**Bài 16: Ngôn ngữ lập trình bậc cao và Python**

**1. Ngôn ngữ lập trình bậc cao**

- Ngôn ngữ lập trình bậc cao có các câu lệnh được viết gắn với ngôn ngữ tự nhiên giúp cho việc đọc, hiểu chương trình dễ dàng hơn.

- Python là một ngôn ngữ lập trình bậc cao phổ biến trong nghiên cứu và giáo dục. Python có mã nguồn mở nên thu hút nhiều nhà khoa học cùng phát triển.

**2. Môi trường lập trình Python**

Môi trường lập trình Python có hai chế độ:

- Chế độ gõ lệnh trực tiếp thường dùng để tính toán và kiểm tra nhanh các lệnh.

- Chế độ soạn thảo dùng để viết các chương trình có nhiều dòng lệnh.

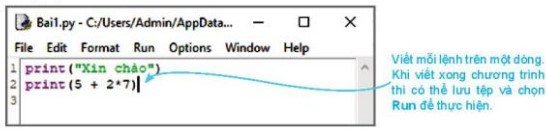
**a) Chế độ gõ lệnh trực tiếp**

- Trong một phiên làm việc với Python, gõ lệnh trực tiếp sau dấu nhắc >>> và nhấn phím Enter để thực hiện lệnh.

Lý thuyết Tin học 10 Bài 16: Ngôn ngữ lập trình bậc cao và Python - Kết nối tri thức  (ảnh 1)

**b) Chế độ soạn thảo**

- Soạn thảo chương trình hoàn chỉnh bằng cách chọn File/New File để mở ra màn hình soạn thảo.

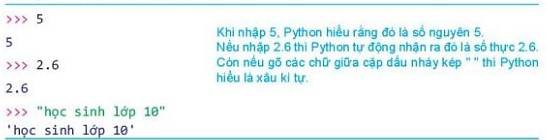


*Hình 1: Màn hình soạn thảo trong môi trường Python*

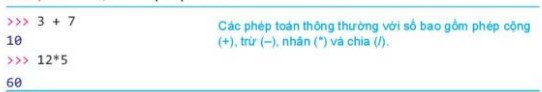
- Có thể soạn thảo chương trình Python bằng phần mềm soạn thảo văn bản hoặc phần mềm lập trình Python như Wingware, Pycharm, …

**3. Một số lệnh Python đầu tiên**

- Khi nhập giá trị số hoặc xâu kí tự từ dòng lệnh, Python tự nhận biết kiểu dữ liệu.



- Python có thể thực hiện các phép toán thông thường với số, phân biệt số thực và số nguyên.



- Lệnh print() có chức năng in dữ liệu ra màn hình, có thể in một hoặc nhiều giá trị đồng thời

Cú pháp lệnh print() như sau:

Lý thuyết Tin học 10 Bài 16: Ngôn ngữ lập trình bậc cao và Python - Kết nối tri thức  (ảnh 1)

Trong đó v1, v2, …, vn là các giá trị cần đưa ra màn hình.

**Thực hành**

**Nhiệm vụ:**Sử dụng chế độ soạn thảo chương trình của Python để tạo, nhập và chạy chương trình đầu tiên có tên Bai1.py:



**Hướng dẫn**

Bước 1: Nháy đúp chuột vào biểu tượng Lý thuyết Tin học 10 Bài 16: Ngôn ngữ lập trình bậc cao và Python - Kết nối tri thức  (ảnh 1) của Python để khởi động.

Bước 2: Chọn chế độ soạn thảo chương trình của môi trường lập trình Python: Trong môi trường lập trình Python, chọn **File/New File**.

Bước 3: Nhập nội dung chương trình như Hình sau:

Lý thuyết Tin học 10 Bài 16: Ngôn ngữ lập trình bậc cao và Python - Kết nối tri thức  (ảnh 1)

Bước 4: Chọn File/Save hoặc nhấn tổ hợp phím Ctrl + S để lưu tệp.

Bước 5: Chọn **Run/Run module** hoặc nhấn phím S5 để thực hiện chương trình.

Bước 6: Để kết thúc một phiên làm việc, nháy nút Lý thuyết Tin học 10 Bài 16: Ngôn ngữ lập trình bậc cao và Python - Kết nối tri thức  (ảnh 1) ở góc trên bên phải màn hình hoặc gõ lệnh quit() hoặc lệnh exit() rồi nhấn ENTER. Ví dụ:

>>>quit()

**Bài 17: Biến và lệnh gán**

**1. Biến và lệnh gán**

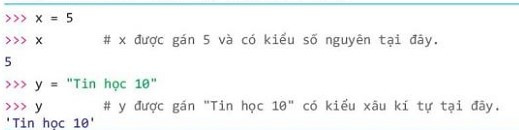
- Biến là tên của một vùng nhớ dùng để lưu trực giá trị (dữ liệu) và giá trị đó có thể được thay đổi khi thực hiện chương trình.

- Cú pháp lệnh gán:

<biến> = <biểu thức>

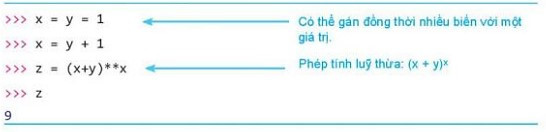
- Khi thực hiện lệnh gán, <giá trị> bên phải sẽ được gán cho <biến>. Nếu biến chưa được khai báo thì nó sẽ được khởi tạo khi thực hiện câu lệnh gán, trong Python không cần khai báo trước kiểu dữ liệu cho biến.

- Ví dụ:



- Có thể thực hiện tất cả các phép toán thông thường như +, -, \*, /, … trên các biến có cùng kiểu dữ liệu.

Ví dụ:

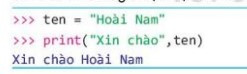


- Có thể gán giá trị cho biến thông qua tính toán giá trị của biểu thức với các biến đã được xác định trước.

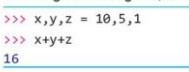
Ví dụ:



- Tên biến đặt dễ nhớ và có ý nghĩa.



- Có thể gán nhiều giá trị đồng thời cho nhiều biến.



- Cú pháp của lệnh gán đồng thời như sau:

Lý thuyết Tin học 10 Bài 17: Biến và lệnh gán - Kết nối tri thức  (ảnh 1)

- Quy tắc đặt tên biến (định danh):

+ Chỉ gồm các chữ cái tiếng Anh, các chữ số từ 0 đến 9 và kí tự gạch dưới \_

+ Không bắt đầu bằng chữ số.

+ Phân biệt chữ hoa và chữ thường.

**2. Các phép toán trên một số kiểu dữ liệu cơ bản**

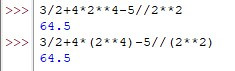
 - Các phép toán trên kiểu dữ liệu số: +, -, \*, /, //, %, \*\*



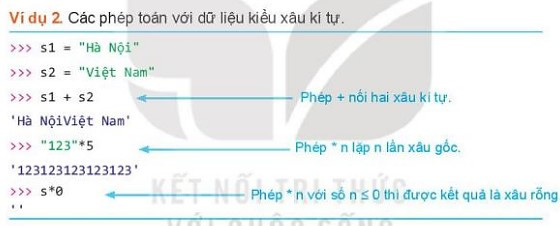
*Ví dụ 1: Các phép toán trên kiểu dữ liệu số*

- Thứ tự các phép tính như sau: phép lũy thừa \*\* có ưu tiên cao nhất, sau đó /, \*, //, % cuối cùng là +, -.

