**Câu 4: Python hỗ trợ những kiểu dữ liệu cơ bản nào?**

**Kiểu dữ liệu số (Numeric Types)**:

* **int**: Số nguyên, ví dụ: 1, -2.
* float: Số thực, ví dụ: 1.5, -3.14.
* **complex**: Số phức, ví dụ: 3 + 5j.

**Kiểu dữ liệu chuỗi (Text Type)**:

* **str**: Chuỗi ký tự, ví dụ: "Hello, World!".

**Kiểu dữ liệu danh sách (Sequence Types)**:

* **list**: Danh sách, ví dụ: [1, 2, 3].
* **tuple**: Bộ giá trị, ví dụ: (1, 2, 3).
* **range**: Dãy số, ví dụ: range(5).

**Kiểu dữ liệu ánh xạ (Mapping Type)**:

* **dict**: Từ điển, ví dụ: {"key": "value"}.

**Kiểu dữ liệu tập hợp (Set Types)**:

* **set**: Tập hợp, ví dụ: {1, 2, 3}.
* **frozenset**: Tập hợp bất biến, ví dụ: frozenset({1, 2, 3}).

**Kiểu dữ liệu boolean (Boolean Type)**:

* **bool**: Giá trị đúng hoặc sai, ví dụ: True, False.

**Kiểu dữ liệu nhị phân (Binary Types)**:

* **bytes**: Dữ liệu nhị phân, ví dụ: b"hello".
* **bytearray**: Mảng byte, ví dụ: bytearray(5).
* **memoryview**: Khung nhìn bộ nhớ, ví dụ: memoryview(b"hello").

**Câu 5: Trình bày các loại ghi chú trong Python.**

Trong Python, có ba loại ghi chú chính mà bạn có thể sử dụng để làm cho mã của mình dễ hiểu hơn:

**Ghi chú một dòng (Single-line comments)**:

* Được bắt đầu bằng dấu # và kéo dài đến hết dòng.

**Ghi chú nhiều dòng (Multi-line comments)**:

* Có thể sử dụng nhiều dấu # liên tiếp hoặc sử dụng chuỗi ký tự ba dấu nháy kép (""") hoặc ba dấu nháy đơn (''').

**Ghi chú cùng dòng (Inline comments)**:

* Được đặt trên cùng một dòng với mã, sau dấu #.

**Câu 6: Trình bày ý nghĩa toán tử /, //, %, \*\*, and, or, is.**

1. **Toán tử chia /**:
   * Dùng để chia hai số và trả về kết quả là một số thực (float).
   * Ví dụ: 10 / 3 sẽ trả về 3.3333333333333335.
2. **Toán tử chia lấy phần nguyên //**:
   * Dùng để chia hai số và trả về phần nguyên của kết quả.
   * Ví dụ: 10 // 3 sẽ trả về 3.
3. **Toán tử chia lấy phần dư %**:
   * Dùng để chia hai số và trả về phần dư của phép chia.
   * Ví dụ: 10 % 3 sẽ trả về 1.
4. **Toán tử lũy thừa \*\***:
   * Dùng để tính lũy thừa của một số.
   * Ví dụ: 2 \*\* 3 sẽ trả về 8.
5. **Toán tử logic and**:
   * Dùng để kiểm tra nếu cả hai biểu thức đều đúng (True).
   * Ví dụ: True and False sẽ trả về False.
6. **Toán tử logic or**:
   * Dùng để kiểm tra nếu ít nhất một trong hai biểu thức đúng (True).
   * Ví dụ: True or False sẽ trả về True.
7. **Toán tử so sánh is**:
   * Dùng để kiểm tra nếu hai biến tham chiếu đến cùng một đối tượng trong bộ nhớ.
   * Ví dụ:

a = [1, 2, 3]

b = a

c = [1, 2, 3]

print(a is b)  # True

print(a is c)  # False

**Câu 7: Trình bày cách nhập dữ liệu bằng bàn phím.**

Để nhập dữ liệu từ bàn phím trong Python, bạn có thể sử dụng hàm input(). Hàm này cho phép bạn nhận dữ liệu từ người dùng dưới dạng chuỗi ký tự (string). Dưới đây là một số ví dụ cơ bản:

**Nhập một chuỗi ký tự:**

* name = input("Nhập tên của bạn: ")
* print("Tên của bạn là:", name)

