DSE5002 Module 4 Assignment 4 revised 2/18/2025 Peter Gyorda April 12, 2025

Question 1

1. import the random library.

2. Use `random.seed(10)` to initialize a pseudorandom number generator.

3. Create a list of 50 random integers from 0 to 15. Call this list `int_list`.

4. Print the 10th and 30th elements of the list.

You will need to use list comprehension to do this. The syntax for list comprehension is: <new_list> = `[<expression> for <item> in <iterable>]`. For this question your expression will be a randint generator from the random library and your iterable will be `range()`. Researh the documentation on how to use both functions.

In [6]:
```python
import random

random.seed(10)  # Initialize the pseudorandom number generator

int_list = [random.randint(0, 15) for _ in range(50)]  # Create a list of 50 random integers

print(int_list[9])  # Print the 10th element (index 9)
print(int_list[29]) # Print the 30th element (index 29)
```

```
1
7
```

Question 2

1. import the string library.

2. Create the string `az_upper` using `string.ascii_uppercase`. This is a single string of uppercase letters

3. Create a list of each individual letter from the string. To do this you will need to iterate over the string and append each letter to the an empty list. Call this list `az_list`.

4. Print the list.

You will need to use a for-loop for this. The syntax for this for-loop should be:

`for i in string>:

`

In [8]:
```
import string

az_upper = string.ascii_uppercase
az_list = []

for letter in az_upper:
    az_list.append(letter)

print(az_list)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z']
```

Question 3

1. Create a set from 1 to 5. Call this `set_1`.

2. Create a set from int_list. Call this `set_2`.

3. Create a set by finding the `symmetric_difference()` of `set_1` and `set_2`. Call this `set_3`.

4. What is the length of all three sets?

In [10]:
```
import random

random.seed(10)
int_list = [random.randint(0, 15) for _ in range(50)]
```

```
set_1 = {1, 2, 3, 4, 5}
set_2 = set(int_list)
set_3 = set_1.symmetric_difference(set_2)

print(f"Length of set_1: {len(set_1)}")
print(f"Length of set_2: {len(set_2)}")
print(f"Length of set_3: {len(set_3)}")
```

```
Length of set_1: 5
Length of set_2: 15
Length of set_3: 12
```

Question 4

Complete exercise 9.15.3 from Think Python by Downey

https://allendowney.github.io/ThinkPython/chap09.html

In [3]:
```
def is_palindrome(word):
    """
    Checks if a string is a palindrome.

    Args:
        word (str): The string to check.

    Returns:
        bool: True if the string is a palindrome, False otherwise.
    """
    word = word.lower()  # Convert to lowercase for case-insensitive comparison
    return word == word[::-1]  # Compare the word to its reversed version

# Example usage with a word list (you'll need to define word_list):
word_list = ["rotator", "noon", "level", "python", "madam", "racecar", "step on no pets", "hello", "Aibohphobia"] # e

for word in word_list:
    if len(word) >= 7 and is_palindrome(word):
        print(word)
```

```
rotator
racecar
step on no pets
Aibohphobia
```

In [ ]:
```
Book exercise Exercise 3-1
Retrieve the employee ID, first name, and last name for all bank employees. Sort by last name and then by first name
```

In [9]:
```python
# Libaries

import sqlalchemy

# we will want Pandas for the data frame structure

import pandas as pd
```

In [11]:
```python
# Connect to the database
# Alter this to reflect your username and password,   this is for postgres on the same machine

engine=sqlalchemy.create_engine('postgresql://todd:password@localhost:5432/bank')
```

In [67]:
```python
import psycopg2  # Import the psycopg2 module for PostgreSQL instead of sqlite3

# Create a connection to the PostgreSQL database
try:
    # Use psycopg2.connect for PostgreSQL connections
    connection = psycopg2.connect(
        host="localhost",
        port="5432",
        database="bank",
        user="todd",
        password="password"
    )

    def get_employees(connection):
        try:
            cursor = connection.cursor()

            cursor.execute("SELECT emp_id, fname, lname FROM employee ORDER BY lname, fname;")

            employees = cursor.fetchall()
            # Print the list of employees
            print("List of employees:")
            for employee in employees:
                print(f"ID: {employee[0]}, Name: {employee[1]} {employee[2]}")
            return employees

        except Exception as e:
            print(f"An error occurred: {e}")
            return None

    # Call the function with your connection object
    employees = get_employees(connection)


except Exception as e:
    print(f"Connection error: {e}")
```

```
List of employees:
ID: 2, Name: Susan Barker
ID: 13, Name: John Blake
ID: 6, Name: Helen Fleming
ID: 17, Name: Beth Fowler
ID: 5, Name: John Gooding
ID: 9, Name: Jane Grossman
ID: 4, Name: Susan Hawthorne
ID: 12, Name: Samantha Jameson
ID: 16, Name: Theresa Markham
ID: 14, Name: Cindy Mason
ID: 8, Name: Sarah Parker
ID: 15, Name: Frank Portman
ID: 10, Name: Paula Roberts
ID: 1, Name: Michael Smith
ID: 7, Name: Chris Tucker
ID: 18, Name: Rick Tulman
ID: 3, Name: Robert Tyler
ID: 11, Name: Thomas Ziegler
```

