BÀI 2 MÔI TRƯỜNG PYTHON SHELL VÀ MỘT SỐ KIẾN THỨC LẬP TRÌNH CƠ BẢN

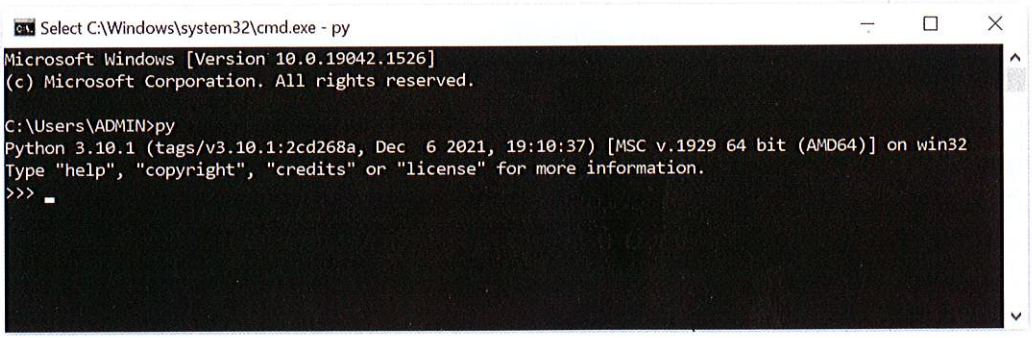
2.1. Giới thiệu cách ngôn ngữ lập trình hoạt động

Máy tính chỉ có thể hiểu, thực thi lệnh và chạy chương trình được viết dưới dạng mã máy (mã nhị phân). Việc viết trực tiếp mã máy sẽ gặp rất nhiều khó khăn, mất nhiều công sức và rất dễ xảy ra lỗi, vì vậy con người tạo ra ngôn ngữ lập trình với mong muốn giúp cho chương trình máy tính trở nên dễ hiểu, qua đó dễ sửa chữa, bảo trì và phát triển hơn. Tất cả các ngôn ngữ lập trình đều đi kèm với một chương trình đặc biệt (được gọi là trình dịch) giúp "dịch" mã nguồn của ngôn ngữ đó thành mã máy.

Các ngôn ngữ lập trình khác nhau có cách dịch mã nguồn khác nhau, xoay quanh 2 phương pháp cơ bản: Biên dịch và thông dịch. Trình dịch sử dụng phương pháp biên dịch được gọi là trình biên dịch (compiler), sẽ dịch toàn bộ tệp tin mã nguồn thành tệp tin mã máy, sau đó máy tính sẽ thực thi lệnh dựa trên tệp tin mã máy đó. Trình dịch sử dụng phương pháp thông dịch được gọi là trình thông dịch (interpreter), sẽ dịch từng dòng lệnh của mã nguồn và thực thi dòng lệnh đó ngay sau khi dịch xong sau đó mới chuyển sang lệnh tiếp theo.

2.2. Giới thiệu về môi trường Python Shell

Python Shell là một công cụ có đồ họa dòng lệnh (còn được gọi là giao diện console) khởi động trình thông dịch của Python. Để truy cập vào Python Shell, các bạn thực hiện các bước như Bài 1 hướng dẫn.



Dấu >>> là dấu nhắc lệnh, tại vị trí này các bạn có thể gõ lệnh và chạy trực tiếp từng lệnh bằng cách nhấn phím Enter, máy tính sẽ thực hiện công việc theo lệnh được nhập và hiển thị kết quả. Ví dụ dưới đây là lệnh thực hiện phép toán 2 + 3, dòng hiển thị số 5 là kết quả máy tính thực hiện và không có dấu nhắc lệnh.

```
>>> 2 + 3
5
>>>
```

Python Shell có mục đích chính dùng để kiểm tra hoạt động của những câu lệnh đơn lẻ hay chương trình đơn giản. Với những chương trình phức tạp, ta cần sử dụng môi trường phát triển tích hợp (IDE). Sau đây, chúng ta sẽ sử dụng Python Shell để tìm hiểu các kiến thức cơ bản về Python.

