Học lập trình Python

Giới thiệu

Lập trình là gì?

Một cách dễ hiểu, lập trình là cách giải các bài toán bằng cách viết chương trình trên máy tính.

Khi chưa có máy tính, con người vẫn giải các bài toán gặp trong cuộc sống. Các bài toán đơn giản có thể tính nhẩm trong đầu, các bài toán phức tạp hơn cần đến giấy và bút.

Ở cấp 1, chúng ta đã học cách giải các bài toán, đơn giản nhất là bài toán có một phép tính, rồi đến các bài toán có nhiều phép tính.

Ví dụ 1:
An và Bình là 2 anh em. Tuổi của An là 10, Bình kém An 2 tuổi. Hỏi Bình bao nhiêu tuổi.

Lời giải (trên giấy)

Tuổi của Bình là:
10 - 2 = 8 (tuổi)

Đáp số : 8 tuổi

Ví dụ 2:
Một người đi làm bằng xe máy, quãng đường từ nhà đến nơi làm việc là 10 km. Trung bình cứ 50km thì xe máy tiêu thụ hết 1 lít xăng. Giá xăng là 21000 đồng/lít. Một tháng người đó phải đi làm 25 ngày. Hỏi tiền xăng cho việc đi lại trong 1 tháng của người đó là bao nhiêu?

Lời giải (trên giấy)
Quãng đường đi lại trong 1 ngày:
2 x 10 = 20 (km)

Quãng đường đi lại trong 1 tháng:
20 x 25 = 500 (km)

Số lít xăng tiêu thụ trong 1 tháng:
500/50 = 10 (lít)

Tiền xăng tiêu thụ trong 1 tháng:
10 x 21000 = 210000 (đồng)

Đáp số : 210000 đồng

Kể từ khi có máy tính, con người đã sử dụng máy trong việc giải các bài toán. Việc giải toán trên máy tính nhanh hơn và chính xác hơn so với việc giải bằng tay.

Ngôn ngữ lập trình Python

Có nhiều ngôn ngữ lập trình có thể sử dụng để giải các bài toán trên máy tính. Hiện nay ngôn ngữ lập trình Python là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất trên thế giới. Ưu

điểm của Python là cho phép viết chương trình ngắn gọn và dễ đọc, gần với ngôn ngữ thể hiện của con người.

Bạn hãy xem lời giải của 2 bài toán trên bằng ngôn ngữ Python :

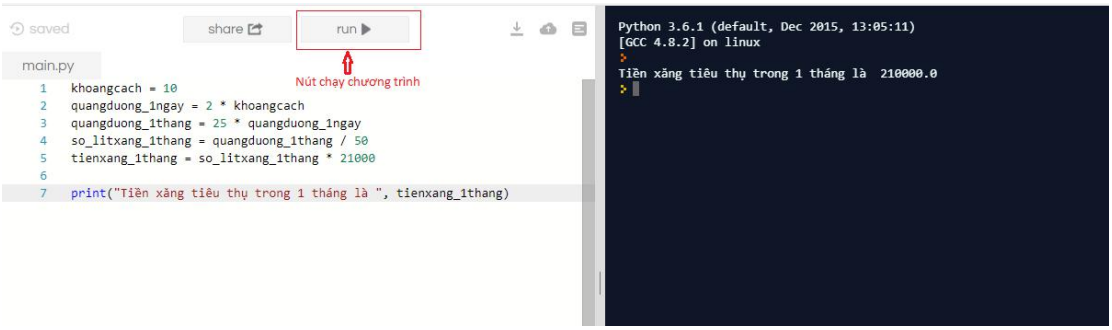
Ví dụ 1:

```
tuoiAn = 10
tuoiBinh = tuoiAn - 2
print("Tuổi của Bình là : ", tuoiBinh)
```

Ví dụ 2:

```
khoangcach = 10
quangduong_1ngay = 2 * khoangcach
quangduong_1thang = 25 * quangduong_1ngay
so_litxang_1thang = quangduong_1thang / 50
tienxang_1thang = so_litxang_1thang * 21000

print("Tiền xăng tiêu thụ trong 1 tháng là ", tiengxang_1thang)
```

Bạn hãy copy đoạn chương trình trên vào cửa sổ soạn thảo của chương trình chạy Python online tại website https://repl.it/site/languages/python3 , sau đó chạy chương trình và quan sát cửa sổ kết quả.



Kết quả chạy ví dụ bằng chương trình Python Online

Cài đặt Python

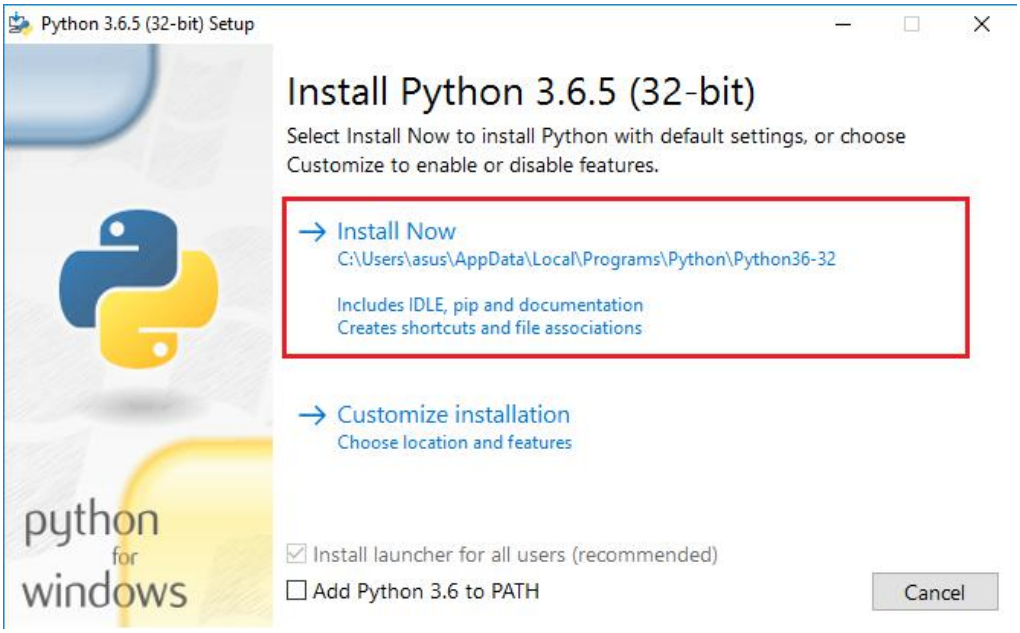
Python được sử dụng ở 2 phiên bản : Python 2 và Python 3, hiện nay phần lớn các chương trình sử dụng Python 3 để phát triển. Các chương trình trong tài liệu này được viết để chạy với Python 3.

