


TRƯỜNG: ĐH Công thương TP HCM KHOA: CÔNG NGHỆ THÔNG TIN BỘ MÔN: KHDL&TTNT MH: LẬP TRÌNH PYTHON	BUỔI 6. Random và Sys Module	
---	-------------------------------------	---

A. MỤC TIÊU

- Làm việc với Random Module
- Làm việc với Sys Module

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

1. Random Module

a. Giới thiệu về Random Module

Module Random trong Python là một công cụ mạnh mẽ cho việc sinh số ngẫu nhiên và thực hiện các hoạt động liên quan đến ngẫu nhiên. Random Module cung cấp một loạt các hàm để tạo ra các giá trị ngẫu nhiên, từ các số nguyên đến các dãy ký tự.

A. Mục đích sử dụng:

- Tạo số ngẫu nhiên cho các thử nghiệm và mô phỏng.
- Xáo trộn dữ liệu trong các ứng dụng liên quan đến xử lý dữ liệu.
- Lựa chọn ngẫu nhiên các phần tử từ một tập hợp.

B. Các chức năng chính:

- `random()`: Hàm này sinh ra một số thực ngẫu nhiên trong khoảng từ 0 đến 1.
- `seed()`: Hàm này thiết lập giá trị khởi tạo cho bộ sinh số ngẫu nhiên.
- `randint(a, b)`: Hàm này tạo ra một số nguyên ngẫu nhiên nằm trong phạm vi từ a đến b.
- `choice(seq)`: Hàm này chọn ngẫu nhiên một phần tử từ một chuỗi hoặc danh sách.

- `shuffle(seq)`: Hàm này xáo trộn ngẫu nhiên các phần tử trong một chuỗi hoặc danh sách.

C. Ứng dụng thực tế

- Trò chơi may rủi: Sử dụng để tạo ra số ngẫu nhiên trong trò chơi may rủi như lật bài, lắc xúc xắc, v.v.
- Xử lý dữ liệu: Dùng để xáo trộn dữ liệu đầu vào để tránh việc dữ liệu bị lặp lại hoặc theo thứ tự nhất định.
- Mô phỏng: Sử dụng để tạo ra các sự kiện ngẫu nhiên trong mô phỏng các hệ thống thực tế.

Module Random là một phần quan trọng của Python với nhiều ứng dụng rộng rãi từ trò chơi đến khoa học dữ liệu. Sử dụng hiệu quả, nó giúp tăng tính ngẫu nhiên và đa dạng trong các ứng dụng của bạn.

b. Hàm `random()`

Hàm `random()` là một trong những hàm cơ bản của module random trong Python. Hàm này được sử dụng để sinh ra một số ngẫu nhiên trong khoảng từ 0 đến 1, bao gồm cả 0 và 1. Đây là một hàm quan trọng khi bạn cần tạo ra các giá trị ngẫu nhiên cho các mục đích như mô phỏng, trò chơi, hoặc xử lý dữ liệu.

Dưới đây là cú pháp của hàm `random()`:

```
import random
random_value = random.random()
```

Trong đó:

- `import random`: Dòng này được sử dụng để nhập module random, cho phép bạn truy cập vào tất cả các hàm và thuộc tính được cung cấp bởi module này.
- `random.random()`: Đây là cách gọi hàm `random()` để sinh ra một số ngẫu nhiên. Giá trị trả về sẽ là một số thực nằm trong khoảng từ 0 đến 1.

Ví dụ minh họa:

```
import random

random_value = random.random()
```

```
print("Giá trị ngẫu nhiên:", random_value)
```

c. Hàm seed()

Hàm seed() trong module random của Python được sử dụng để thiết lập giá trị khởi tạo cho bộ sinh số ngẫu nhiên. Điều này rất hữu ích khi bạn muốn tái tạo chuỗi ngẫu nhiên cụ thể trong các chương trình. Khi bạn đặt cùng một giá trị seed cho bộ sinh số ngẫu nhiên, bạn sẽ luôn nhận được cùng một chuỗi giá trị ngẫu nhiên sau mỗi lần chạy chương trình, giúp cho việc kiểm thử và gỡ lỗi trở nên dễ dàng hơn.

```
import random

random.seed(a=None, version=2)
```

Trong đó:

- a: Đây là giá trị khởi tạo cho bộ sinh số ngẫu nhiên. Nếu không được chỉ định, hàm seed sẽ sử dụng một giá trị mặc định từ hệ thống máy tính. Nếu bạn muốn nhận được kết quả giống nhau sau mỗi lần chạy, bạn có thể đặt một giá trị cụ thể cho a.
- version: Đây là phiên bản của thuật toán seed. Theo mặc định, nó là 2.