2.3. Một số kiến thức cơ bản về lập trình và ngôn ngữ lập trình Python

2.3.1. Các lệnh mặc định

Python cung cấp một số lệnh mặc định có thể sử dụng trực tiếp, ví dụ lệnh **print()** dùng để in một hoặc nhiều dữ liệu ra màn hình. Các bạn gõ lệnh **print("Hello world!!!")** và chạy lệnh sẽ thấy màn hình hiển thị dòng chữ "Hello world".

```
>>> print("Hello world!!!")
Hello world!!!
```

Dữ liệu mà lệnh **print()** in ra màn hình có thể là các số, xâu ký tự, biểu thức tính toán, giá trị các biến nhớ, hàm... Ngoài ra, chúng ta có thể in ra màn hình nhiều dữ liệu liên tiếp, các dữ liệu cách nhau bởi dấu phẩy.

```
>>> print("Toi co", 1 + 2, "qua tao")
Toi co 3 qua tao
```

Các bạn thử thay đổi từ khóa **print** thành **Print**, chương trình sẽ báo lỗi. Việc phân biệt chữ hoa, chữ thường trong Python như vậy được gọi là Case-sensitive, khi lập trình các bạn cần lưu ý gõ đúng các ký tự giống trong sách.

```
>>> Print("Xin chao")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Print' is not defined. Did you mean: 'print'?
```

Các lệnh mặc định khác như **input()**, **str()**, **int()**... sẽ được giới thiệu trong các phần tiếp theo.

2.3.2. Biến nhớ

Trong các ngôn ngữ lập trình, biến nhớ (gọi tắt là biến – variable) là tên được đặt cho các giá trị nhất định mà chúng ta muốn lưu trữ. Trong Python, việc tạo biến được thực hiện bằng phép gán giá trị cho biến và không cần khai báo trước với cú pháp: **<tên biến> = <giá trị>**. Ví dụ dưới đây tạo một biến có tên là **so_hang** và đặt giá trị bằng 4. Các bạn có thể gõ trực tiếp tên biến và nhấn Enter để hiển thị giá trị biến mà không cần sử dụng lệnh **print()**.

```
>>> so_hang = 4
>>> so_hang
4
```

Giá trị của biến có thể là số, xâu ký tự, giá trị logic, biểu thức... và có thể được thay đổi.

```
>>> x = 5
>>> x
5
>>> x = "xin chao"
>>> x
'xin chao'
>>> x = 3.14
>>> x
3.14
```

Lệnh gán là bắt buộc để có thể tạo biến, nếu sử dụng biến chưa được tạo sẽ gây ra lỗi, ví dụ dưới đây biến **b** chưa được tạo nên không thể sử dụng. Chúng ta có thể sử dụng lệnh **print()** để in ra màn hình giá trị biến.

```
>>> a = 10
>>> print(a)
10
>>> print(b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
```

Các bạn lưu ý, chúng ta nên có thói quen tốt là đặt tên biến theo đúng ý nghĩa, vai trò của biến đó. Tên biến được đặt theo quy tắc:

- Gồm các chữ cái tiếng Anh, chữ số và dấu gạch dưới.
- Không bắt đầu bằng chữ số.
- Phân biệt chữ hoa và chữ thường (Case-sensitive), biến **so_hang** sẽ khác biến **So_hang.**

2.3.3. Các kiểu dữ liệu cơ bản

Các ngôn ngữ lập trình hầu hết đều hỗ trợ các kiểu dữ liệu cơ bản sau:

- **int**: Kiểu số nguyên, ví dụ: 1, 2, 25, 1000...
- **float**: Kiểu số thực, ví dụ: 0.4, 12,5, 3.14... Lưu ý trong lập trình sử dụng dấu chấm để ngăn cách giữa phần nguyên và phần thập phân thay vì sử dụng dấu phẩy như môn Toán.
- **str**: Kiểu xâu ký tự, được biểu diễn trong cặp dấu nháy đơn ' ' hoặc nháy kép " ", ví dụ: "Hello world!", 'xin chao'...