In [69]:

```
def get_account_columns(connection):
    cursor = connection.cursor()
    cursor.execute("""
        SELECT column_name
        FROM information_schema.columns
        WHERE table_name = 'account'
    """)
    columns = [col[0] for col in cursor.fetchall()]
    cursor.close()
    return columns

# Get the list of column names
account_columns = get_account_columns(connection)
print(account_columns)
```

```
['account_id', 'product_cd', 'cust_id', 'open_date', 'close_date', 'last_activity_date', 'status', 'open_branch_id',
'open_emp_id', 'avail_balance', 'pending_balance', 'trial104']
```

In [ ]:

```
QUESTION: Exercise 3-2
Retrieve the account ID, customer ID, and available balance for all accounts whose status equals 'ACTIVE' and whose a
```

In [79]:

```
import psycopg2

try:
    connection = psycopg2.connect(
        host="localhost",
        port="5432",
        database="bank",
        user="todd",
        password="password"
    )

    def get_active_high_balance_accounts(connection):
        try:
            cursor = connection.cursor()

            cursor.execute(
                "SELECT account_id, cust_id, avail_balance "
                "FROM account "
                "WHERE status = 'ACTIVE' AND avail_balance > 2500;"
            )

            accounts = cursor.fetchall()

            if accounts:
                print("Active Accounts with Balance > $2,500:")
                for account in accounts:
                    print(f"Account ID: {account[0]}, Customer ID: {account[1]}, Available Balance: ${account[2]:.2f
                else:
                    print("No active accounts found with balance greater than $2,500.")

                return accounts

        except psycopg2.Error as e:
            print(f"Database error: {e}")
            return None

    accounts = get_active_high_balance_accounts(connection)

except psycopg2.Error as e:
    print(f"Connection error: {e}")

finally:
    if 'connection' in locals() and connection:
        connection.close()
```

```
Active Accounts with Balance > $2,500:
Account ID: 3, Customer ID: 1, Available Balance: $3000.00
Account ID: 12, Customer ID: 4, Available Balance: $5487.09
Account ID: 15, Customer ID: 6, Available Balance: $10000.00
Account ID: 17, Customer ID: 7, Available Balance: $5000.00
Account ID: 18, Customer ID: 8, Available Balance: $3487.19
Account ID: 22, Customer ID: 9, Available Balance: $9345.55
Account ID: 24, Customer ID: 10, Available Balance: $23575.12
Account ID: 27, Customer ID: 11, Available Balance: $9345.55
Account ID: 28, Customer ID: 12, Available Balance: $38552.05
Account ID: 29, Customer ID: 13, Available Balance: $50000.00
```

In [ ]:
```
QUESTION: Exercise 3-3
Write a query against the account table that returns the IDs of the employees who opened the accounts (use the accou
```

In [81]:
```
import psycopg2

try:
    connection = psycopg2.connect(
        host="localhost",
        port="5432",
        database="bank",
        user="todd",
        password="password"
    )

    def get_distinct_account_openers(connection):
        try:
            cursor = connection.cursor()

            cursor.execute("SELECT DISTINCT open_emp_id FROM account;")

            employee_ids = cursor.fetchall()

            if employee_ids:
                print("Employee IDs of Account Openers:")
                for emp_id in employee_ids:
                    print(f"Employee ID: {emp_id[0]}")
                return employee_ids
            else:
                print("No accounts found.")
                return None

        except psycopg2.Error as e:
            print(f"Database error: {e}")
            return None

    employee_ids = get_distinct_account_openers(connection)

except psycopg2.Error as e:
    print(f"Connection error: {e}")

finally:
    if 'connection' in locals() and connection:
        connection.close()
```

```
Employee IDs of Account Openers:
Employee ID: 13
Employee ID: 10
Employee ID: 1
Employee ID: 16
```

In [ ]:

```
QUESTION Exercise 4-3
Construct a query that retrieves all accounts opened in 2002.
```

In [83]:

```python
import psycopg2

try:
    connection = psycopg2.connect(
        host="localhost",
        port="5432",
        database="bank",
        user="todd",
        password="password"
    )

    def get_accounts_opened_in_2002(connection):
        try:
            cursor = connection.cursor()

            cursor.execute(
                "SELECT account_id, cust_id, open_date "
                "FROM account "
                "WHERE EXTRACT(YEAR FROM open_date) = 2002;"
            )

            accounts = cursor.fetchall()

            if accounts:
                print("Accounts Opened in 2002:")
                for account in accounts:
                    print(f"Account ID: {account[0]}, Customer ID: {account[1]}, Open Date: {account[2]}")
            else:
                print("No accounts opened in 2002.")

            return accounts

        except psycopg2.Error as e:
            print(f"Database error: {e}")
            return None

    accounts = get_accounts_opened_in_2002(connection)

except psycopg2.Error as e:
    print(f"Connection error: {e}")
```

```
finally:
    if 'connection' in locals() and connection:
        connection.close()
```

```
Accounts Opened in 2002:
Account ID: 7, Customer ID: 3, Open Date: 2002-11-23
Account ID: 8, Customer ID: 3, Open Date: 2002-12-15
Account ID: 14, Customer ID: 6, Open Date: 2002-08-24
Account ID: 24, Customer ID: 10, Open Date: 2002-09-30
Account ID: 25, Customer ID: 10, Open Date: 2002-10-01
```

```
In [ ]:
```