Ví dụ minh họa:

```
import random

# Thiết lập seed
random.seed(10)

# Sinh ra số ngẫu nhiên
random_value = random.random()
print("Giá trị ngẫu nhiên:", random_value)
```

Lưu ý rằng khi bạn thiết lập cùng một giá trị seed, chuỗi các giá trị ngẫu nhiên sinh ra sau đó sẽ luôn giống nhau. Điều này hữu ích trong việc tái tạo các kịch bản ngẫu nhiên trong các ứng dụng phức tạp.

d. Hàm randint()

Hàm randint() trong module random của Python được sử dụng để sinh ra một số nguyên ngẫu nhiên nằm trong một phạm vi đã cho. Phạm vi này bao gồm cả hai giá trị đầu và cuối. Điều

này rất hữu ích khi bạn cần tạo ra các giá trị ngẫu nhiên để sử dụng trong các ứng dụng như trò chơi, mô phỏng, hoặc kiểm thử.

```
import random

random_integer = random.randint(a, b)
```

Trong đó:

- a: Là giá trị dưới cùng trong phạm vi bạn muốn sinh ra.
- b: Là giá trị trên cùng trong phạm vi bạn muốn sinh ra.

Hàm randint() sẽ trả về một số nguyên ngẫu nhiên nằm trong phạm vi từ a đến b, bao gồm cả a và b.

Ví dụ minh họa:

```
import random

# Sinh số nguyên ngẫu nhiên trong phạm vi từ 1 đến 100
random_integer = random.randint(1, 100)
print("Số nguyên ngẫu nhiên:", random_integer)
```

Hàm randint() rất tiện lợi khi bạn muốn sinh ra các số nguyên ngẫu nhiên trong một phạm vi cụ thể.

e. Hàm choice()

Hàm choice() trong module random của Python được sử dụng để chọn một phần tử ngẫu nhiên từ một chuỗi, danh sách hoặc bất kỳ đối tượng tuần tự nào khác. Điều này rất hữu ích khi bạn cần lựa chọn ngẫu nhiên một phần tử từ một tập hợp các phần tử đã cho.

```
import random

random_choice = random.choice(seq)
```

Trong đó:

- seq: Đây là chuỗi, danh sách hoặc bất kỳ đối tượng tuần tự nào khác từ đó bạn muốn chọn một phần tử ngẫu nhiên.

Hàm choice() sẽ trả về một phần tử ngẫu nhiên từ seq.

Ví dụ minh họa:

```
import random

# Danh sách các môn học
subjects = ['Math', 'Physics', 'Biology', 'Chemistry',
            'History']

# Lựa chọn một môn học ngẫu nhiên từ danh sách
random_subject = random.choice(subjects)
print("Môn học được chọn:", random_subject)
```

Hàm choice() giúp bạn dễ dàng lựa chọn ngẫu nhiên một phần tử từ một danh sách các phần tử đã cho trong Python.

f. Hàm shuffle()

Hàm shuffle() trong module random của Python được sử dụng để xáo trộn ngẫu nhiên các phần tử trong một chuỗi, danh sách hoặc bất kỳ đối tượng tuần tự nào khác. Điều này rất hữu ích khi bạn muốn xáo trộn thứ tự các phần tử trong một tập hợp để tạo ra sự ngẫu nhiên hoặc đảm bảo rằng dữ liệu không bị ảnh hưởng bởi thứ tự ban đầu.

```
import random

random.shuffle(seq)
```

Trong đó:

- seq: Đây là chuỗi, danh sách hoặc bất kỳ đối tượng tuần tự nào khác mà bạn muốn xáo trộn các phần tử.

Hàm shuffle() sẽ xáo trộn các phần tử trong seq theo thứ tự ngẫu nhiên.