- **bool**: Kiểu logic, chỉ có 2 giá trị True hoặc False, với ý nghĩa là giá trị đúng và giá trị sai. Ví dụ dưới đây biến **a** có giá trị True.

```
>>> a = 5 > 2
>>> a
True
```

2.3.4. Các phép toán cơ bản

Dưới đây là các phép toán thường gặp trong Python.

Phép toán	Ý nghĩa
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Phép chia lấy phần dư
//	Phép chia lấy phần nguyên
**	Phép lũy thừa

Với kiểu dữ liệu số nguyên và số thực, các phép toán được thực hiện như bình thường, nhưng với kiểu dữ liệu xâu ký tự thì phép cộng hai xâu ký tự là phép ghép xâu, phép nhân một xâu với một số nguyên là phép lặp xâu.

```
>>> s = "Học viện "
>>> s + "VIETSTEM"
'Học viện VIETSTEM'
>>> 'Toán' * 5
'ToánToánToánToánToán'
```

Khi thực hiện các phép toán, các bạn cần chú ý đến kiểu dữ liệu để tránh xảy ra lỗi. Ví dụ ta có bài toán:

Nhập 2 số hạng từ bàn phím, in ra màn hình tổng của 2 số đó.

Đầu tiên, ta sử dụng lệnh mặc định **input()** để thực hiện nhập dữ liệu từ bàn phím và lưu vào một biến với cú pháp **<biến> = input([nội dung thông báo])**. Sau khi chạy, thông báo sẽ hiện kèm dấu nháy chờ người dùng nhập giá trị từ bàn phím, sau khi nhấn Enter thì biến được gán giá trị đó với kiểu xâu ký tự.

```
>>> so_hang_1 = input("Nhập số hạng thứ nhất: ")
Nhập số hạng thứ nhất: 12
>>> so_hang_1
'12'
```

Tiếp theo, chúng ta nhập số hạng thứ 2 và thực hiện in ra tổng, nhưng kết quả sẽ không như ý muốn vì kiểu dữ liệu của 2 biến so_hang_1 và so_hang_2 là xâu ký tự nên kết quả sẽ là phép ghép xâu.

```
>>> so_hang_2 = input("Nhập số hạng thứ hai: ")
Nhập số hạng thứ hai: 28
>>> print(so_hang_1 + so_hang_2)
1228
```

Để thực hiện tính tổng, chúng ta cần chuyển kiểu dữ liệu xâu thành kiểu dữ liệu số nguyên bằng lệnh **int()** với cú pháp **int(<xâu ký tự>)**. Chúng ta có thể chuyển ngay khi gán giá trị cho biến hoặc khi in ra màn hình. Ngược lại, để chuyển kiểu dữ liệu số thành kiểu dữ liệu xâu ký tự, các bạn dùng lệnh **str()** với cú pháp tương tự.

```
>>> print(int(so_hang_1) + int(so_hang_2))
40
```

2.3.5. Các từ khóa trong Python

Trong một ngôn ngữ lập trình bất kỳ, luôn có một số các từ được sử dụng với mục đích riêng cố định và không được sử dụng vào mục đích khác như đặt làm tên biến, tên hàm... Dưới đây là một số từ khóa trong Python 3, danh sách này có thể được mở rộng thêm các từ mới tùy thuộc vào sự phát triển các phiên bản tiếp theo của Python.

False	**await**	**else**	**import**	**pass**	**assert**	**del**
None	**break**	**except**	**in**	**raise**	**async**	**elif**
True	**class**	**finally**	**is**	**return**	**global**	**not**
and	**continue**	**for**	**lambda**	**try**	**if**	**or**
as	**def**	**from**	**nonlocal**	**while**	**with**	**yield**