Ví dụ:



- Các phép toán trên kiểu dữ liệu xâu: + và \*



*Ví dụ 2: Các phép toán với dữ liệu kiểu xâu kí tự*

**3. Từ khóa**

- Từ khóa là các từ đặc biệt tham gia vào cấu trúc của ngôn ngữ lập trình. Không được phép đặt tên biến hay các định danh trùng từ khóa.

- Một số từ khóa trong Python 3.x

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

**Thực hành:**Tạo và làm việc với biến, tính toán với các kiểu dữ liệu cơ bản trong Python

**Nhiệm vụ 1:**Thực hiện các phép tính sau trong môi trường lập trình Python, so sánh kết quả với việc tính biểu thức toán học

a) (1+2+3+...+10)3(1+2+3+...+10)3 .

b) 1/2 + 1/3 + 1/4 + 1/5.

c) Thực hiện lệnh gán x = 2, y = 5 rồi tính giá trị biểu thức (x+y)(x2+y2-1).

d) Thực hiện lệnh gán a = 2, b = 3, c= 4 rồi tính giá trị biểu thức (a+b+c)(a+b-c).

**Hướng dẫn**

Thực hiện như sau:

>>> (1+2+3+4+5+6+7+8+9+10)\*\*3

>>>1/2+1/3+1/4+1/5

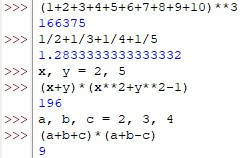
>>> x, y = 2, 5

>>> (x+y)\*(x\*\*2+y\*\*2-1)

>>> a, b, c = 2, 3, 4

>>> (a+b+c)\*(a+b-c)

Kết quả:



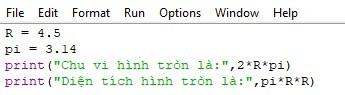
**Nhiệm vụ 2:**Gán giá trị cho biến R là bán kính hình tròn rồi viết chương trình tính và in ra kết quả theo mẫu

Chu vi hình tròn là: …

Diện tích hình tròn là: …

**Hướng dẫn**

Thực hiện chương trình và kiểm tra kết quả, so sánh với chế độ gõ lệnh trực tiếp.



**Bài 18: Các lệnh vào ra đơn giản**

**1. Các lệnh vào ra đơn giản**

- Các lệnh vào ra đơn giản của Python bao gồm lệnh input() và lệnh print()

+ Lệnh input() có chức năng nhập dữ liệu từ thiết bị vào chuẩn, thường là bàn phím. Nội dung nhập có thể là số, biểu thức, xâu và cho kết quả là một xâu kí tự.

Cú pháp: <biến> = input (<Dòng thông báo>)

+ Lệnh print() có chức năng đưa dữ liệu ra thiết bị chuẩn, thường là màn hình. Thông tin cần đưa ra có thể gồm một hay nhiều kiểu dữ liệu khác nhau, cho phép cả biểu thức tính toán.

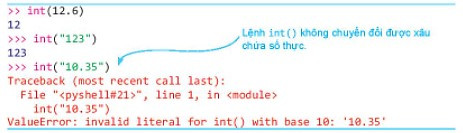
- Ví dụ:

**Lý thuyết Tin học 10 Bài 18: Các lệnh vào ra đơn giản - Kết nối tri thức  (ảnh 1)**

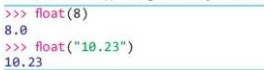
**2. Chuyển đổi kiểu dữ liệu cơ bản của Python**

- Một số kiểu dữ liệu cơ bản của python gồm int (số nguyên), float (số thực), str (xâu kí tự), bool (lôgic).

+ Lệnh int() có chức năng chuyển đổi số thực hoặc xâu chứa số nguyên thành số nguyên.



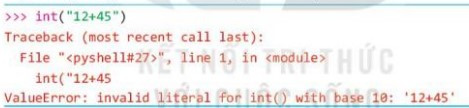
+ Lệnh float() dùng để chuyển đổi số nguyên và xâu kí tự thành số thực.



+ Lệnh str() dùng để chuyển đổi các kiểu dữ liệu khác thành xâu kí tự.



+ Chú ý các lệnh int(), float() chỉ có thể chuyển đổi các xâu ghi giá trị số trực tiếp, không chuyển đổi xâu có công thức.



- Nếu cần nhập số nguyên thì sau khi nhập giá trị số cần dùng lệnh int() để chuyển đổi sang kiểu số nguyên như sau:

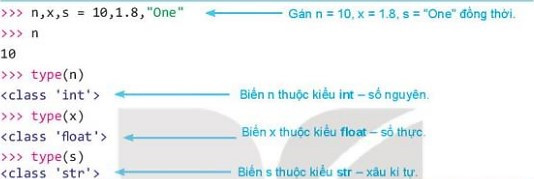
Lý thuyết Tin học 10 Bài 18: Các lệnh vào ra đơn giản - Kết nối tri thức  (ảnh 1)

- Nếu cần nhập số thực thì sau khi nhập giá trị số cần dùng lệnh float() để chuyển  đổi sang kiểu số thực như sau:

Lý thuyết Tin học 10 Bài 18: Các lệnh vào ra đơn giản - Kết nối tri thức  (ảnh 1)

- Lệnh type() dùng để nhận biết kiểu dữ liệu cơ bản và dữ liệu kiểu này chỉ có hai giá trị là True (đúng) hoặc False (sai).

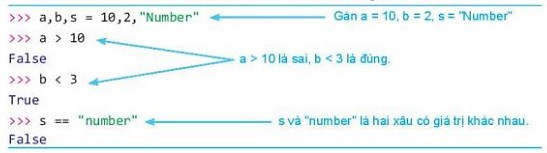
Ví dụ:



**Bài 19: Câu lệnh điều kiện if**

**1. Biểu thức lôgic**

- Biểu thức lôgic là biểu thức chỉ nhận giá trị True hoặc False. Giá trị biểu thức lôgic thuộc kiểu bool.



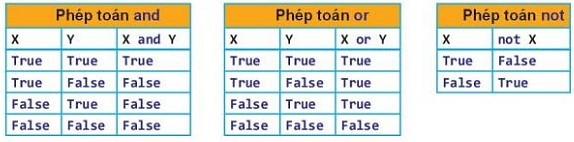
- Các phép so sánh các giá trị số trong Python:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| < | Nhỏ hơn | > | Lớn hơn | == | Bằng nhau |
| <= | Nhỏ hơn hoặc bằng | >= | Lớn hơn hoặc bằng | != | Khác nhau |

Chú ý: Với xâu kí tự cũng có đầy đủ các phép so sánh.

- Các phép toán trên kiểu dữ liệu lôgic gồm phép and (và), or (hoặc) và not (phủ định).