Ví dụ minh họa:

```
import random

# Danh sách các số từ 1 đến 10
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Xáo trộn thứ tự các số
random.shuffle(numbers)
```

```
print("Danh sách số sau khi xáo trộn:", numbers)
```

g. Ứng dụng thực tế của Random Module

Module random trong Python có nhiều ứng dụng thực tế quan trọng, bao gồm nhưng không giới hạn:

- Trò chơi may rủi và giải trí: Trong các trò chơi như lắc xúc xắc, bài poker, hoặc trò chơi lật bài, việc sinh số ngẫu nhiên là rất cần thiết để tạo ra sự hấp dẫn và tính ngẫu nhiên cho trò chơi.
- Mô phỏng và phân tích thống kê: Trong lĩnh vực khoa học và kỹ thuật, module random có thể được sử dụng để mô phỏng các biến ngẫu nhiên trong các mô hình toán học hoặc các kịch bản thử nghiệm. Điều này giúp các nhà khoa học và kỹ sư hiểu và dự đoán các hiện tượng thực tế dựa trên dữ liệu ngẫu nhiên.
- Tạo dữ liệu kiểm thử: Trong quá trình phát triển phần mềm, việc tạo dữ liệu kiểm thử ngẫu nhiên là một phần quan trọng của việc đảm bảo chất lượng. Module random có thể được sử dụng để tạo ra các tập hợp dữ liệu ngẫu nhiên để kiểm thử các hàm và phương thức trong phần mềm.
- Xử lý dữ liệu: Trong các ứng dụng xử lý dữ liệu, việc xáo trộn dữ liệu là quan trọng để đảm bảo tính ngẫu nhiên và bảo mật của dữ liệu. Module random cung cấp các hàm để xáo trộn các phần tử trong một danh sách hoặc chuỗi một cách ngẫu nhiên.
- Tạo số liệu thống kê mô phỏng: Trong lĩnh vực thống kê, module random có thể được sử dụng để tạo ra các tập hợp dữ liệu mô phỏng để nghiên cứu các phương pháp thống kê và kiểm định giả thuyết.
- Bảo mật và mã hóa: Trong một số trường hợp, số ngẫu nhiên được sử dụng để tạo ra các khóa mã hóa hoặc token bảo mật trong các ứng dụng như hệ thống đăng nhập, giao dịch tài chính trực tuyến, và bảo mật mạng.

Tổng quát, module random trong Python là một công cụ mạnh mẽ cho việc tạo ra sự ngẫu nhiên và đa dạng trong các ứng dụng từ giải trí đến khoa học và kỹ thuật.

2. Sys Module

a. Giới thiệu về Sys Module

Module `sys` trong Python cung cấp các chức năng để tương tác với hệ thống hoạt động, bao gồm truy cập vào các tham số dòng lệnh, tương tác với môi trường và máy tính, và điều khiển các tiến trình Python. Dưới đây là một số tính năng chính của module `sys`:

- Truy cập tham số dòng lệnh: Module `sys` cho phép bạn truy cập các tham số được truyền vào từ dòng lệnh khi bạn chạy một chương trình Python. Bạn có thể sử dụng các tham số này để điều chỉnh hành vi của chương trình.
- Thông tin về phiên bản Python: Module `sys` cung cấp thông tin về phiên bản của Python đang chạy, bao gồm cả phiên bản chính và các thông tin khác như số phiên bản, biên dịch, và hệ điều hành.
- Truy cập vào các biến môi trường: Module `sys` cho phép bạn truy cập vào các biến môi trường hiện đang được sử dụng bởi chương trình, như biến `PATH`, biến `SHELL`, và các biến khác mà bạn có thể cần trong quá trình thực thi chương trình.
- Thoát khỏi chương trình: Module `sys` cung cấp phương thức `exit()` để thoát khỏi chương trình Python, có thể với một mã lỗi cụ thể được chỉ định. Điều này rất hữu ích khi bạn muốn thoát khỏi chương trình trong trường hợp lỗi hoặc kết thúc chương trình một cách kiểm soát.
- Các hằng số và biến được sử dụng bởi Python: Module `sys` cung cấp các hằng số và biến như `sys.argv`, `sys.platform`, `sys.version`, `sys.path` để cung cấp thông tin về môi trường thực thi của Python.
- Tương tác với tiến trình Python: Module `sys` cung cấp một số phương thức để tương tác với tiến trình Python đang chạy, bao gồm `sys.getsizeof()` để tính kích thước của các đối tượng Python và `sys.stdin`, `sys.stdout`, `sys.stderr` để thực hiện nhập xuất chuẩn.