**Nhập một số nguyên:**

* age = int(input("Nhập tuổi của bạn: "))
* print("Tuổi của bạn là:", age)

**Nhập một số thực:**

* height = float(input("Nhập chiều cao của bạn (m): "))
* print("Chiều cao của bạn là:", height)

**Nhập nhiều giá trị cùng lúc:**

* x, y = input("Nhập hai số, cách nhau bởi dấu cách: ").split()
* x = int(x)
* y = int(y)
* print("Tổng của hai số là:", x + y)

**Nhập nhiều giá trị cùng lúc và chuyển đổi kiểu dữ liệu:**

* x, y = map(int, input("Nhập hai số nguyên, cách nhau bởi dấu cách: ").split())
* print("Tổng của hai số là:", x + y)

Hàm input() sẽ tạm dừng chương trình cho đến khi người dùng nhập dữ liệu và nhấn Enter. Dữ liệu nhập vào luôn được coi là chuỗi, vì vậy bạn cần chuyển đổi nó sang kiểu dữ liệu mong muốn (như int hoặc float) nếu cần.

**Câu 8: Trình bày các loại lỗi khi lập trình và cách bắt lỗi trong Python?**

Trong Python, có hai loại lỗi chính mà bạn có thể gặp phải khi lập trình:

1. **Lỗi cú pháp (Syntax Errors)**:
   * Xảy ra khi mã nguồn không tuân thủ cú pháp của ngôn ngữ Python.
   * Ví dụ:
   * print "Hello, World!"  # Thiếu dấu ngoặc đơn trong Python 3
   * Khi gặp lỗi cú pháp, Python sẽ dừng thực thi và hiển thị thông báo lỗi.
2. **Lỗi thực thi (Runtime Errors)**:
   * Xảy ra trong quá trình chương trình đang chạy.
   * Ví dụ:
   * x = 10 / 0  # Lỗi chia cho 0
   * Lỗi thực thi thường khó phát hiện hơn lỗi cú pháp vì chúng chỉ xuất hiện khi chương trình chạy đến đoạn mã gây lỗi.

**Cách bắt lỗi trong Python**

Python cung cấp cơ chế xử lý ngoại lệ (exception handling) để bắt và xử lý lỗi thực thi. Bạn có thể sử dụng các khối try, except, else, và finally để quản lý lỗi:

1. **Khối try và except**:
   * Dùng để bắt và xử lý ngoại lệ.
2. **Khối else**:
   * Được thực thi nếu không có ngoại lệ nào xảy ra trong khối try.
3. **Khối finally**:
   * Luôn được thực thi, bất kể có ngoại lệ xảy ra hay không.
4. **Bắt nhiều loại ngoại lệ**:
   * Bạn có thể bắt nhiều loại ngoại lệ khác nhau bằng cách sử dụng nhiều khối except.

Việc sử dụng cơ chế xử lý ngoại lệ giúp chương trình của bạn trở nên ổn định hơn và dễ dàng xử lý các tình huống lỗi một cách linh hoạt.

**Câu 9: Giải thích kết quả tính toán cho các biểu thức.**

Đề bài cho:

i1 = 2

i2 = 5

i3 = -3

d1 = 2.0

d2 = 5.0

d3 = -0.5

Các biểu thức và kết quả:

**(a) i1 + (i2 \* i3)**

Tính toán: i2 \* i3 = 5 \* (-3) = -15

Sau đó: i1 + (-15) = 2 + (-15) = -13

Kết quả: -13

**(b) i1 \* (i2 + i3)**

Tính toán: i2 + i3 = 5 + (-3) = 2

Sau đó: i1 \* 2 = 2 \* 2 = 4

Kết quả: 4

**(c) i1 / (i2 + i3)**

Tính toán: i2 + i3 = 5 + (-3) = 2

Sau đó: i1 / 2 = 2 / 2 = 1.0 (Phép chia luôn trả về kiểu float)

Kết quả: 1.0

**(d) i1 // (i2 + i3)**

Tính toán: i2 + i3 = 5 + (-3) = 2

Sau đó: i1 // 2 = 2 // 2 = 1 (Phép chia nguyên trả về kiểu int)

Kết quả: 1

**(e) i1 / i2 + i3**

Tính toán: i1 / i2 = 2 / 5 = 0.4

Sau đó: 0.4 + i3 = 0.4 + (-3) = -2.6

Kết quả: -2.6

**(f) i1 // i2 + i3**

Tính toán: i1 // i2 = 2 // 5 = 0 (Phép chia nguyên trả về int)

Sau đó: 0 + i3 = 0 + (-3) = -3

Kết quả: -3

**(g) 3 + 4 + 5 / 3**

Tính toán: 5 / 3 = 1.666... (Phép chia trả về kiểu float)

Sau đó: 3 + 4 + 1.666... = 8.666...