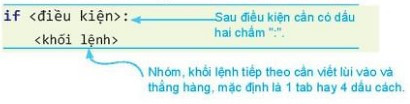
*Bảng 1: Các phép toán lôgic*



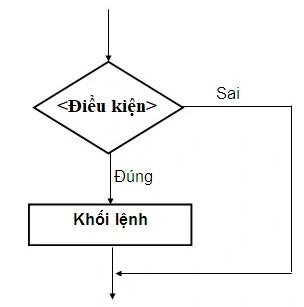
**2. Lệnh if**

- Để xử lí các tình huống rẽ nhánh, Python cũng có các câu lệnh để mô tả cấu trúc rẽ nhánh:

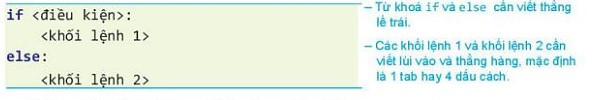
+ **Câu điều kiện dạng thiếu:**



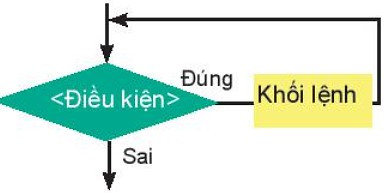
Khi thực hiện lệnh, Python sẽ kiểm tra <điều kiện> nếu đúng thì thực hiện <khối lệnh>, ngược lại thì bỏ qua chuyển sang lệnh tiếp theo sau lệnh if.



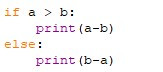
**+ Câu điều kiện dạng đầy đủ:**



Khi thực hiện lệnh, Python sẽ kiểm tra <điều kiện> nếu đúng thì thực hiện <khối lệnh 1>, ngược lại thì thực hiện <khối lệnh 2>.



- Ví dụ: Nếu hai số đã cho đã được tạo thì lệnh sau sẽ in ra giá trị tuyệt đối của hiệu hai số:

****

- Chú ý: Các khối lệnh trong Python đều cần viết sau dấu “:” và lùi vào, thẳng hàng.

**Bài 20: Câu lệnh lặp for**

**1. Lệnh lặp for**

- **for**là lệnh lặp với số lần biết trước. Số lần lặp thường được xác định bởi vùng giá trị của lệnh range( ).

- Lệnh range(n) trả lại vùng giá trị gồm n số từ 0 đến n – 1. Cú pháp của lệnh lặp với số lần biết trước for:

for *in range(n):*

<khối lệnh>

⇒ Khi thực hiện, ở mỗi vòng lặp biến i sẽ được gán lần lượt các giá trị trong vùng giá trị của lệnh range( ) và thực hiện <khối lệnh>.

*Ví dụ 1:* Tính tổng các số tự nhiên chẵn nhỏ hơn n, với n cho trước (n=10)



*Ví dụ 2:* Đếm số nguyên nhỏ hơn n (n = 20) và là bội của 3



**2. Lệnh range**

- Lệnh tạo vùng giá trị range() có các dạng sau:

+ range(stop) trả lại vùng giá trị từ 0 đến stop – 1.

+ range(start, stop) trả lại vùng giá trị từ start đến stop – 1.

Ví dụ: range(n) cho vùng gồm các số 0, 1, …, n – 1.

**Bài 21: Câu lệnh lặp while**

**1. Lệnh while**

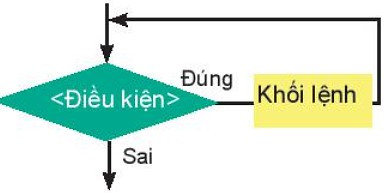
- Câu lệnh lặp while thực hiện khối lệnh với số lần lặp không biết trước. Khối lệnh lặp được thực hiện cho đến khi <điều kiện> = false

- Cú pháp:

while <điều kiện>:

          <khối lệnh lặp>

Trong đó, <điều kiện> là biểu thức lôgic. Khi thực hiện lệnh, Python sẽ kiểm tra <điều kiện>, nếu đúng thì thực hiện khối lệnh lặp, nếu sai thì kết thúc lệnh while.



- Ví dụ 1: Đoạn chương trình tính tổng bình phương các số tự nhiên nhỏ hơn 10.

S = 0

k = 1

while k\*k < 100:

S = S + k\*k

k = k + 1

**2. Cấu trúc lập trình**

Ba cấu trúc lập trình cơ bản của các ngôn ngữ lập trình bậc cao gồm:

- Cấu trúc tuần tự: gồm các lệnh được thực hiện theo trình tự từ trên xuống dưới, được thể hiện bằng các lệnh gán giá tị, nhập/xuất dữ liệu.

- Cấu trúc rẽ nhánh: chỉ được thực hiện tùy thuộc vào điều kiện nào đó là đúng hay sai, thể hiện bằng lệnh if.

- Cấu trúc lặp: được thực hiện lặp lại tùy theo điều kiện nào đó đúng hay sai, thể hiện bằng các lệnh for, while.

**Bài 22: Kiểu dữ liệu danh sách**

**1. Kiểu dữ liệu danh sách**

- List là kiểu dữ liệu danh sách (dãy, mảng) trong Python. Tạo list bằng lệnh gán với các phần tử trong cặp dấu ngoặc [].

- Kiểu dữ liệu danh sách trong Python được khởi tạo:

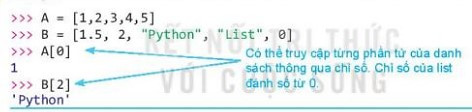
<tên list> = [<v1>, <v2>, …, <vn>]

Lưu ý: Các giá trị <vk> có thể có kiểu dữ liệu khác nhau (số nguyên, số thực, xâu kí tự, …).

- Có thể truy cập từng phần tử của danh sách thông qua chỉ số:

[<chỉ số>]

Ví dụ 1:



- Chỉ số bắt đầu từ 0 đến len( ) -1, trong đó len( ) là lệnh tính độ dài danh sách.

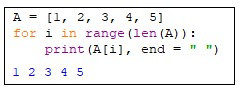
Ví dụ 2: Tạo danh sách rỗng (có độ dài 0) và các phép toán ghép danh sách (+)



**2. Duyệt các phần tử của danh sách**

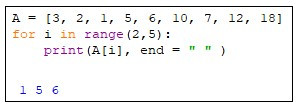
Có thể duyệt lần lượt các phần tử của danh sách bằng lệnh for kết hợp với vùng giá trị của lệnh range().

- Ví dụ 1: Duyệt và in ra từng phần tử của danh sách



⇒ Biến I chạy trên vùng chỉ số từ 0 đến len(A) -1.

- Ví dụ 2: Duyệt và in một phần của danh sách.

****

**3. Thêm phần tử vào danh sách**

- Python có một số lệnh dành riêng (phương thức) cho kiểu dữ liệu danh sách.

- Cú pháp:

.<phương thức>

- Lệnh thêm phần tử vào cuối danh sách là .append()

Ví dụ: Thêm phần tử vào cuối danh sách

Lý thuyết Tin học 10 Bài 22: Kiểu dữ liệu danh sách - Kết nối tri thức  (ảnh 1)

**Bài 23: Một số lệnh làm việc với dữ liệu danh sách**

**1. Duyệt danh sách với toán tử in**

- Toán tử in dùng để kiểm tra một phần tử có nằm trong danh sách đã cho không. Kết quả trả lại True (Đúng) hoặc False (Sai).