Module `sys` là một phần quan trọng của Python và cung cấp các công cụ mạnh mẽ để tương tác với hệ thống và kiểm soát hành vi của chương trình.

b. Hàm `argv()`

Trong module `sys`, hàm `argv()` được sử dụng để truy cập các tham số được truyền vào từ dòng lệnh khi bạn chạy một chương trình Python. Thông thường, các tham số này được sử dụng để điều chỉnh hành vi của chương trình hoặc cung cấp đầu vào từ người dùng.

```
import sys
```

```
arguments = sys.argv
```

Khi chương trình Python được chạy, sys.argv sẽ chứa một danh sách các chuỗi, trong đó:

- Phần tử đầu tiên là tên của chương trình Python được gọi.
- Các phần tử tiếp theo là các tham số được truyền từ dòng lệnh.

Ví dụ minh họa:

Nếu bạn chạy chương trình Python sau từ dòng lệnh như sau:

```
python script.py arg1 arg2 arg3
```

Trong file script.py, bạn có thể truy cập các tham số này thông qua sys.argv như sau:

```
import sys

# Lấy tên của chương trình Python
script_name = sys.argv[0]

# Lấy các tham số từ dòng lệnh
args = sys.argv[1:]

print("Tên của chương trình:", script_name)
print("Các tham số:", args)
```

Kết quả sẽ là:

```
Tên của chương trình: script.py
Các tham số: ['arg1', 'arg2', 'arg3']
```

Với hàm argv(), bạn có thể linh hoạt xử lý các tham số dòng lệnh khi chạy chương trình Python của mình.

c. Hàm exit()

Trong module sys, hàm exit() được sử dụng để kết thúc chương trình Python một cách ngay lập tức. Hàm này có thể nhận một số nguyên làm đối số, đại diện cho mã lỗi thoát. Thông thường, khi không có lỗi xảy ra, bạn có thể sử dụng mã lỗi là 0 để biểu thị việc thoát chương trình thành công, trong khi các mã lỗi khác thường được sử dụng để biểu thị việc xảy ra lỗi cụ thể.

```
import sys
```



```
sys.exit(exit_code)
```

Trong đó:

- `exit_code` là một số nguyên, đại diện cho mã lỗi khi thoát chương trình. Mặc định là 0 nếu không được chỉ định.

Ví dụ minh họa:

```
import sys

def main():
    # Kiểm tra điều kiện
    if something_wrong:
        print("Có lỗi xảy ra.")
        sys.exit(1) # Thoát chương trình với mã lỗi 1

    print("Chương trình hoạt động bình thường.")
    sys.exit(0) # Thoát chương trình với mã lỗi 0

if __name__ == "__main__":
    main()
```

Trong ví dụ này, nếu có lỗi xảy ra trong hàm `main()`, chương trình sẽ kết thúc với mã lỗi 1. Nếu không có lỗi, chương trình sẽ kết thúc bình thường với mã lỗi 0.

Hàm `exit()` thường được sử dụng để kiểm soát việc thoát chương trình trong các điều kiện đặc biệt hoặc khi gặp phải lỗi không thể xử lý được.

d. Hàm `version_info()`

Hàm `version_info()` trong module `sys` của Python được sử dụng để truy cập thông tin về phiên bản của trình thông dịch Python đang được sử dụng. Thông tin này bao gồm các thành phần như số phiên bản chính, số phiên bản nhỏ, số phiên bản con, và các thông tin khác liên quan đến phiên bản hiện đang chạy.

Khi bạn gọi `version_info()`, nó sẽ trả về một tuple chứa thông tin về phiên bản của Python đang chạy, trong đó mỗi thành phần tương ứng với một thông tin nhất định về phiên bản:

- `major`: Số phiên bản chính.
- `minor`: Số phiên bản nhỏ.

