```python
from cs50 import SQL
from flask import redirect, render_template, url_for, request

db = SQL("sqlite:///tripplanner.db")


# For errors found throughout the doc regarding numbers (cash).
def error(amount):
    # Return error with custom message if negative number
    if amount < 0:
        positive = "Please enter a valid amount value. Only positive values."
        return render_template("error.html", errortype=positive)
    # Return error with custom message if zero
    elif amount == 0:
        zero = "Please enter some positive value."
        return render_template("error.html", errortype=zero)
    # Return error with custom message if expense is greater than current cash
    else:
        notenough = "Not enough money! Please enter more on allowance."
        return render_template("error.html", errortype=notenough)
```

```python
# This project is inspired by "C$50" Finance from CS50.
# https://docs.cs50.net/2017/ap/problems/finance/finance.html

from cs50 import SQL
from flask import Flask, flash, redirect, render_template, request, url_for
from helpers import error

app = Flask(__name__)

# Configure CS50 Library to use SQLite database
db = SQL("sqlite:///tripplanner.db")


@app.route("/")
def homepage():
    # Import data from SQL Tables to display on homepage
    flightinfo = db.execute("SELECT Number, Location, Date, Time FROM \
        flight WHERE 1 ORDER BY date ASC")
    traininfo = db.execute("SELECT Number, Location, Date, Time FROM \
        train WHERE 1 ORDER BY date ASC")
    hotelinfo = db.execute("SELECT Name, Date, Time, Outdate FROM \
        hotel WHERE 1 ORDER BY date ASC")

    # Select cash from SQL
    cashs = db.execute("SELECT cash FROM cash WHERE 1")
    for cash in cashs:
        cash = cash['cash']

    # Display homepage with all information
    return render_template("homepage.html", flightinfo=flightinfo, traininfo=traininfo,
                           hotelinfo=hotelinfo, cash=cash)


@app.route("/flight", methods=["GET", "POST"])
def flight():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        return render_template("flight.html")
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get flight data from form
        number = (request.form.get("number"))
        location = (request.form.get("location"))
        date = (request.form.get("date"))
        time = (request.form.get("time"))
        # Save flight data to SQL
        db.execute("INSERT INTO flight (number, date, location, time) \
            VALUES (:number, :date, :location, :time);", number=number,
                   location=location, date=date, time=time)

        # Redirect to homepage
        return redirect(url_for("homepage"))


@app.route("/train", methods=["GET", "POST"])
def train():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        return render_template("trains.html")
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get train data from form
        number = (request.form.get("number"))
        location = (request.form.get("location"))
        date = (request.form.get("date"))
        time = (request.form.get("time"))

        # Save train data to SQL
        db.execute("INSERT INTO train (number, date, location, time) \
            VALUES (:number, :date, :location, :time);", number=number,
                   location=location, date=date, time=time)

        # Redirect to homepage
        return redirect(url_for("homepage"))


@app.route("/hotel", methods=["GET", "POST"])
def hotel():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        return render_template("hotels.html")
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get hotel data from form
        name = (request.form.get("name"))
        date = (request.form.get("date"))
        time = (request.form.get("time"))
        outdate = (request.form.get("outdate"))
        # Save hotel data to SQL
        db.execute("INSERT INTO hotel (name, date, time, outdate) \
```

```python
                VALUES (:name, :date, :time, :outdate);", name=name,
                       date=date, time=time, outdate=outdate)

        # Redirect to homepage
        return redirect(url_for("homepage"))


@app.route("/expenses", methods=["GET", "POST"])
def expenses():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        return render_template("expenses.html")
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get information from form
        Amount = int(request.form.get("amount"))
        Expense = request.form.get("expense")
        Date = request.form.get("date")
        # Return error if negative number
        if Amount <= 0:
            return error(Amount)
        else:
            # Select cash from SQL
            cashs = db.execute("SELECT cash FROM cash WHERE 1")
            for cash in cashs:
                cash = cash['cash']
            # Return error if amount is greater than current cash
            if cash < Amount:
                return error(Amount)
            elif cash >= Amount:
                # Add transaction to expense history
                db.execute("INSERT INTO expenses (Amount, Expense, Date) VALUES \
                (:Amount, :Expense, :Date)", Expense=Expense, Amount=Amount, Date=Date)
                # Update cash, subtracting cash from transaction
                db.execute("UPDATE cash SET Cash = Cash - :Amount", Amount=Amount)
        # Redirect to homepage
        return redirect(url_for("homepage"))


@app.route("/allowance", methods=["GET", "POST"])
def allowance():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        return render_template("allowance.html")
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get allowance from form
        allowance = int(request.form.get("allowance"))
        # Return error if negative number
        if allowance <= 0:
            return error(allowance)
        # Add allowance to current cash
        db.execute("UPDATE cash SET Cash = Cash + :allowance", allowance=allowance)
    # Redirect to homepage
    return redirect(url_for("homepage"))


@app.route("/expensehistory", methods=["GET", "POST"])
def expensehistory():
    # If page is reached route through POST (submitting a form)
    if request.method == "POST":
        return render_template("expensehistory.html")
    # Else if page is reached through GET (Clicking a link or redirect)
    else:
        # Import expenses from SQL
        expenseinfo = db.execute("SELECT Amount, Expense, Date FROM \
            expenses ORDER BY date ASC")
        # Select all expenses from SQL
        for expense in expenseinfo:
            amount = expense['Amount']
            expense = expense['Expense']

        # Display history with information
        return render_template("expensehistory.html", expenseinfo=expenseinfo)


@app.route("/delete", methods=["GET", "POST"])
def delete():
    # Reached page through GET (Clicking a link or redirect)
    if request.method == "GET":
        # Import tables from SQL
        train = db.execute("SELECT Number FROM train")
        flights = db.execute("SELECT Number FROM flight")
        hotel = db.execute("SELECT Name FROM hotel")
        expense = db.execute("SELECT expense FROM expenses")
        # Display page with information
        return render_template("delete.html", flights=flights, train=train, hotel=hotel, expense=expense)
    # Else if page is reached route through POST (submitting a form)
    else:
        # Get information from form
```

```python
        train = request.form.get("train")
        flight = request.form.get("flight")
        hotel = request.form.get("hotel")
        expense = request.form.get("expense")

        # Remove information of corresponding table
        if train:
            db.execute("DELETE FROM train WHERE Number=:train", train=train)
        elif flight:
            db.execute("DELETE FROM flight WHERE Number=:flight", flight=flight)
        elif hotel:
            db.execute("DELETE FROM hotel WHERE Name=:hotel", hotel=hotel)
        elif expense:
            amount = (db.execute("SELECT Amount FROM expenses WHERE expense=:expense", expense=expense))
            # Update cash (refund) if expense is deleted
            for amount in amount:
                amount = int(amount['Amount'])
            db.execute("DELETE FROM expenses WHERE expense=:expense", expense=expense)
            db.execute("UPDATE cash SET Cash = Cash + :amount", amount=amount)

        # Redirect to homepage
        return redirect(url_for("homepage"))
```