<giá trị> in

*Ví dụ 1:* Dùng toán tử in để kiểm tra <giá trị> có trong không?

>>> A = [1, 2, 3, 4, 5]

>>> 2 in A

True

>>> 10 in A

False

⇒ Số nguyên 2 nằm trong dãy A, kết quả trả lại True. Số 10 không nằm trong dãy A, kết quả trả lại False.

- Có thể duyệt nhanh từng phần tử của danh sách bằng toán tử in và lệnh for mà không cần sử dụng lệnh range().

*Ví dụ 2:* Sử dụng toán tử in để duyệt từng phần tử của danh sách

>>> A = [10, 11, 12, 13, 14, 15]

>>> for k in A:

print(k, end = “ “ )

10 11 12 13 14 15

⇒ Khi thực hiện lệnh này, biến k sẽ lần lượt nhận các giá trị từ dãy A.

**2. Một số lệnh làm việc với danh sách**

|  |  |
| --- | --- |
| A.append(x) | Bổ sung phần tử x và cuối danh sách A |
| A.insert(k, x) | Chèn phần tử x vào vị trí k của danh sách |
| A.clear() | Xóa toàn bộ dữ liệu của danh sách A |
| A.remove(x) | Xóa phần tử x từ danh sách A |

*- Ví dụ 1*: Lệnh clear() xóa toàn bộ danh sách

>>> A = [1, 2, 3, 4, 5]

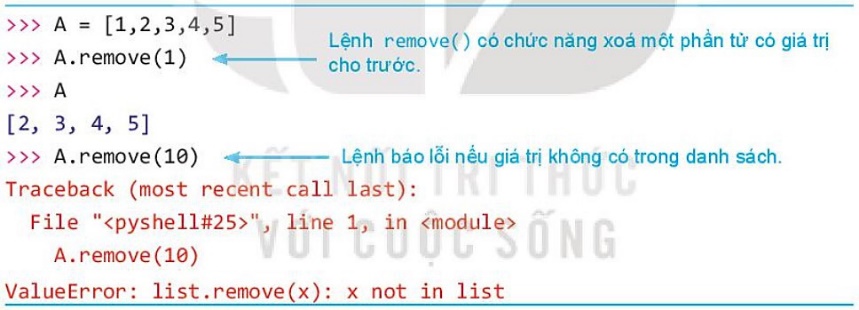
>>> A.clear()

>>> A

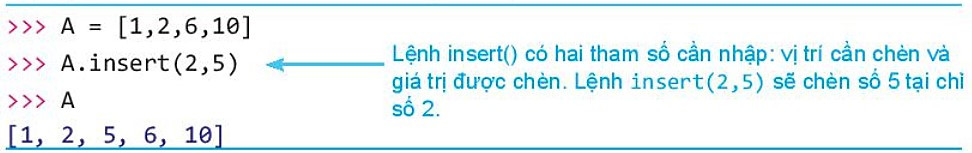
[ ]

⇒ Sau khi thực hiện lệnh clear( ), danh sách gốc trở thành rỗng.

*- Ví dụ 2:* Lệnh remove(value) xóa phần tử đầu tiên của danh sách có ghi giá trị value. Nếu không có phần tử nào thì sẽ báo lỗi.



*- Ví dụ 3:* Lệnh insert có chức năng chèn phần tử vào danh sách tại chỉ số cho trước.



**Bài 24: Xâu kí tự**

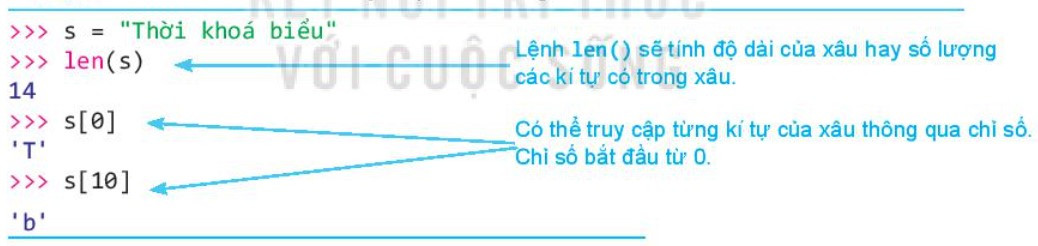
**1. Xâu là một dãy các kí tự**

- Xâu kí tự trong Python là dãy các kí tự Unicode.

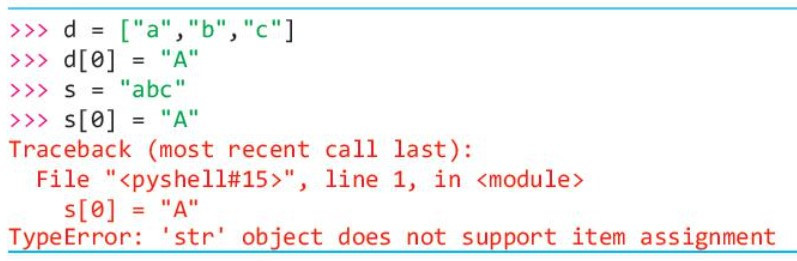
- Xâu có thể được coi là danh sách các kí tự nhưng không thay đổi từng kí tự của xâu.

- Truy cập từng kí tự của xâu qua chỉ số, chỉ số từ 0 đến độ dài len() -1

- Ví dụ 1: Xâu kí tự và cách truy cập đến từng kí tự của xâu



- Ví dụ 2:



- Python không cho phép thay đổi từng kí tự của một xâu (khác với danh sách).

- Python không có kiểu dữ kiệu kí tự.

- Xâu rỗng được định nghĩa: empty = " "

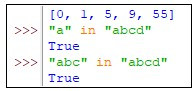
**2. Lệnh duyệt kí tự của xâu**

- Có thể duyệt các kí tự của xâu bằng lệnh for tương tự như với danh sách. S1 in S2 trả lại giá trị True nếu S1 là xâu con của S2. Có 2 cách duyệt:

+ Cách thứ nhất, biến i lần lượt chạy theo chỉ số của xâu kí tự, từ 0 đến len(s) – 1. Kí tự tại chỉ số là s[i].

+ Cách thứ hai duyệt theo từng kí tự của xâu s. Biến ch sẽ được gán lần lượt các kí tự của xâu s từ đầu đến cuối.

- Chú ý: Từ khóa in, tùy trường hợp cụ thể, hoặc là toán tử lôgic dùng để kiểm tra giá trị có mặt hay không trong một vùng giá trị/danh sách/xâu, hoặc để chọn lần lượt từng phần tử trong một vùng giá trị/danh sách/xâu.