- micro: Số phiên bản con.
- releaselevel: Mức phát hành (ví dụ: "alpha", "beta", "final").
- serial: Số serial.

Ví dụ minh họa:

```
import sys

version_info = sys.version_info

print("Số phiên bản chính:", version_info.major)
print("Số phiên bản nhỏ:", version_info.minor)
print("Số phiên bản con:", version_info.micro)
print("Mức phát hành:", version_info.releaselevel)
print("Số serial:", version_info.serial)
```

Hàm version_info() rất hữu ích khi bạn cần kiểm tra các tính năng hoặc hành vi của chương trình của mình dựa trên phiên bản của Python.

e. Hàm platform()

Trong module sys, hàm platform() được sử dụng để truy cập thông tin về nền tảng hệ thống mà Python đang chạy trên đó. Thông tin này bao gồm tên hệ điều hành, phiên bản hệ điều hành, kiến trúc hệ thống và các chi tiết khác liên quan đến môi trường thực thi của Python.

```
import sys

platform_info = sys.platform()
```

Khi bạn gọi platform(), nó sẽ trả về một chuỗi đại diện cho nền tảng hệ thống, chẳng hạn như "win32" cho Windows, "linux" cho Linux, "darwin" cho macOS, và các giá trị khác tùy thuộc vào hệ thống mà bạn đang sử dụng.

f. Ứng dụng thực tế của Sys Module

Module sys trong Python có nhiều ứng dụng thực tế quan trọng, bao gồm nhưng không giới hạn:

- Truy cập thông tin hệ thống: Module sys cung cấp các phương thức để truy cập thông tin về hệ thống mà Python đang chạy trên đó, bao gồm tên hệ điều hành, phiên bản hệ

điều hành, kiến trúc hệ thống và nền tảng hệ thống. Điều này rất hữu ích khi bạn muốn thực hiện các hành vi khác nhau dựa trên môi trường thực thi của Python.

- Truy cập các tham số dòng lệnh: Module sys cung cấp phương thức argv để truy cập các tham số dòng lệnh mà người dùng đã truyền vào khi chạy chương trình Python. Điều này cho phép bạn điều chỉnh hành vi của chương trình dựa trên các tham số được chỉ định từ dòng lệnh.
- Điều khiển quy trình chương trình: Module sys cung cấp các phương thức như exit() để thoát khỏi chương trình Python một cách ngay lập tức với mã lỗi cụ thể. Điều này có thể được sử dụng để xử lý các điều kiện đặc biệt hoặc lỗi không thể xử lý được.
- Kiểm soát đầu ra chuẩn: Module sys cho phép bạn truy cập vào các luồng đầu ra chuẩn của chương trình, bao gồm stdin, stdout, và stderr, để điều khiển và xử lý dữ liệu nhập và xuất của chương trình.
- Tùy chỉnh môi trường: Module sys cung cấp các phương thức để tùy chỉnh môi trường thực thi của Python, bao gồm path, path_hooks, và các biến môi trường khác.
- Kiểm tra thông tin phiên bản và module: Module sys cho phép bạn kiểm tra thông tin về phiên bản của Python và các module được tải vào trong quá trình thực thi của chương trình.

Tổng quát, module sys cung cấp các công cụ quan trọng để tương tác với hệ thống và kiểm soát hành vi của chương trình Python trong môi trường thực thi.

II. Bài tập hướng dẫn mẫu

Bài 1. Game đoán số

Viết một chương trình Python sử dụng module random và sys để tạo một trò chơi đoán số. Trò chơi sẽ cho phép người chơi đoán một số nguyên từ 1 đến 100. Chương trình sẽ sinh ra một số ngẫu nhiên từ 1 đến 100 và yêu cầu người chơi đoán số. Sau mỗi lần đoán, chương trình sẽ cung cấp gợi ý nếu số đoán của người chơi là quá cao hoặc quá thấp. Trò chơi sẽ tiếp tục cho đến khi người chơi đoán đúng số hoặc nhấn phím thoát.

Yêu cầu:

- Chương trình sẽ sinh ra một số ngẫu nhiên từ 1 đến 100.
- Chương trình sẽ hiển thị thông báo yêu cầu người chơi đoán số.