Để cài đặt Python 3, bạn hãy tải file cài đặt tại trang chủ của Python:

https://www.python.org/downloads/

Tải Python từ trang chủ

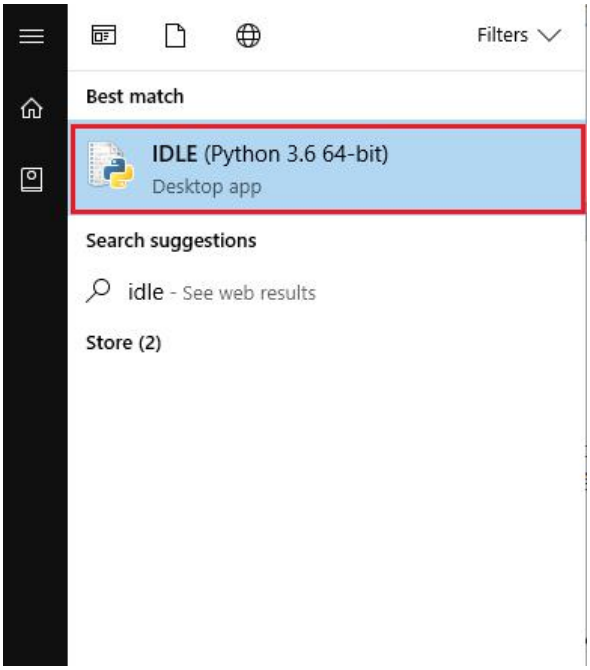
Sau khi tải file cài đặt về, bạn hãy chạy file này và chọn các thiết lập cài đặt mặc định, cho đến khi chương trình cài đặt hoàn thành:



Cài đặt Python ở chế độ mặc định

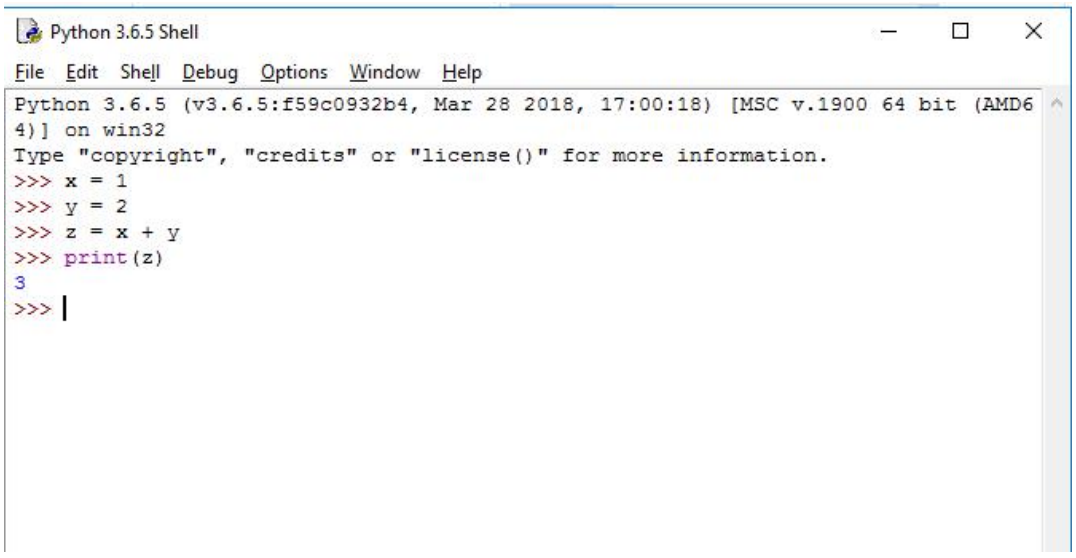
Chạy chương trình Python

Sau khi đã cài đặt Python xong, bạn hãy tìm chương trình IDLE (Integrated DeveLopment Environment) của Python từ Start Menu và khởi động chương trình này :



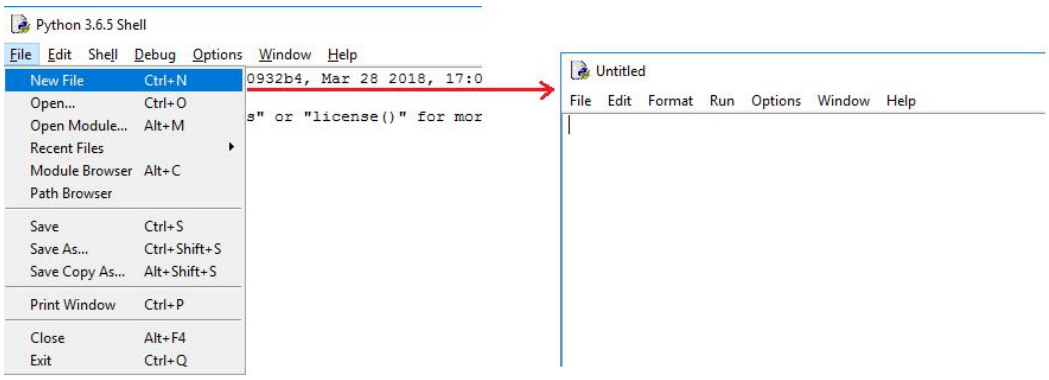
Khởi động chương trình soạn thảo Python IDLE