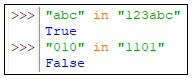
**Bài 25: Một số lệnh làm việc với xâu kí tự**

**1. Xâu con và lệnh tìm vị trí xâu con**

Biểu thức kiểm tra <xâu 1> nằm trong <xâu 2> là:

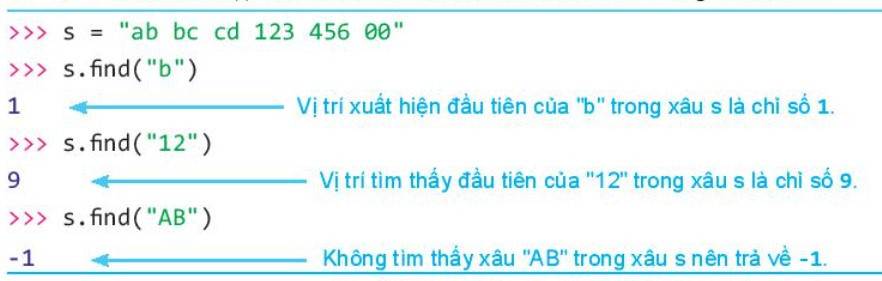
<xâu 1> in <xâu 2>

Ví dụ 1: Dùng toán tử in để kiểm tra một xâu có là xâu con của xâu khác không?



- Để tìm một xâu trong một xâu khác có thể dùng toán tử in hoặc lệnh find(). Lệnh find() trả về vị trí của xâu con trong xâu mẹ.

Ví dụ 2: Lệnh find() tìm vị trí xuất hiện của một xâu trong xâu khác.



- Cách thực hiện phương thức là:

<xâu>.<phương thức>

- Cú pháp đơn của lệnh find():

<xâu mẹ>.find(<xâu con>)

⇒ Lệnh sẽ tìm vị trí đầu tiên của xâu con trong xâu mẹ và trả về vị trí đó. Nếu không tìm thấy thì trả về -1.

- Cú pháp đầy đủ của lệnh find():

<xâu mẹ>.find(<xâu con>,start)

⇒ Lệnh sẽ tìm vị trí xâu con bắt đầu từ vị trí start.

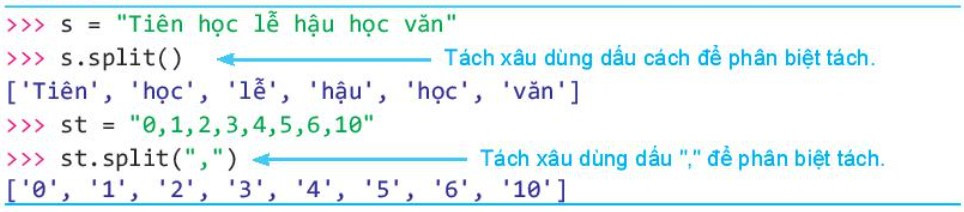
**2. Một số lệnh thường dùng với xâu kí tự**

- Python có các lệnh đặc biệt để xử lí xâu là split() dùng để tách xâu thành danh sách và lệnh join() dùng để nối danh sách các xâu thành một xâu.

**+ Cú pháp lệnh split( ):**

<xâu mẹ>.split(<kí tự tách>)

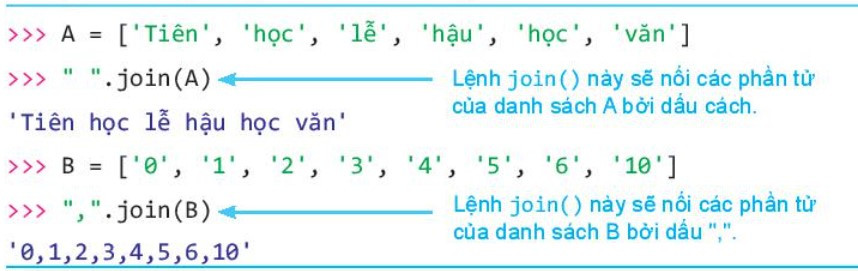
 Ví dụ 1: Lệnh split() tách một xâu thành danh sách các từ.



**+ Cú pháp lệnh joint( ):**

"kí tự nối ".joint()

Ví dụ 2: Lệnh join() nối danh sách các từ thành một xâu.



**Bài 26: Hàm trong Python**

**1. Một số hàm thiết kế sẵn của Python**

- Python cung cấp sẵn nhiều hàm thực hiện được những công việc khác nhau cho phép người dùng được tùy ý sử dụng khi viết chương trình bằng câu lệnh gọi hàm tương ứng.

**Bảng 1: Một số lệnh trong Python**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abs() | len() | range() | bool() | float() |
| list() | round() | chr() | input() | ord() |
| str() | divmod() | int() | print() | type() |

- Ví dụ:

+ Lệnh print(“Học mãi”) thực hiện việc in xâu kí tự “Học mãi” trong dấu ngoặc ra màn hình.

+ Lệnh int(“52”) chuyển xâu “52” thành số nguyên 52.

+ Lệnh type(y) trả lại kiểu dữ liệu của biến y.

**2. Thiết lập các hàm tự định nghĩa**

- Hàm trong Python được định nghĩa bằng từ khóa def, theo sau là tên hàm (tên hàm sẽ theo quy tắc đặt tên định danh).

- Hàm có thể có hoặc không có tham số. Khối lệnh mô tả hàm được viết sau dấu “:” và viết lùi vào, thẳng hàng. Hàm có thể có hoặc không có giá tị trả lại sau từ khóa return.

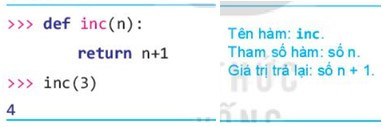
*+ Cú pháp thiết lập hàm có trả lại giá trị:*

def <tên hàm>():

<khối lệnh>

return <giá trị>

*Ví dụ 1:* Cách viết hàm có trả lại giá trị



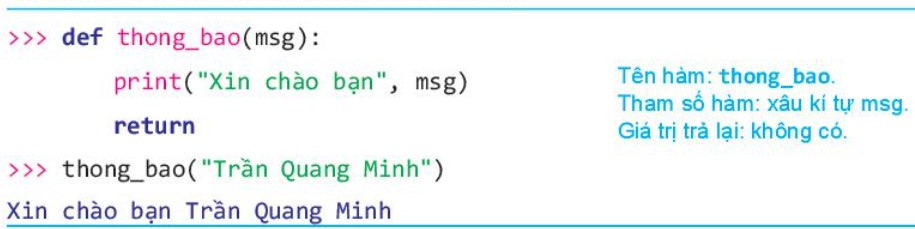
*+ Cú pháp thiết lập hàm không trả lại giá trị:*

def <tên hàm>():

<khối lệnh>

return

*Ví dụ 2:* Cách viết hàm không trả lại giá trị



**Bài 27: Tham số của hàm**

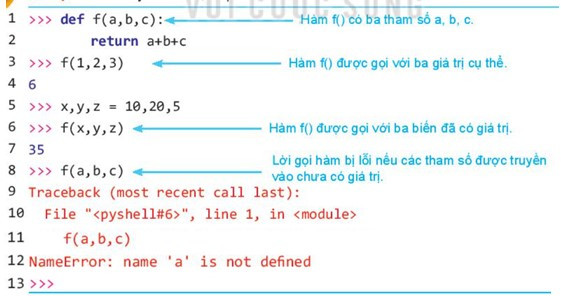
**1. Tham số và đối số của hàm**

- Tham số của hàm được định nghĩa khi khai báo hàm và được dùng như biến trong định nghĩa hàm.