- Người chơi nhập một số từ bàn phím.
- Chương trình sẽ kiểm tra xem số đoán của người chơi có đúng không và cung cấp gợi ý nếu số đó là quá cao hoặc quá thấp.
- Trò chơi sẽ tiếp tục cho đến khi người chơi đoán đúng số hoặc nhấn phím thoát.

```
import random
import sys

def guess_number():
    # Sinh số ngẫu nhiên từ 1 đến 100
    secret_number = random.randint(1, 100)

    # Lặp cho đến khi người chơi đoán đúng hoặc thoát
    while True:
        try:
            guess = int(input("Hãy đoán số từ 1 đến 100: "))
        except ValueError:
            print("Bạn cần nhập một số nguyên.")
            continue

        if guess < secret_number:
            print("Số của bạn quá thấp.")
        elif guess > secret_number:
            print("Số của bạn quá cao.")
        else:
            print("Chúc mừng! Bạn đã đoán đúng số.")
            break

        try:
            choice = input("Bạn có muốn tiếp tục chơi không? (Nhập exit để thoát): ")
            if choice.strip().lower() == 'exit':
                sys.exit(0)
        except KeyboardInterrupt:
            sys.exit(0)

if __name__ == "__main__":
    guess_number()
```

Bài 2. Ứng dụng máy tính phát sinh bài toán toán học

Viết một chương trình Python sử dụng module random và sys để tạo ra các bài toán toán học ngẫu nhiên cho người dùng giải. Chương trình sẽ phát sinh một loạt các phép tính cộng, trừ, nhân hoặc chia với các số nguyên ngẫu nhiên từ 1 đến 10. Người dùng sẽ được yêu cầu giải các bài toán này. Sau mỗi bài toán, chương trình sẽ cung cấp kết quả và tiếp tục hoặc kết thúc nếu người dùng muốn dừng lại.

Yêu cầu:

- Chương trình sẽ phát sinh một phép tính ngẫu nhiên từ các phép tính cộng, trừ, nhân hoặc chia với các số nguyên ngẫu nhiên từ 1 đến 10.
- Người dùng sẽ được yêu cầu nhập kết quả của phép tính đó.
- Chương trình sẽ kiểm tra xem kết quả người dùng nhập có chính xác không và cung cấp kết quả đúng của phép tính.
- Chương trình sẽ tiếp tục phát sinh và yêu cầu giải các bài toán cho đến khi người dùng muốn dừng lại hoặc nhấn phím thoát.

```
import random
import sys

def generate_random_math_problem():
    operators = ['+', '-', '*', '/']
    operator = random.choice(operators)
    num1 = random.randint(1, 10)
    num2 = random.randint(1, 10)

    if operator == '/':
        # Đảm bảo phép chia có kết quả là số nguyên
        while num1 % num2 != 0:
            num1 = random.randint(1, 10)
            num2 = random.randint(1, 10)

    return num1, operator, num2

def ask_user_solution(problem):
    num1, operator, num2 = problem
```

```

    solution = input(f"Giải bài toán: {num1} {operator} {num2} = ")
    return solution

def check_solution(problem, solution):
    num1, operator, num2 = problem
    correct_solution = eval(f"{num1} {operator} {num2}")
    return int(solution) == correct_solution

def main():
    while True:
        problem = generate_random_math_problem()
        solution = ask_user_solution(problem)

        if solution.lower() == 'exit':
            sys.exit(0)

        if check_solution(problem, solution):
            print("Chính xác!")
        else:
            print("Sai! Đáp án đúng là:", eval(f"{problem[0]} {problem[1]} {problem[2]}"))

        try:
            choice = input("Bạn muốn tiếp tục chơi không? (Nhấn Enter để tiếp tục, hoặc nhập 'exit' để kết thúc): ")
            if choice.strip().lower() == 'exit':
                sys.exit(0)
        except KeyboardInterrupt:
            sys.exit(0)

if __name__ == "__main__":
    main()

```

III. Bài tập ở lớp

Bài 1. Tìm Số Ngẫu Nhiên: Viết một chương trình yêu cầu người dùng nhập một số nguyên dương n , sau đó phát sinh n số ngẫu nhiên và in chúng ra màn hình.

