_label = QLabel("No file selected")
        self.selected_file_label.setStyleSheet("font-size: 20px")
        self.selected_file_label.setAlignment(Qt.AlignCenter)

        # Create a table widget to display the data
        self.table_widget = QTableWidgetItem()
        self.table_widget.setEditTriggers(QTableWidgetItem.NoEditTriggers)

        # Create a "Browse" button to select the Excel file
        browse_button = QPushButton("Browse")
        browse_button.clicked.connect(self.browse_file)

        # Create a "View" button to display the data in the table widget
        view_button = QPushButton("View")
        view_button.clicked.connect(self.view_file)

        # Create a horizontal box layout for the buttons
        button_layout = QHBoxLayout()
        button_layout.addWidget(browse_button)
        button_layout.addWidget(view_button)

```

```

# Create a vertical box layout for the main window
main_layout = QVBoxLayout()
main_layout.addWidget(self.selected_file_label)
main_layout.addWidget(self.table_widget)
main_layout.addLayout(button_layout)

# Create a widget to hold the main layout
widget = QWidget()
widget.setLayout(main_layout)

# Set the main widget of the main window
self.setCentralWidget(widget)

def browse_file(self):
    # Open a file dialog to select an Excel file
    file_path, _ = QFileDialog.getOpenFileName(self, "Open Excel File", "", "Excel Files (*.xlsx *.xls)")

    # Update the selected file label with the file name
    if file_path:
        self.selected_file_label.setText(file_path)

def view_file(self):
    # Get the selected file path
    file_path = self.selected_file_label.text()

    if not file_path:
        # Show an error message if no file is selected
        QMessageBox.warning(self, "Error", "No file selected")
        return

    try:
        # Read the Excel file using pandas
        df = pd.read_excel(file_path)

        # Set the number of rows and columns in the table widget
        num_rows, num_cols = df.shape
        self.table_widget.setRowCount(num_rows)
        self.table_widget.setColumnCount(num_cols)

        # Populate the table widget with the data
        for i in range(num_rows):
            for j in range(num_cols):
                item = QTableWidgetItem(str(df.iloc[i, j]))
                self.table_widget.setItem(i, j, item)

        # Resize the columns to fit the data
        self.table_widget.resizeColumnsToContents()

    except Exception as e:
        # Show an error message if the file cannot be read
        QMessageBox.warning(self, "Error", f"Unable to read file: {e}")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = ExcelViewer()
    ex.show()
    sys.exit(app.exec_())

```

After Error ChatGPT Query: Your code gave me this error:

NameError: name 'Qt' is not defined

Steps to correct the error:

1. To fix the error, you need to import the `Qt` module from `PyQt5`.

Add the following import statement at the top of the file:

```
from PyQt5.QtCore import Qt
```

2. Then, replace `AlignCenter` with `AlignCenter` in the `ExcelViewer` class:

```
self.selected_file_label.setAlignment(Qt.AlignCenter)
```