- Đối số là giá trị được truyền vào khi gọi hàm.

- Khi gọi hàm, các tham số (parameter) sẽ được truyền bằng giá trị thông qua đối số (argument) của hàm, số lượng giá trị được truyền vào hàm bằng với số tham số trong khai báo của hàm.

- Ví dụ: Cách truyền dữ liệu qua tham số



**2. Cách sử dụng chương trình con**

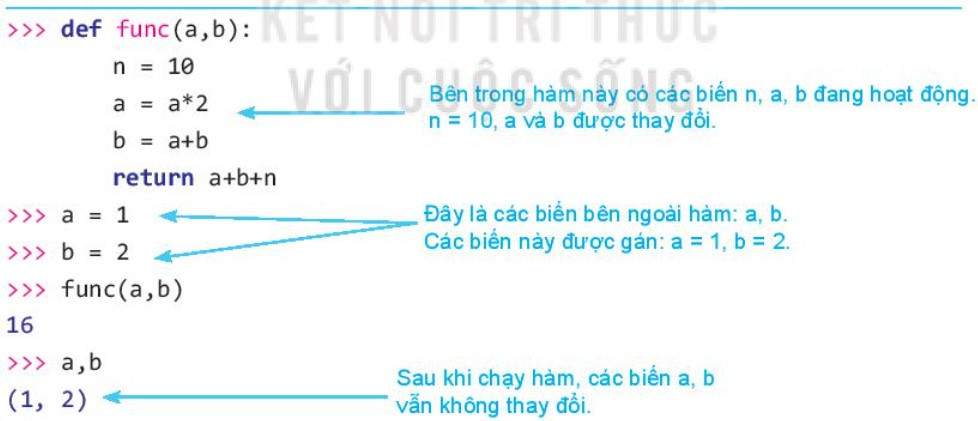
- Sử dụng chương trình con có thể giúp phân chia việc giải một bài toán lớn thành giải quyết các bài toán nhỏ và phát huy được tinh thần làm việc nhóm.

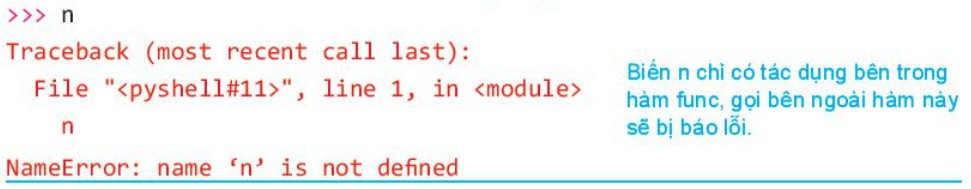
- Chương trình chính có cấu trúc rõ ràng, dễ hiểu hơn, nếu cần hiệu chỉnh, phát triển và nâng cấp cũng thuận tiện hơn.

**Bài 28: Phạm vi của biến**

**1. Phạm vi của biến trong khai báo hàm**

Trong Python tất cả các biến khai thác bên trong hàm đều có tính địa phương (cục bộ), không có hiệu lực ở bên ngoài hàm.

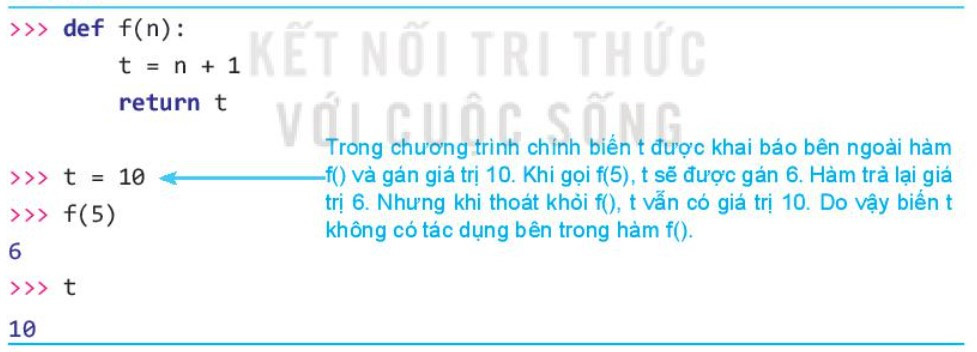




**2. Phạm vi của biến khai báo ngoài hàm**

- Biến đã khai báo bên ngoài hàm chỉ có thể truy cập giá trị để sử dụng bên trong hàm mà không làm thay đổi được giá trị của biến đó (trừ trường hợp với từ khóa global).

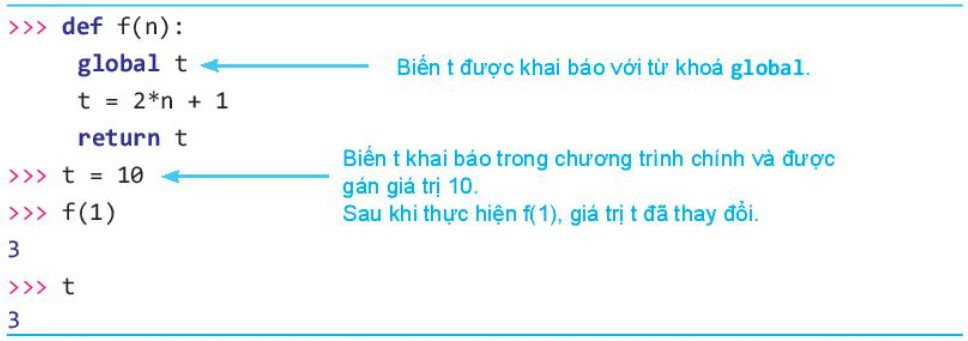
Ví dụ 1: Biến khai báo bên ngoài hàm không có tác dụng bên trong hàm.



Ví dụ 2: Bên trong hàm có thể truy cập đến sử dụng giá trị của biến đã khai báo trước đó ở bên ngoài hàm.



- Ví dụ 3: Dùng từ khóa global.

****

**Bài 29: Nhận biết lỗi chương trình**

**1. Nhận biết lỗi chương trình**

- Có thể phân biệt lỗi chương trình Python làm ba loại:

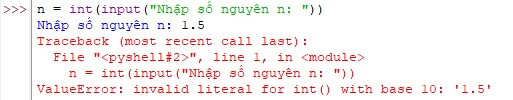
1. Lỗi khi có lệnh viết sai cú pháp hoặc sai cấu trúc ngôn ngữ Python quy định. Chương trình sẽ lập tức dừng và thông báo lỗi **Syntax Error**.

*Ví dụ 1:* Sai cú pháp lệnh

Lý thuyết Tin học 10 Bài 29: Nhận biết lỗi chương trình - Kết nối tri thức  (ảnh 1)

2. Lỗi khi không thể thực hiện một lệnh trong chương trình. Chương trình dừng lại và thông báo một mã lỗi. Lỗi này gọi lỗi ngoại lệ (**Exceptions Error**), mã lỗi trả lại gọi là mã lỗi ngoại lệ.

