Application to sign up/verify appointment information

1. reading the file path and converting it to a df for later use. Using try except so if an issue occurs it wont start the application and cause a crash further down the line.
2. In future models I need to use a different library to save the work, its current format removes all formatting and changes to cells in regards to excel sheets

In [ ]:
```
import pandas as pd
from datetime import datetime, timedelta
import numpy as np
from openpyxl import load_workbook


pricing_map = {
        'Tape': 15, "Haircut": 25, 'Wash': 50, 'Silkpress': 175,
        'Beardtrim': 20, 'Fade': 35, 'Shapeup': 10, 'Braid': 125, 'Twist': 235
    }


try:
  file_name = "C:Youre\\file\\path"
  df = pd.read_excel("C:Youre\\file\\path")
except FileNotFoundError:
  print("FileNotFoundError")
except Exception as e:
  print("{}".format(e))


services = list(df['Service'].unique())
services = ', '.join(services)
print(services)
```

Introduction:

Gathers user input. Later in script in conjunction with another function it will be the catalyst to assign which function will run. On its own just checks if user wants to quit and acts accordingly

In [ ]:
```
def intro():
    while True:
        intro = input("Hi, press (1) to quit. If you have an appointment type (2), if you need to make an appoi
        try:
            intro = int(intro)
            return intro
        except ValueError:
            print("Please enter a valid number.")
```

Check For Appointments:

function that checks and confirms user appointments based off of inputted ID:

In [ ]:
```
def check_appointment():
    while True:
        app_id = input("Press 'Q' to exit or Enter your appointment ID: ")
        if app_id.upper() == 'Q':
            print("Goodbye!")
            return

        try:
            app_id = int(app_id) #convert user input to int

            #originally had it as .values() but switched for any for more flexibility
            if (df['Appointment_ID'] == app_id).any(): #checks if any matches in the df['Appointment_ID'] colum

                #getting information associated with account
                user = df[df['Appointment_ID'] == app_id]
                user_name = user['Name'].iloc[0]
                user_date = user['Date']
                print("Welcome {}".format(user_name))
                print("Your appointment is on {} at {}",format(user_name, user_date))
                while True:
                    try:                #confirming appointment block
                        confirmation = input("Type 1 to confirm your appointment and 2 to cancel: ")
                        confirmation = int(confirmation)
                        if confirmation == 1:
                            print("You've now been confirmed.")
                            app_id_index = user.index[0]
                            df.at[app_id_index, 'Confirmed'] = 1
                            df.to_excel(file_name, index=False) #saving so change can be reflected in excel shee
                            return
                        elif confirmation == 2:
                            print("Call or use the application to reschedule your appointment.")
                            app_id_index = user.index[0]
                            df.at[app_id_index, 'Confirmed'] = 0
                            df.to_excel(file_name, index = False) #saving so change can be reflected in excel sl
                            return
                        else:
                            print("Invalid option, please type 1 or 2.")
                    except ValueError:
                        print("Please enter a valid number.")
            else:
                print("Appointment ID not found.")
        except ValueError:
            print("Please enter a valid number.")
```

Create New Appointments

Function that generates new user appointments by gathering information required.