Bài 2. Sắp Xếp Dãy Số Ngẫu Nhiên: Viết một chương trình phát sinh một dãy số ngẫu nhiên, sau đó sắp xếp dãy số đó theo thứ tự tăng dần và in ra màn hình.

Bài 3. Tính Toán Trung Bình: Viết một chương trình tính toán trung bình của một dãy số nguyên ngẫu nhiên từ 1 đến 100.

Bài 4. Tìm Số Lớn Nhất và Nhỏ Nhất: Viết một chương trình tạo một dãy số ngẫu nhiên, sau đó tìm và in ra số lớn nhất và nhỏ nhất trong dãy số đó.

Bài 5. Xoá Phần Tử Trùng Lặp: Viết một chương trình tạo một danh sách ngẫu nhiên gồm các phần tử có thể trùng lặp, sau đó xoá các phần tử trùng lặp và in ra danh sách sau khi đã xoá.

Bài 6. Chọn Số Ngẫu Nhiên từ Danh Sách: Viết một chương trình cho phép người dùng nhập một danh sách các số nguyên, sau đó chọn ngẫu nhiên một số từ danh sách đó và in ra màn hình.

Bài 7. Mô Phỏng Ném Xúc Xắc: Viết một chương trình mô phỏng việc ném xúc xắc. Chương trình sẽ phát sinh ngẫu nhiên giá trị từ 1 đến 6 cho hai xúc xắc và tính tổng của chúng.

Bài 8. Mô Phỏng Lựa Chọn Ngẫu Nhiên: Viết một chương trình mô phỏng việc lựa chọn ngẫu nhiên từ một danh sách các mục. Chương trình sẽ cho phép người dùng chọn một mục ngẫu nhiên từ danh sách đó.

Bài 9. Đổi Chỗ Ngẫu Nhiên trong Danh Sách: Viết một chương trình đổi chỗ ngẫu nhiên các phần tử trong một danh sách và in ra danh sách sau khi đã đổi chỗ.

Bài 10. Tạo Mật Khẩu Ngẫu Nhiên: Viết một chương trình tạo một mật khẩu ngẫu nhiên bằng cách sử dụng các ký tự chữ cái, số và ký tự đặc biệt từ bộ ký tự ASCII. Chương trình sẽ yêu cầu người dùng nhập độ dài của mật khẩu và tạo một mật khẩu ngẫu nhiên có độ dài đó.

Bài 11. Quản Lý Danh Sách Học Sinh:

- Xây dựng một ứng dụng quản lý danh sách học sinh với các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa).
- Dữ liệu ban đầu sẽ được tạo ngẫu nhiên với các thông tin như tên, tuổi, giới tính, lớp học, điểm số, v.v.
- Người dùng có thể thêm mới, xem danh sách, cập nhật thông tin và xóa học sinh khỏi danh sách.
- Dữ liệu sẽ được lưu trữ trong một tệp JSON.

Bài 12. Quản Lý Danh Sách Sản Phẩm:

- Xây dựng một ứng dụng quản lý danh sách sản phẩm với các thao tác CRUD.
- Sản phẩm sẽ được mô tả bởi các thuộc tính như tên, mô tả, giá, số lượng, v.v.
- Dữ liệu ban đầu sẽ được tạo ngẫu nhiên với một danh sách các sản phẩm.
- Người dùng có thể thêm mới, xem danh sách, cập nhật thông tin và xóa sản phẩm khỏi danh sách.
- Dữ liệu sẽ được lưu trữ trong một tệp JSON.

Bài 13. Quản Lý Danh Sách Công Việc:

- Xây dựng một ứng dụng quản lý danh sách công việc với các thao tác CRUD.
- Mỗi công việc sẽ có các thuộc tính như tiêu đề, mô tả, ngày bắt đầu, ngày kết thúc, trạng thái (hoàn thành/chưa hoàn thành), v.v.
- Dữ liệu ban đầu sẽ được tạo ngẫu nhiên với một danh sách các công việc.
- Người dùng có thể thêm mới, xem danh sách, cập nhật thông tin và xóa công việc khỏi danh sách.
- Dữ liệu sẽ được lưu trữ trong một tệp JSON.