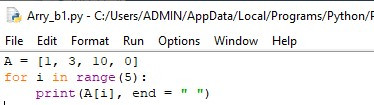
*Ví dụ 2:* Người dùng nhập sai dữ liệu



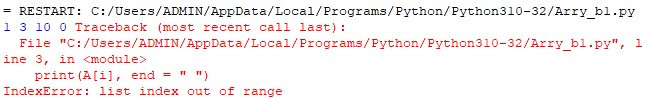
3. Chương trình chạy không lỗi ngoại lệ, nhưng kết quả đưa ra sai, không chính xác. Đây là lỗi lôgic bên trong chương trình.

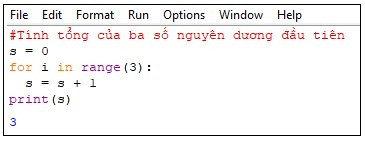
*Ví dụ 3:* Chương trình thông báo lỗi chỉ số vượt quá giới hạn cho phép

Chương trình:



Khi chạy chương trình sẽ báo lỗi:

     *Ví dụ 4:* Chương trình có lỗi lôgic



Chương trình cho kết quả là 3 mà kết quả đúng là 1 + 2 + 3 = 6. Lí do là hàm range(3) trả lại vùng giá trị là 0, 1, 2 chứ không phải 1, 2, 3.

- Với mỗi loại lỗi, cách xử lí và kiểm soát lỗi sẽ khác nhau.

**2. Một số lỗi ngoai lệ thường gặp**

**Bảng 1: Một số lỗi ngoại lệ thường gặp**

|  |  |
| --- | --- |
| **Mã lỗi ngoại lệ** | **Mô tả lỗi** |
| ZeroDivisionError | Khi thực hiện lệnh phép chia cho 0. |
| IndexError | Lệnh cố gắng truy cập phần tử của danh sách nhưng chỉ số vượt quá giới hạn. |
| NameError | Chương trình muốn tìm một tên nhưng không thấy. Ví dụ lệnh gọi một hàm nhưng không có hàm đó. |
| TypeError | Lỗi kiểu dữ liệu.  - Lệnh truy cập một phần tử của danh sách những chỉ số không là số nguyên.  - Lệnh tính biểu thức số nhưng toán hạng không phải là số. |
| ValueError | Lỗi liên quan đến giá trị của đối tượng.  Lỗi khi thực hiện lệnh chuyển đổi kiểu dữ liệu, đổi số của hàm có giá trị mà hàm không hỗ trợ. Ví dụ lệnh int(“1.55”). |
| IndentationError | Lỗi khi các dòng lệnh thụt vào không hàng hoặc không đúng vị trí. |
| SyntaxError | Lỗi cú pháp. |

**Bài 30: Kiểm thử và gỡ lỗi chương trình**

**1. Một vài phương pháp kiểm thử chương trình**

**a) Quan sát mã lỗi Runtime và bắt lỗi ngoại lệ**

Nếu chương trình có lỗi Runtime (đang chạy mà dừng lại), cần quan sát các mã lỗi để kiểm tra vị trí dòng lệnh sinh lỗi này. Từ đó tìm và sửa lỗi.

**b) Kiểm thử chương trình với các bộ dữ liệu test**

- Chương trình cần được thử với một số bộ dữ liệu test gồm đầu vào tiêu biểu phụ thuộc đặc thù của bài toán và kết quả đầu ra đã biết trước.

- Cần chú ý:

+ Cần có nhiều bộ test (theo các tiêu chí khác nhau).

+ Cần có bộ test ngẫu nhiên.

+ Cần có bộ test dữ liệu ở vùng biên. Ví dụ dữ liệu đầu vào là cặp (x, y) xác định trên miền [0, 1]. Khi đó cần kiểm tra với bộ dữ liệu biên (0, 0), (0; 1), (1; 0), (1; 1). Thực tế cho thấy lỗi thường phát sinh tại vùng biên hoặc lân cận của biên.

**c) In các thông số trung gian**

- Bổ sung vào giữa các dòng lệnh câu lệnh print() để in ra biến trung gian, kiểm tra quy trình hay thuật toán.

- Giả sử chương trình có đầu vào là (x1, x2) đầu ra là (a1, a2) nhưng sử sụng biến trung gian (y1, y2). Khi đó bổ sung thêm các dòng lệnh để in ra các giá trị trung gian, từ đó sẽ dễ tìm lỗi hơn.

**d) Sử dụng công cụ break point (điểm dừng)**

- Công cụ break point cho phép tạo ra các điểm dừng bên trong chương trình. Khi chạy, chương trình sẽ tạm dừng tại các điểm dừng cho phép người kiểm thử có thể quan sát các thông tin khác bên trong chương trình, kiểm tra tính đúng đắn của chương trình.

**Bài 31: Thực hành viết chương trình đơn giản**

**Nhiệm vụ 1:**Viết chương trình nhập số tự nhiên n, kiểm tra xem n có là số nguyên tố không. Nếu n là hợp số thì in ra kết quả phân tích n thành tích các thừa số nguyên tố. Chú ý số 1 không là số nguyên tố cũng không là hợp số.

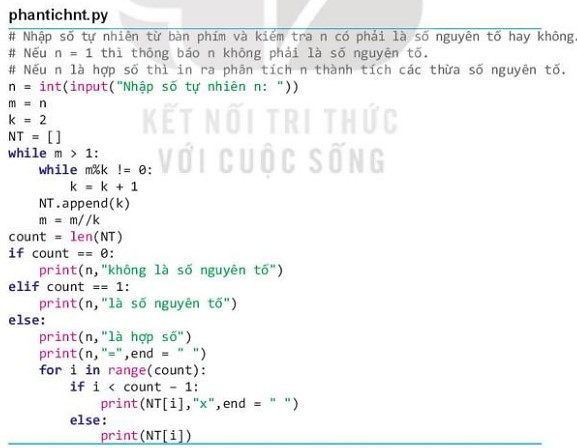
**Hướng dẫn**

Sử dụng biến danh sách NT để lưu các thừa số nguyên tố của n. Chương trình sẽ thiết lập danh sách NT khi n > 1. Kết quả sẽ như sau:

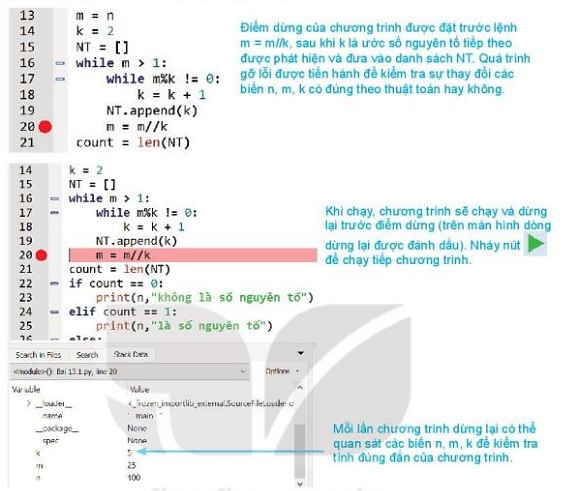
- Nếu n = 1 thì danh sách NT rỗng.

- Nếu n > 1 thì danh sách NT không rỗng. Độ dài danh sách len(NT) sẽ bằng 1 khi và chỉ khi n là số nguyên tố.