In [ ]:
```
def create_appointment():
    global df
    print("Let's create a new appointment.")

    while True:                         #capitalized everything to keep uniformity
        new_user_name = input('What is your name: ').capitalize()
        if all(name.isalpha() or name.isspace() for name in new_user_name): #making sure input is either a lett
            break
        else:
            print("The name should only contain letters. Please try again.")

                            #getting user preferred appointment date
    while True:
        new_user_date_str = input("{}, set an appointment between Monday and Saturday(format MM/DD/YYYY): ".forr
        try:    # formatting the earlier input to properly formatted date as long as the day isnt a sunday
            new_user_date = datetime.strptime(new_user_date_str, "%m/%d/%Y")
            if 0 <= new_user_date.weekday() < 6:  # Check if it's between Monday and Saturday
                break
            else:
                print("Appointments cannot be set on Sundays. Please choose another day.")
        except ValueError:
            print("Invalid date format. Please follow the format MM/DD/YYYY.")

                    #getting users preferred appointment time(theres an issue with the way the time translates
                    #currently working on it) excel has a difficult time formatting as a time.
    while True:
        new_user_time_str = input("Hours of operation are between 8:00am to 3:00pm, what time works for you (HH
        try: #same process as earlier formatting the user time as a time
            new_user_time = datetime.strptime(new_user_time_str, "%H:%M").time()
            if datetime.combine(new_user_date, new_user_time): #combining the two to make new column ---->
                break
        except ValueError:
            print("Invalid time format. Please follow the format HH:MM in 24-hour format.")

     #newly created column based off user inputted time and date
    new_user_datetime = datetime.combine(new_user_date, new_user_time)



                                #Selecting service
    #implementing pricing map
    pricing_map = {
        'Tape': 15, "Haircut": 25, 'Wash': 50, 'Silkpress': 175,
        'Beardtrim': 20, 'Fade': 35, 'Shapeup': 10, 'Braid': 125, 'Twist': 235
    }
    services = list(pricing_map.keys())
    print("What service are you interested in:")

    #using enumerate to create pairs(if i didnt pass the parameter to start from 1 it would start from 0. I add
    #later in the function
    for number, service in enumerate(services, start = 1):

        print("{}. {}: ${}".format(number, service, pricing_map[service]))

    while True:
        try:
            service_choice = input("Please select a service by entering the number: ")
            service_choice = int(service_choice)
            if 1 <= service_choice <= len(services): #check to make sure user choice is in range
                new_user_service = services[service_choice - 1] #subtracting 1 so that it lines up accordingly(
                new_user_price = pricing_map[new_user_service] #applying the map to get user price
                break
            else:
                print("Invalid selection. Please choose a number from the list.")
        except ValueError:
            print("Please enter a number.")


            #pricing map to avoid having to manually implement prices
    pricing_map = {'Tape': 15, "Haircut": 25, 'Wash': 50, 'Silkpress':175, 'Beardtrim': 20, 'Fade': 35,'Shapeup
    new_user_price = pricing_map.get(new_user_service, "Service not found")
    if new_user_price == "Service not found":
        print("Invalid service selected. Please start over.")
        return

        #Payment method(if i was thinking a bit more would've made this a map like i did the service chart and
    new_user_payment = input("How will you be paying (Cash, Zelle, Applepay): ").capitalize()

        #Contact information
    new_user_contact = input("What email or number can we reach you regarding your appointment, to skip press (
    if new_user_contact == "1":
        new_user_contact = np.nan #place holder if no contact found

    # Create new df based on user response
    new_appointment = pd.DataFrame({

        #new appointment id will be one more than most recent entry in original df
        "Appointment_ID": [df['Appointment_ID'].max() + 1],

        #just plugging in information gathered earlier
        'Name': [new_user_name],
        'MonthDate': [new_user_date.strftime('%m/%d/%Y')],
        'Time': [new_user_time_str],
        'Service': [new_user_service],
        'Payment': [new_user_payment],
        'DateTime': [new_user_datetime],
        'Price': [new_user_price],
        'Contact': [new_user_contact],
         #confirmation starts as 0 until user goes back and confirms
        'Confirmed': [0],
        'Cancelled/Missed':[0],
        'Contacted': [0]
    })

    #adding new row(new_appointment) to the original df
    df = pd.concat([df, new_appointment], ignore_index=True)

    try:
        # Load the existing workbook
        book = load_workbook(file_name)

        # Write to the existing workbook
        with pd.ExcelWriter(file_name, engine='openpyxl') as writer:
            writer.book = book

            # Copy existing sheets
            writer.sheets = {ws.title: ws for ws in book.worksheets}

            # Write out the new data
            df.to_excel(writer, index=False, sheet_name='Table1')

            # Save the workbook
            writer.save()
        print(f"{new_user_name}, you're booked! Your confirmation number is {new_appointment['Appointment_ID'][
    except PermissionError:
        print("Error saving your appointment, please try again later.")
```

In [ ]:

```
def modify_appointment():
    global df, file_name  # Reference the global variables

    try:
        user_id = int(input("Enter your user ID: "))
        if user_id not in df['Appointment_ID'].values:
            print("You don't currently have an appointment, I'll bring you back to the main menu where you can
            return
    except ValueError:
        print("Please enter a valid number.")
        return

    user = df[df['Appointment_ID'] == user_id]
    if user.empty:
        print("Appointment ID not found.")
        return
    user_index = user.index[0]
    change = input("To change service type (1), for time (2) and for date (3): ")
    if change == "1":
        print("Current service:", user['Service'].iloc[0])
        new_service = input("Enter the new service: ")
        df.at[user_index, 'Service'] = new_service
    elif change == "2":
        print("Current time:", user['Time'].iloc[0])
        new_time = input("Enter the new time (HH:MM): ")
        df.at[user_index, 'Time'] = new_time
    elif change == "3":
        print("Current date:", user['Date'].iloc[0].strftime('%m/%d/%Y'))
        new_date_str = input("Enter the new date (MM/DD/YYYY): ")
        new_date = datetime.strptime(new_date_str, "%m/%d/%Y").date()
        df.at[user_index, 'Date'] = new_date.strftime('%Y-%m-%d')
    else:
        print("Invalid option selected.")
        return

    # Save the changes back to the Excel file
    df.to_excel(file_name, index=False)
    print("Appointment updated successfully.")
```

In [ ]:
```
def cancel_appointment():
    global df, file_name  # Reference the global variables

    try:
        user_id = int(input("Enter your user ID to cancel your appointment: "))
        if not df['Appointment_ID'].isin([user_id]).any():
            print("Appointment ID not found, returning to the main menu.")
            return
    except ValueError:
        print("Please enter a valid number.")
        return

    # Confirm cancellation
    confirm = input("Are you sure you want to cancel your appointment? Type 'yes' to confirm: ")
    if confirm.lower() != 'yes':
        print("Cancellation aborted. No changes made.")
        return

    # marking the appointment as canceled/missed
    df.loc[df['Appointment_ID'] == user_id, 'Cancelled/Missed'] = 1

    # Save the updated DataFrame back to the Excel file
    df.to_excel(file_name, index=False)
    print("Your appointment has been successfully marked as canceled.")
```

Application:

Combination of all created formulas.

In [ ]:
```
def application():
    intro_choice = intro()
    if intro_choice == 1:
        print("Thank you, come back soon!")
        return
    elif intro_choice == 2:
        check_appointment()
    elif intro_choice == 3:
        create_appointment()
    elif intro_choice == 4:
        modify_appointment()
    else:
        print("That's not a valid option, please try again.")
```

## Running the application

In [ ]:
```
application()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js