Kết quả: 8.666... hoặc 8.67 nếu làm tròn

**(h) 3 + 4 + 5 // 3**

Tính toán: 5 // 3 = 1 (Phép chia nguyên)

Sau đó: 3 + 4 + 1 = 8

Kết quả: 8

**(i) (3 + 4 + 5) / 3**

Tính toán: 3 + 4 + 5 = 12

Sau đó: 12 / 3 = 4.0 (Phép chia trả về kiểu float)

Kết quả: 4.0

**(j) (3 + 4 + 5) // 3**

Tính toán: 3 + 4 + 5 = 12

Sau đó: 12 // 3 = 4 (Phép chia nguyên trả về kiểu int)

Kết quả: 4

**(k) d1 + (d2 \* d3)**

Tính toán: d2 \* d3 = 5.0 \* (-0.5) = -2.5

Sau đó: d1 + (-2.5) = 2.0 + (-2.5) = -0.5

Kết quả: -0.5

**(l) d1 + d2 \* d3**

Tính toán: d2 \* d3 = 5.0 \* (-0.5) = -2.5

Sau đó: d1 + (-2.5) = 2.0 + (-2.5) = -0.5

Kết quả: -0.5

**(m) d1 / d2 - d3**

Tính toán: d1 / d2 = 2.0 / 5.0 = 0.4

Sau đó: 0.4 - d3 = 0.4 - (-0.5) = 0.4 + 0.5 = 0.9

Kết quả: 0.9

**(n) d1 / (d2 - d3)**

Tính toán: d2 - d3 = 5.0 - (-0.5) = 5.0 + 0.5 = 5.5

Sau đó: d1 / 5.5 = 2.0 / 5.5 = 0.3636...

Kết quả: 0.3636... hoặc 0.36 nếu làm tròn

**(o) d1 + d2 + d3 / 3**

Tính toán: d3 / 3 = -0.5 / 3 = -0.1666...

Sau đó: d1 + d2 - 0.1666... = 2.0 + 5.0 - 0.1666... = 6.8333...

Kết quả: 6.8333... hoặc 6.83 nếu làm tròn

**(p) (d1 + d2 + d3) / 3**

Tính toán: d1 + d2 + d3 = 2.0 + 5.0 - 0.5 = 6.5

Sau đó: 6.5 / 3 = 2.1666...

Kết quả: 2.1666... hoặc 2.17 nếu làm tròn

**(q) d1 + d2 + (d3 / 3)**

Tính toán: d3 / 3 = -0.5 / 3 = -0.1666...

Sau đó: d1 + d2 - 0.1666... = 2.0 + 5.0 - 0.1666... = 6.8333...

Kết quả: 6.8333... hoặc 6.83 nếu làm tròn

**(r) 3 \* (d1 + d2) \* (d1 - d3)**

Tính toán: d1 + d2 = 2.0 + 5.0 = 7.0

Tính toán: d1 - d3 = 2.0 - (-0.5) = 2.0 + 0.5 = 2.5

Sau đó: 3 \* 7.0 \* 2.5 = 21.0 \* 2.5 = 52.5

Kết quả: 52.5

**Câu 10: Hãy viết ngắn gọn lại các lệnh dưới đây.**

Dưới đây là cách viết ngắn gọn các lệnh trong Python, sử dụng các toán tử gán rút gọn:

**(a) \*\*`x = x + 1`\*\***

   Viết ngắn gọn: `x += 1`

**(b) \*\*`x = x / 2`\*\***

   Viết ngắn gọn: `x /= 2`

**(c) \*\*`x = x - 1`\*\***

   Viết ngắn gọn: `x -= 1`

**(d) \*\*`x = x + y`\*\***

   Viết ngắn gọn: `x += y`

**(e) \*\*`x = x - (y + 7)`\*\***

   Viết ngắn gọn: `x -= (y + 7)`

Các phép toán trên là những toán tử gán rút gọn giúp làm mã nguồn ngắn gọn và dễ đọc hơn.