Nếu len(NT) > 1 thì chương trình sẽ in ra khai triển n thành tích các thừa số nguyên tố, khai triển này có dạng n=p1×p2×...×pkn=p1×p2×...×pk



- Chạy chương trình với công cụ gỡ lỗi của phần mềm lập trình. Thiết lập một điểm dừng tại dòng 20:



- Thiết lập bảng theo dõi các giá trị trung gian k, m, n, NT. Giả sử giá trị ban đầu của n = 100.



**Nhiệm vụ 2:**Viết chương trình nhập từ bàn phím ba số thực a, b, c và tìm nghiệm cảu phương trình bậc 2ax2+bx+c=0ax2+bx+c=0. Cần xét đầy đủ các trường hợp xảy ra.

**Hướng dẫn**

- Nếu a = b = c = 0 phương trình có vô số nghiệm.

- Nếu a = b = c; c ≠ 0, phương trình vô nghiệm.

- Nếu a = 0; b ≠ 0 phương trình bậc nhất và có nghiệm duy nhất.

- Nếu a ≠ 0, giải phương trình bậc hai. Nghiệm sẽ phụ thuộc vào giá trị delta = b2 – 4ac.

Phương trình vô nghiệm, có một nghiệm kép hoặc hai nghiệm phân biệt phụ thuộc vào giá trị delta.

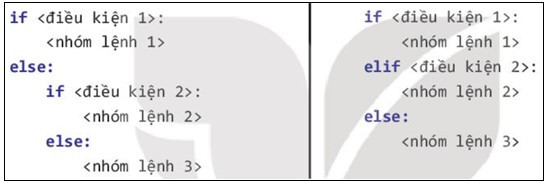
- Các hàm được thiết kế qua các hàm:

+ NhapDL(): nhập 3 số a, b, c từ bàn phím

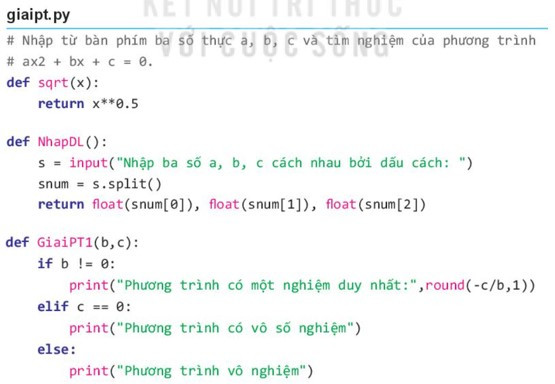
+ GiaiPT1(b, c): hàm giải phương trình bậc nhất bx+c=0bx+c=0

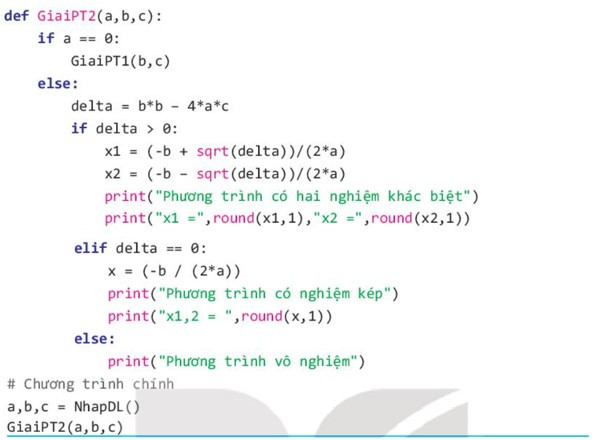
+ GiaiPT2(a, b, c): hàm giải phương trình bậc hai ax2+bx+c=0ax2+bx+c=0

- Sử dụng cấu trúc câu lệnh rẽ nhánh if…else…. Trong đó mô hình bên trái viết gọn hơn mô hình bên phải.



- Chương trình đầy đủ:





**Bài 32: Ôn tập lập trình Python**

**Nhiệm vụ 1:**Viết chương trình nhập họ tên đầy đủ từ bàn phím, ví dụ “Nguyễn Thị Mai Hương”, sau đó tách riêng phần họ, tên, đệm ra màn hình.

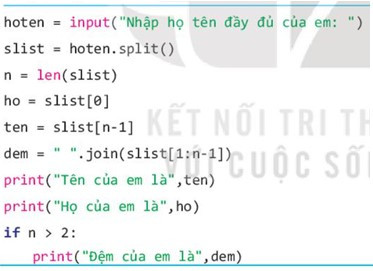
**Hướng dẫn**

**-**Sử dụng câu lệnh join() để lấy phần nằm giữa tên và đệm như sau:

dem = “ ”.join(slist[1:n-1])

Trong đó slist là danh sách được tách ra từ xâu ban đầu, n là độ dài của xâu slist.

Nhập, chạy thử và kiểm tra chương trình sau:



**Nhiệm vụ 2:**Trọng lượng của em trên các hành tinh khác.

Chương trình yêu cầu nhập trọng lượng của em (tính theo đơn vị N) trên Trái Đất và tính trọng lượng của em trên một hành tinh khác (ví dụ Mặt Trăng, Hỏa tinh, Kim tinh, Thổ tinh, Mộc tinh, Mặt trời).

**Hướng dẫn**

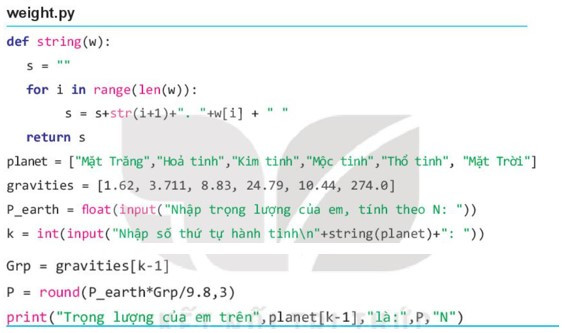
Trọng lượng đo lực hút của Trái Đất hay vật thể tính theo công thức P = m × g

Trong đó giá trị g trên các hành tinh khác nhau, ví dụ trên Trái đất là g = 9.8 m/s2

Gọi P là trọng lượng cần tìm, khi đó ta có công thức sau:

m = P0/9.8=P/g, vậy suy ra P = P0×9.8

Nhập phương trình sau và kiểm tra tính đúng đắn của phương trình:



**Nhiệm vụ 3:**Kiểm tra tính hợp lệ của ba tham sốngày, tháng, năm.

Chương trình sẽ yêu cầu nhập ba số tự nhiên ngày, tháng, năm từ bàn phím theo khuôn dạng, ví dụ 08-02-2021. Chương trình sẽ thông báo bộ dữ liệu đã nhập là hợp lệ hay không hợp lệ.

**Hướng dẫn**

- Bộ dữ liệu chính cần nhập sẽ đặt tên là day, month, year. Nhiệm vụ của bài toán là nhập bộ dữ liệu này và kiểm tra tính hợp lệ theo các yêu cầu về lịch của ngày, tháng, năm.

Chú ý kiểm tra năm year có là năm nhuận không. Nếu có cần cập nhật tháng 2.

- Chương trình:

