Computer Science Basic

Lab 8. Hàm và module

Bài 1. Số Nguyên

Yêu cầu người dùng nhập vào một số thực. Hãy kiểm tra số thực đó có phải là số nguyên hay không.

Yêu cầu: Viết hàm **`is_int()`** nhận vào một số thực. Hàm trả về **`True`** nếu số nhận vào là số nguyên, ngược lại trả về **`False`**. Khi đó ta có thể sử dụng hàm như sau:

```
if is_int(num):
  print(f'{int(num)} is an integer')
else:
  print(f'{num} is not an integer')
```

Kết quả mong đợi của chương trình:

Ví dụ 1	Ví dụ 2
Input number: **3** 3 is an integer	Input number: **3.14** 3.14 is not an integer

Với phần in đậm là nội dung được nhập từ người dùng.

Bài 2. Số Nguyên Tố

Số nguyên tố là số nguyên dương *lớn hơn 1, chỉ chia hết cho 1 và chính nó*.

Các số nguyên tố đầu tiên là 2, 3, 5, 7, 11, 13, 17, 19, 23, ...
Số 4 không phải số nguyên tố vì chia hết cho 2. Số 9 không phải số nguyên tố vì chia hết cho 3,...

Yêu cầu người dùng nhập vào một số nguyên. Kiểm tra số đã nhập có phải là số nguyên tố hay không.

Yêu cầu: Viết hàm **`is_prime()`** nhận vào một số nguyên, trả về **`True`** nếu số nhận vào là số nguyên tố, ngược lại trả về **`False`**.

Kết quả mong đợi của chương trình:

Ví dụ 1	Ví dụ 2
Input number: **17** 17 is a prime	Input character: **9** 9 is not a prime

Gợi ý: Để kiểm tra số nguyên tố, ta kiểm tra số đó có chia hết cho *bất kì* số nguyên > 1 nào bé hơn nó hay không. Nếu có thì số đó không phải số nguyên tố.

Bài 3. Liệt Kê Số Nguyên Tố

In ra màn hình **n** số nguyên tố đầu tiên, với **n** là một số nguyên dương được nhập từ người dùng.

Kết quả mong đợi của chương trình:

Ví dụ 1	Ví dụ 2
`Input number: 1` `First 1 prime(s): 2`	`Input number: 9` `First 9 prime(s): 2 3 5 7 11 13 17 19 23`

Gợi ý: Sử dụng hàm **`is_prime()`** để lần lượt kiểm tra tính nguyên tố của các số nguyên dương. Chương trình kết thúc sau khi tìm thấy **n** số nguyên tố.

Bài 4. Biểu Thức Đặc Biệt

Viết chương trình tính giá trị của biểu thức sau, với **n** là một số nguyên dương được nhập từ người dùng.

$$\frac{1!}{1} + \frac{2!}{2} + \ldots + \frac{n!}{n}$$

Kết quả mong đợi của chương trình:

Ví dụ 1	Ví dụ 2	Ví dụ 3
`Input number: 1` `Result: 1`	`Input number: 2` `Result: 2`	`Input number: 5` `Result: 34`

Gợi ý: Viết hàm tính giai thừa để đưa vào vòng lặp tính tổng lớn.

Bài 5. Datetime

Trong Python, module **`datetime`** hỗ trợ các xử lý trên ngày tháng.

Ví dụ, để lấy thời gian hiện tại trong hệ thống, ta viết:

```
from datetime import datetime

now = datetime.now()
print(now)
```

Biến **`now`** là một đối tượng **`datetime`**, lưu thông tin về một thời điểm cụ thể. Để thay đổi định dạng in ra mặc định, ta viết định dạng mong muốn vào phương thức **`strftime()`** có sẵn trong module **`datetime`**.

```
print(now.strftime('%Y/%m/%d'))
print(now.strftime('%H:%M:%S %Y-%m-%d'))
```

Hãy chạy các đoạn code trên để xem kết quả, sau đó chỉnh sửa để chương trình in ra nội dung sau, tương ứng với ngày hiện tại trong hệ thống:

```
Today is 21/07/2021
Time right now: 13:36:10
```