Sau khi khởi động xong, IDLE sẽ cho phép bạn chạy từng dòng lệnh Python bằng cách gõ trực tiếp vào cửa sổ chương trình:



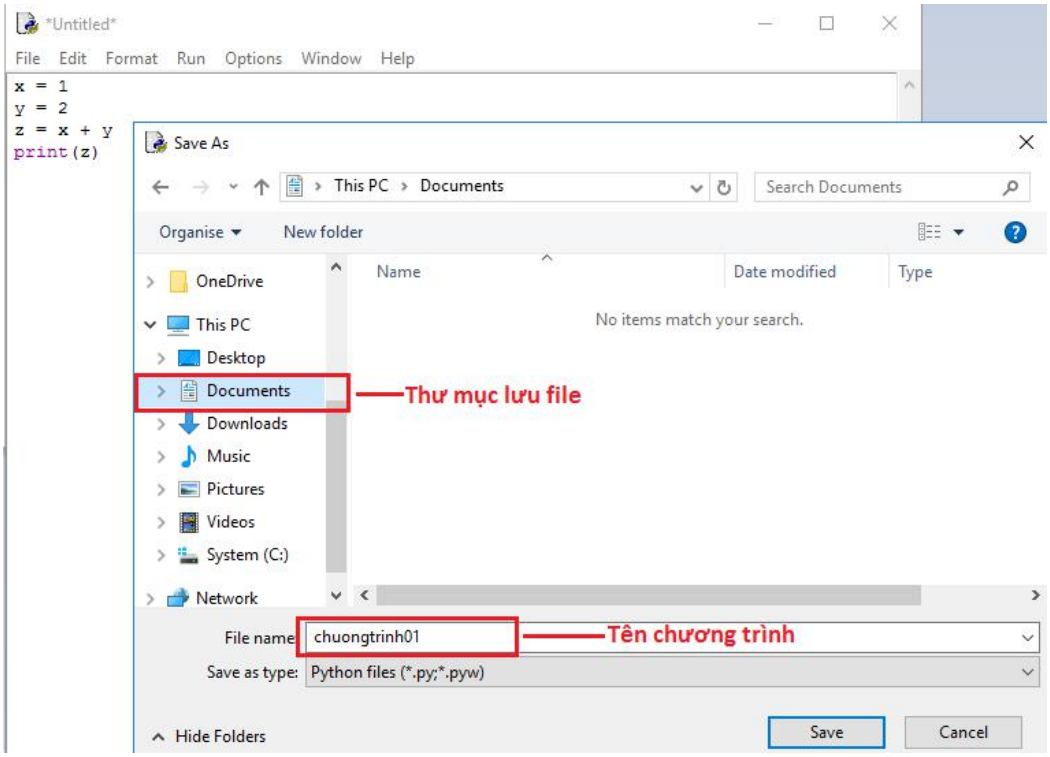
Chạy Python ở chế độ từng lệnh trong cửa sổ IDLE

Với các chương trình lớn, việc chạy từng lệnh sẽ không phù hợp, bạn có thể tạo ra một file với tên mở rộng py để lưu chương trình của mình. Cách thực hiện như sau : từ menu **File** của IDLE chọn **New File,** một cửa sổ mới sẽ được tạo ra cho phép bạn soạn thảo chương trình:



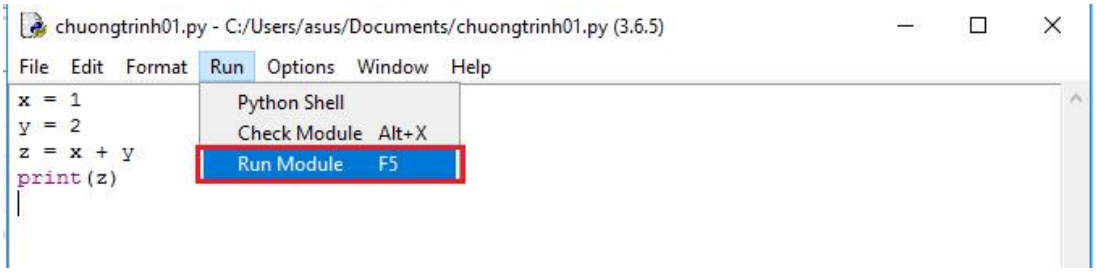
Tạo file mới để soạn thảo

Sau khi soạn thảo xong ở cửa sổ mới, bạn cần lưu lại file này : Từ menu **File** hãy chọn **Save**, cửa sổ lưu file sẽ hiện ra yêu cầu bạn đặt tên cho file và chọn thư mục để chứa file. Bạn hãy nhớ địa chỉ lưu trữ này để tìm đến trong các lần mở file tiếp theo:



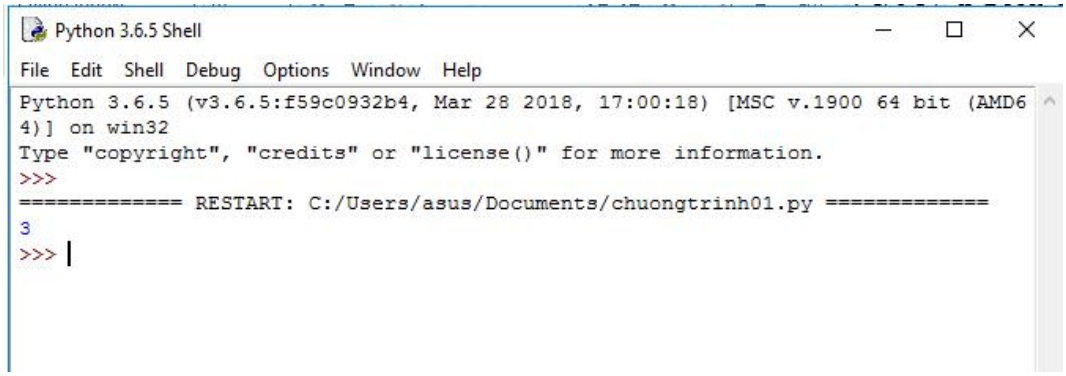
Lưu lại chương trình đã soạn thảo xong

Sau khi đã lưu chương trình, bạn có thể thực hiện chương trình bằng cách từ menu **Run** chọn **Run Module,** hoặc nhấn F5 :



Chạy chương trình đã lưu.

Kết quả chương trình sẽ được hiển thị trong cửa sổ dòng lệnh IDLE :



Kết quả chương trình được hiển thị trong cửa sổ IDLE

Chú thích trong chương trình:

Khi viết chương trình, bạn có thể thêm các chú thích để giải thích ý nghĩa của các dòng lệnh. Trong Python, các chú thích được đặt sau dấu **#**, tức toàn bộ phần sau dấu # đến hết cuối dòng sẽ được Python bỏ qua khi chạy chương trình. Ví dụ :

```
# Chương trình tính tổng 2 số và in ra màn hình

x = 1
y = 2
z = x + y

print(z)        # In kết quả ra màn hình
```

Phần I : Lập trình Python cơ bản

Chương 1 : Biến số

Biến số là gì?

Biến số là khái niệm cơ bản nhất trong các chương trình. Chúng được dùng để lưu giá trị trung gian trong quá trình tính toán.

```
tuoiAn = 10
tuoiBinh = tuoiAn - 2
```

Ở chương trình trên (chương trình trong phần giới thiệu), `tuoiAn, tuoiBinh` là các biến số.

Cú pháp để khai báo một biến số:

```
bienSo = <giá trị>
```

Trong đó `bienSo` là tên của biến số cần khai báo, `<giá trị>` là giá trị ban đầu chúng ta muốn đặt cho biến số. Giá trị này có thể là một hằng số hoặc một biểu thức của các biến số đã được khai báo.

Sau khi được khai báo, biến số sẽ được lưu trong bộ nhớ máy tính. Giá trị của biến số có thể thay đổi trong chương trình:

```
x = 1
x = 2
x = x + 1
```

Ở chương trình trên:
Dòng đầu tiên khai báo biến x và đặt giá trị cho biến bằng 1
Dòng thứ 2 thay đổi giá trị của x thành 2
Dòng thứ 3 tăng x thêm 1 đơn vị

Để xóa một biến số khỏi bộ nhớ, chúng ta dùng lệnh:

```
del bienSo
```

Chương 2 : Kiểu dữ liệu

Mỗi biến số sau khi được khai báo sẽ chứa giá trị nhất định, gọi là dữ liệu. Giá trị dữ liệu này sẽ thuộc vào một trong các kiểu dữ liệu mà Python hỗ trợ.

Khi giải toán trên giấy, chúng ta thường chỉ quan tâm tới dữ liệu kiểu số. Trên máy tính, ngoài kiểu dữ liệu số, còn có dữ liệu văn bản (String). Loại dữ liệu này dùng để lưu các biến số như tên, địa chỉ, số điện thoại …

Ngoài ra trong Python, còn có các kiểu dữ liệu mà chúng ta sẽ biết đến ở phần sau:

- Kiểu danh sách (List)
- Kiểu bộ nhóm (Tuple)
- Kiểu tập hợp (Set)
- Kiểu từ điển (Dictionary)

- Kiểu tệp tin (File)

Chúng ta tạm quan tâm đến 2 kiểu dữ liệu chính là kiểu số và kiểu văn bản.

Kiểu dữ liệu số

Đây là kiểu dữ liệu sử dụng nhiều nhất. Chúng ta đã biết kiểu dữ liệu này qua các ví dụ ở phần giới thiệu.

Trong Python có 2 loại số : số nguyên và số thập phân.

Kiểu dữ liệu số nguyên

```
x = 1
y = 2
z = x + y
```

Kiểu dữ liệu số thập phân

```
x = 1.2
y = 0.0001
z = 1.7e6
```

Cách viết 1.7e6 có nghĩa là 1.7×10^6

Kiểu dữ liệu văn bản (String)

Dữ liệu văn bản được đặt trong 2 dấu nháy đơn (') hoặc 2 dấu nháy kép (")

```
hoten = 'Nguyễn Văn An'
diachi = "Hà Nội"
```

Chúng ta sẽ tìm hiểu chi tiết hơn về kiểu dữ liệu này trong chương *"Dữ liệu kiểu văn bản"* ở phần sau

Chương 3 : Lấy dữ liệu vào từ bàn phím và in dữ liệu ra màn hình

Lấy dữ liệu vào từ bàn phím

Giá trị của các biến số có thể được khai báo trực tiếp trong chương trình, tuy nhiên, trong nhiều trường hợp, chúng ta muốn sử dụng lại chương trình đã viết với các bộ dữ liệu vào khác, trong trường hợp đó cần cho phép người sử dụng nhập dữ liệu vào cho các biến từ bàn phím.

Trong Python, việc này được thực hiện qua lệnh `input` :

```
giatri = input('Nhập giá trị cho biến : ')
```

Lệnh trên sẽ đọc các kí tự nhập từ bàn phím cho đến khi gặp kí tự Enter, giá trị nhập vào được lưu vào biến khai báo ở đầu dòng lệnh, giá trị này có kiểu dữ liệu là văn bản (String).

Ví dụ 1:

Viết chương trình yêu cầu người sử dụng giới thiệu tên, sau đó in ra lời chào theo tên đã nhập vào.

```
ten = input('Mời bạn cho biết tên của bạn : ')
print('Xin chào ', ten)
```

Trong trường hợp muốn lấy giá trị vào ở dạng số, chúng ta sử dụng các lệnh sau để chuyển đổi từ dữ liệu văn bản sang dữ liệu số:

- `int(text)` : chuyển giá trị `text` có kiểu văn bản thành số nguyên
- `float(text)` : chuyển giá trị `text` có kiểu văn bản thành số thập phân

Ví dụ 2:

Viết chương trình nhập vào từ bàn phím 2 số và in ra tổng của 2 số đó.

```
a = input('Số thứ nhất : ')
a = float(a)

b = input('Số thứ hai : ')
b = float(b)

print('Tổng của 2 số là : ', a+b)
```

In dữ liệu ra màn hình

Sau khi tính hoàn thành một bài toán, chúng ta cần in kết quả ra màn hình. Để thực hiện việc này chúng ta dùng lệnh `print` theo cú pháp:

```
print(<danh sách giá trị ngăn cách nhau bởi dấu phảy>)
```

Ví dụ 3:

```
x = 1
y = 2
z = x + y
print(x, '+', y, '=', z)
```

Ở dòng cuối trong chương trình trên, danh sách các giá trị được in ra nằm trong lệnh `print`, theo thứ tự là:

- `x` → `1`
- `'+'` → `+`
- `y` → `2`
- `'='` → `=`
- `z` → `3`

Như vậy, thông tin được in ra trên màn hình là:

```
1 + 2 = 3
```

Chương 4 : Các phép tính

Python hỗ trợ các phép tính toán học thông dụng. Các phép tính thường được sử dụng là:

- Phép cộng (+)
- Phép trừ (-)
- Phép nhân (*)
- Phép chia (/)
- Phép chia số nguyên (//)
- Phép lấy phần dư (%)
- Phép lũy thừa (**)
- Phép làm tròn số (round)
- Phép lấy giá trị tuyệt đối (abs)

Các phép tính `+, -, *, /` tương tự như các phép tính số học chúng ta đã biết.

Ví dụ 1:

```
print(1 + 2)
print(3 - 2)
print(3 * 4)
print(1/2)
```

Giá trị được in ra màn hình khi chạy chương trình trên :

```
3
1
12
0.5
```

Phép chia số nguyên và phép lấy phần dư

Phép chia số nguyên (//) và phép lấy phần dư (%) được thực hiện để chia các số nguyên.

Ví dụ 2:

```
print(10 // 3)
print(10 % 3)
```

Giá trị in ra màn hình khi chạy chương trình trên:

```
3
1
```

Điều này tương đương với : 10 chia 3 được 3 dư 1

Phép lũy thừa

Trong Python, phép lũy thừa được thể hiện bằng 2 dấu * viết liền nhau.

Ví dụ 3:

```
print(2 ** 10)
```

Giá trị in ra màn hình là 1024, có nghĩa 2^{10} = 1024

Phép tính dạng rút gọn

Python cho phép sử dụng các phép tính dạng rút gọn:

- +=
- -=
- *=
- /=
- //=
- %=
- **=

Ý nghĩa của phép tính rút gón như sau:

- `a += b` tương đương với `a = a + b`
- `a -= b` tương đương với `a = a - b`
- `...`

Phép làm tròn số thập phân

Lệnh `round` dùng để làm tròn số thập phân

Ví dụ 4:

```
print(round(1.2))
```

Giá trị in ra là 1.0

Hàm `round` còn có thể dùng để làm tròn theo một số lượng chữ số sau dấu phảy

Ví dụ 5:

```
print(round(4/3, 2))
```

Giá trị in ra là 1.33

Phép lấy giá trị tuyệt đối

Hàm `abs` dùng để lấy giá trị tuyệt đối của một số

Ví dụ 6:

```
print(abs(2))
print(abs(-1))
```

Giá trị in ra màn hình:

```
2
1
```

Các hàm toán học

Ngoài các phép tính ở trên, trong Python còn có thể sử dụng các hàm toán học, bằng cách khai báo sử dụng thư viện `math` có sẵn của Python.

Các hàm toán học thông dụng là:

- Các hàm lượng giác : `sin, cos, tan, asin, acos, atan`
- Hàm căn bậc 2 : `sqrt`
- Hàm phần nguyên : `floor`, hàm phần nguyên trên : `ceil`
- Hàm logarith : `log10` (cơ số 10), `log` (cơ số e)

Ví dụ 7:

```
import math
print(math.sin(math.pi/2))
print(math.sqrt(4))
```

Để sử dụng được các hàm toán học, trước hết chúng ta phải khai báo thư viện `math` qua lệnh:

```
import math
```

Sau đó, các hàm toán học sẽ được viết theo cấu trúc `math.<tên hàm>`, như ở ví dụ trên.

Để quen với các phép tính, chúng ta hãy tìm hiểu một số ví dụ tổng hợp sau.

Ví dụ 8:

Viết chương trình in ra giá trị của các biểu thức :

- 1999 * 2001
- 2000 * 2000 - 1999 * 2001

Lời giải:

```
print(1999 * 2001)
print(2000 * 2000 - 1999 * 2001)
```

Ví dụ 9:

Tìm 2 số biết tổng và hiệu của chúng. Nhập vào từ bàn phím các giá trị S : tổng 2 số, D : hiệu 2 số , in ra màn hình giá trị 2 số đó.

Lời giải:

```
S = input('S = ')
S = float(S)

D = input('D = ')
D = float(D)

a = (S - D)/2
```

```
b = (S + D)/2

print('a = ', a)
print('b = ', b)
```

Ví dụ 10:

Trên mặt phẳng cho 3 điểm có tọa độ (0,0) , (1,1) và (2,3). Tính diện tích tam giác tạo thành từ 3 điểm trên.

Lời giải:

Để tích diện tích một tam giác tạo thành từ 3 điểm, ta sử dụng công thức diện tích tam giác do 2 vector tạo thành (X1, Y1) , (X2, Y2) là: S = 0.5 * |X1*Y2 - X2*Y1|

Như vậy để tính diện tích tam giác tạo thành từ 3 điểm, cần lấy một điểm trong 3 điểm làm mốc, tính tọa độ 2 vector xuất phát từ điểm đó đến 2 điểm còn lại, sau đó sử dụng công thức ở trên.

Chương trình :

```
x1, y1 = 1, 1
x2, y2 = 2, 3
x3, y3 = 0, 0

X1 = x1 - x3
Y1 = y1 - y3

X2 = x2 - x3
Y2 = y2 - y3

S = abs(0.5*(X1*Y2 - X2*Y1))

print('Diện tích tam giác : ', round(S, 6))
```

Chương 5 : Các lệnh điều khiển

Khi giải các bài toán trên giấy, thứ tự thực hiện các phép tính là từ trên xuống dưới. Tuy nhiên, với các bài toán trên máy tính, có hiện tượng "rẽ nhánh", tức là tùy vào kết quả của phép tính trước mà một số phép tính sau có thể được thực hiện hay không.

Để thực hiện việc này, Python sử dụng lệnh `if` theo các cấu trúc:

- Cấu trúc 1:

```
if <điều kiện> :
    <Lệnh>
```

Cấu trúc trên có nghĩa : Nếu điều kiện đúng thì thực hiện lệnh, không thì không thực hiện lệnh.

- Cấu trúc 2:

```
if <điều kiện> :
    <Lệnh 1>
else:
    <Lệnh 2>
```

Cấu trúc trên có nghĩa : Nếu điều kiện đúng thì thực hiện lệnh 1, không thì thực hiện lệnh 2.

- Cấu trúc 3:

```
if <điều kiện 1> :
    <Lệnh 1>
elif <điều kiện 2>  :
    <Lệnh 2>
elif <điều kiện 3>  :
    <Lệnh 3>
else:
    <Lệnh 4>
```

Cấu trúc trên có nghĩa : Nếu điều kiện 1 đúng thì thực hiện lệnh 1, không thì kiểm tra điều kiện 2, nếu điều kiện 2 đúng thì thực hiện lệnh 2, nếu không tiếp tục kiểm tra các điều kiện tiếp theo.

Điều kiện trong câu lệnh `if` là điều kiện logic, chỉ có giá trị đúng (`True`) hoặc sai (`False`). Để tạo ra các điều kiện này, chúng ta sử dụng các phép so sánh và các phép tính logic:

- Các phép so sánh : lớn hơn (>), nhỏ hơn (<), lớn hơn hoặc bằng (>=), nhỏ hơn hoặc bằng (<=), bằng nhau (==), khác nhau (!=)
- Phép phủ định `not` : "`not dieukien`" có giá trị `True` nếu `dieukien` là `False` và ngược lại
- Phép `and` : "`dieukien1 and dieukien2`" có giá trị `True` nếu cả 2 điều kiện thành phần có giá trị `True`
- Phép `or` : "`dieukien1 or dieukien2`" có giá trị `True` nếu ít nhất 1 trong 2 điều kiện thành phần có giá trị `True`

Một số lưu ý:

- Cuối các dòng của lệnh `if` hoặc `else` là dấu 2 chấm (`:`). Python sử dụng dấu hai chấm để bắt đầu cho một khối lệnh con, bạn sẽ quen dần với điều này.
- Các dòng lệnh bên dưới `if` hoặc `else` cần được viết lùi vào một số khoảng trắng (thường là 4 khoảng trắng, hoặc 1 phím tab) so với dòng chứa lệnh `if/else`, nếu có nhiều hơn một dòng lệnh thì chúng phải được viết thẳng hàng.

Ví dụ 1:

So sánh 2 số :

- A = 1999 *2001
- B = 2000 * 2000

Lời giải:

```
A = 1999 * 2001
B = 2000 * 2000

if A > B:
   print('A > B')

if A < B:
   print('A < B')
```

```
if A == B:
   print('A = B')
```

Ví dụ 2:

Nhập vào từ bàn phím 2 số và in ra giá trị nhỏ nhất của 2 số đó

Lời giải:

```
a = input('Số thứ nhất : ')
a = float(a)

b = input('Số thứ hai : ')
b = float(b)

if a < b:
   print(a)

else:
   print(b)
```

Ví dụ 3:

Chỉ số BMI (Body mass index) dùng để đánh giá thân hình của một người. Chỉ số này được đo bằng tỉ số giữa cân nặng (tính theo kg) và bình phương của chiều cao (tính theo mét). Dựa vào chỉ số BMI, người ta có thể đánh giá được thân hình của một người:

- BMI nhỏ hơn 15 : thân hình quá gầy
- BMI từ 15 đến 16 : thân hình gầy
- BMI từ 16 đến 18.5 : thân hình hơi gầy
- BMI từ 18.5 đến 25 : thân hình bình thường
- BMI từ 25 đến 30 : thân hình hơi béo
- BMI từ 30 đến 35: thân hình béo
- BMI trên 35: thân hình quá béo

Nhập vào giá trị chiều cao và cân nặng của một người từ bàn phím và cho biết tình trạng thân hình của người đó.

Lời giải:

```
height = input('Chiều cao (mét) : ')
height = float(height)

mass = input('Cân nặng (kg) : ')
mass = float(mass)

bmi = mass / (height * height)

if bmi < 15:
    print('Thân hình quá gầy')

elif bmi < 16:
    print('Thân hình gầy')

elif bmi < 18.5:
    print('Thân hình hơi gầy')
```

```
elif bmi < 25:
    print('Thân hình bình thường')

elif bmi < 30:
    print('Thân hình hơi béo')

elif bmi < 35:
    print('Thân hình béo')

else:
    print('Thân hình quá béo')
```

Chương 6 : Các lệnh lặp

Trong lập trình, nhiều lúc chúng ta muốn thực hiện một công việc nào đó lặp lại nhiều lần. Các lệnh lặp sẽ giúp chúng ta thực hiện điều đó.

Lệnh lặp for

Lệnh `for` được dùng để thực hiện một vòng lặp với số lần lặp biết trước. Cấu trúc lệnh này như sau:

```
for bienLap in tapGiaTri:
    <khoi_lenh>
```

Cấu trúc này có nghĩa : với mỗi giá trị `bienLap` của `tapGiaTri`, thực hiện khối lệnh `<khoi_lenh>`

Các thành phần của vòng lặp for:

- Tập giá trị : chứa tập giá trị cần duyệt qua
- Biến lặp : biến được gán giá trị từ mỗi phần tử trong tập giá trị lặp
- Khối lênh lặp : khối lệnh cần thực hiện lặp lại

Các cách biểu diễn tập giá trị lặp:

- Liệt kê các thành phần. Ví dụ [1, 2, 3, 4, 5]. Đây thực chất là dữ liệu kiểu danh sách mà chúng ta sẽ biết ở các phần sau.

- Khoảng "`range(end)`" : bao gồm các số tự nhiên từ 0 đến trước `end`. Lưu ý, với Python (và nhiều ngôn ngữ lập trình), khoảng này không chứa giá trị `end`, tức giá trị cuối của khoảng là `end-1`

- Khoảng "`range(start, end)`" : bao gồm các số nguyên bắt đầu từ `start` đến trước `end`.

- Khoảng "`range(start, end, increment)`" : bao gồm các số nguyên từ `start` đến trước `end` và tăng đều theo khoảng cách `increment`

Một số ví dụ về tập giá trị lặp:

- `range(5)` → 0, 1, 2, 3, 4
- `range(1, 5)` → 1, 2, 3, 4
- `range(0, 10, 2)` → 0, 2, 4, 6, 8

Ví dụ 1:

Tính tổng các số nguyên từ 1 đến 100

Lời giải:

```
S = 0
for i in range(1, 101):
    S += i

print('S = ', S)
```

Lưu ý: Khoảng `range(start, end)` bao gồm các số từ `start` đến `end-1`, do đó để thể hiện các số từ 1 đến 100 chúng ta phải dùng `range(1, 101)`

Ví dụ 2:

Tính tổng
`S = 1/(1*2) + 1/(2*3) + 1/(3*4) + ... + 1/(999*1000)`

Lời giải:

```
S = 0

for i in range(1, 1000):
    S += 1/(i*(i+1))

print('S = ', round(S,6))
```

Bạn hãy kiểm tra xem kết quả có phải bằng `1-1/1000` không.

Ví dụ 3:

Viết chương trình in ra bảng cửu chương.

Lời giải:

```
for i in range(2, 10):
    for j in range(2, 10):
        print(i, '*', j, '=', i*j)
    print()
```

Lệnh lặp while

Lệnh lặp `while` dùng để thực hiện một vòng lặp với số lần lặp không biết trước. Cấu trúc lệnh này như sau:

```
while dieukien:
    <khoi_lenh>
```

Cấu trúc trên có nghĩa : Chừng nào điều kiện `dieukien` còn đúng thì thực hiện khối lệnh `khoi_lenh`.

Ví dụ 4:

Kiểm tra số 8477216359 có phải số nguyên tố không.

Lời giải:

Để kiểm tra một số có phải số nguyên tố hay không, chúng ta cần kiểm tra số đó có chia hết cho số nguyên tố nào trong phạm vi từ 2 đến căn bậc 2 của số đó hay không. Tuy nhiên do không có danh sách các số nguyên tố, chúng ta có thể dùng cách kiểm tra xem số đó có chia hết cho số tự nhiên nào từ 2 đến căn bậc 2 của số đó không.

Chương trình :

```
x = 8477216359

i = 2
while i * i <= x:
    if x % i == 0:
        j = x // i
        print(x, '=', i , '*', j)
        exit()
    i += 1

print(x, ' là số nguyên tố')
```

Ví dụ 5:

Chương trình đoán số tự nhiên.

Bạn hãy nghĩ trong đầu một số trong phạm vi từ 0 đến 1000. Máy tính sẽ đưa ra các câu hỏi, với mỗi câu bạn chỉ cần trả lời câu đó là đúng hay sai. Sau không quá 10 câu hỏi máy tính sẽ đưa ra số bạn đang nghĩ là gì.

Lời giải:

```
low = 0
high = 1000

print('Bạn hãy nghĩ một số trong phạm vi từ 0 đến 1000, sau đó trả lời các
câu hỏi sau.')

while low + 1 != high:
    mid = (low + high) // 2
    a = input('Số đó lớn hơn ' + str(mid)  + ' ? (Y/N) : ')

    if a.upper() == 'Y':
        low = mid
    else:
        high = mid

print('Số bạn nghĩ là ', high)
```

Giải thích:

- Chương trình trên dựa trên thuật toán tìm kiếm nhị phân. Máy tính đã biết số bạn đang nghĩ nằm trong khoảng từ 0 đến 1000, do đó đặt ra 2 giá trị `low = 0` và `high = 1000` để giới hạn khoảng giá trị của số bạn nghĩ.

- Ở mỗi bước, máy tính tính giá trị chính giữa của khoảng giá trị `mid = (low+high)//2`, sau đó hỏi bạn số bạn nghĩ có lớn hơn điểm giữa đó không. Dựa vào câu trả lời của bạn, máy tính sẽ thu hẹp khoảng giá trị chứa số bạn nghĩ.

- Sau mỗi câu hỏi, khoảng giá trị chứa số bạn nghĩ sẽ bị thu hẹp đi một nửa. Vì ban đầu khoảng này có độ rộng là 1000, nhỏ hơn 2^{10} (1024), nên sau không quá 10 câu hỏi, độ rộng khoảng này sẽ thu về 1 và máy tính sẽ xác định được số bạn đang nghĩ là bao nhiêu. Điều kiện "`low + 1 != high`" là điều kiện giúp máy tính kiểm tra khoảng giá trị đã thu về 1 điểm hay chưa.

Chương 7 : Hàm giá trị

Hàm (function) là một đoạn chương trình được tổ chức thành một khối độc lập để có thể được sử dụng lại nhiều lần từ các vị trí khác nhau trong chương trình.

Cách khai báo một hàm như sau:

```
def tenham(<danh_sach_bien>):
    <noi_dung_ham>
    return <ket_qua>.
```

Các thành phần của hàm:

- Khai báo hàm : bắt đầu bằng từ khóa `def` của Python, tiếp theo là tên hàm, tiếp theo là danh sách các biến nằm trong 2 dấu (), các biến ngăn cách nhau bởi dấu phảy.

- Nội dung hàm : bao gồm các lệnh thực hiện của hàm. Các dòng lệnh này phải được viết lùi vào so với dòng khai báo hàm một số (thường là 4) khoảng trắng.

- Kết quả trả về của hàm : kết quả được trả về với từ khóa `return` đi kèm các giá trị trả về, cũng được ngăn cách bởi dấu phảy.

Trong chương trình, việc gọi hàm được thực hiện nhờ cú pháp sau:

```
<danh_sach_bien_ketqua> = tenham(<danh_sach_bien_dau_vao>)
```

Trong trường hợp không quan tâm đến kết quả trả về của hàm, chúng ta dùng cú pháp:

```
tenham(<danh_sach_bien_dau_vao>)
```

Giá trị mặc định của biến đầu vào trong hàm:

Trong một số trường hợp, chúng ta muốn một/một vài biến đầu vào của hàm có giá trị mặc định nếu chúng không được truyền vào khi gọi hàm, khi đó chúng ta khai báo hàm theo cấu trúc:

```
tenham(tenbien=giatri_macdinh)
```

Ví dụ 1:

Viết hàm tính bình phương của một số. Sử dụng hàm này để in ra bình phương của các số tự nhiên từ 1 đến 10.

Lời giải:

```
def square(x):
   return x*x
```

```
for i in range(1,11):
   print(square(i))
```

Ví dụ 2:

Viết hàm tính diện tích và chu vi của một hình chữ nhật khi biết độ dài 2 cạnh. Sử dụng hàm để tính diện tích, chu vi của hình chữ nhật có kích thước 5 x 4

Lời giải:

```
def calcAreaAndPerimeter(width, height):
    S = width * height
    P = 2*(width + height)
    return S, P

S, P = calcAreaAndPerimeter(5, 4)
print('S = ', S)
print('P = ', P)
```

Ví dụ 3:

Viết chương trình tìm giá trị nhỏ nhất của hàm số $2x^2 - x + 1$ trên đoạn [0,1]

Lời giải:

Chúng ta chia nhỏ đoạn [0,1] thành N vị trí, tính giá trị của hàm số tại các vị trí đó, sau đó tìm ra giá trị nhỏ nhất trong các vị trí này.

Chương trình :

```
def f(x):
    return 2*x*x - x + 1

a = 0
b = 1
xmin = a
fmin = f(a)
N = 1000

for i in range(N):
    x = a + (b-a) * (i/N)
    fx = f(x)
    if fx < fmin:
        fmin = fx
        xmin = x

print('fmin=', fmin, ' , xmin=', xmin)
```

Ví dụ 4:

Viết chương trình tìm nghiệm gần đúng của phương trình $x^3 + x - 1 = 0$

Lời giải:

Chúng ta nhận thấy hàm số f(x) = x^3+ x - 1 thỏa mãn điều kiện f(0) < 0 và f(1) > 0, do đó có thể dùng thuật toán tìm kiếm nhị phân giống như ở bài toán đoán số nguyên trong chương 6 để tìm nghiệm gần đúng của phương trình trên đoạn [0,1].

Cách thức thực hiện như sau:

- Đặt 2 giá trị đầu mút của đoạn chứa nghiệm là 0 và 1
- Với mỗi bước, tính giá trị hàm số tại điểm chính giữa của đoạn chứa nghiệm. Thu hẹp đoạn chứa nghiệm sao cho giá trị của hàm số tại 2 đầu của đoạn này luôn trái dấu nhau.
- Khi độ dài đoạn chứa nghiệm đủ nhỏ thì dừng lại.

Chương trình :

```
def f(x):
    return x*x*x + x - 1

a = 0
b = 1
eps = 1e-6
fa = f(a)
fb = f(b)

while True:
    x = (a+b)/2
    fx = f(x)

    if abs(fx) < eps:
        break
    elif fx * fa > 0:
        a = x
    else:
        b = x

print('x=', round(x,6))
```

Chương 8 : Dữ liệu kiểu văn bản (String)

Trong các chương đầu chúng ta đã biết về dữ liệu kiểu văn bản (chủ yếu là để hiển thị các dòng thông báo ra màn hình). Trong chương này, chúng ta sẽ tìm hiểu kỹ hơn về kiểu dữ liệu này.

Như ở phần đầu đã giới thiệu, các giá trị dữ liệu văn bản được đặt trong 2 dấu nháy đơn (') hoặc 2 dấu nháy kép (")

```
hoten = 'Nguyễn Văn An'
diachi = "Hà Nội"
```

Trong chương này, chúng ta tìm hiểu các phép tính trên dữ liệu kiểu văn bản.

Các hàm và phép tính thông dụng về dữ liệu văn bản trong Python:

- Hàm `str` : chuyển đổi số (và các kiểu dữ liệu khác) sang dữ liệu văn bản
- Phép + 2 chuỗi : ghép 2 chuỗi văn bản thành một
- Hàm `lower` chuyển văn bản sang chữ thường

- Hàm `upper` chuyển văn bản sang chữ hoa
- Hàm `replace` : thay nội dung một xâu con trong xâu văn bản
- Hàm `split` : tách một xâu thành các xâu con

Ví dụ về các phép tính trên dữ liệu văn bản:

```
str(1000) -> '1000'
'Chào ' + 'bạn' -> 'Chào bạn'
'Chào bạn'.lower() -> 'chào bạn'
'Chào bạn'.upper() -> 'CHÀO BẠN'
'Tôi sống ở Hà Nội'.replace('Hà Nội', 'Huế') -> 'Tôi sống ở Huế'
'Hà Nội'.split() -> ['Hà', 'Nội']
'Hà Nội, Việt Nam'.split(',') -> ['Hà Nội', 'Việt Nam']
```

Kí tự (character):

Một xâu văn bản thực chất là một dãy kí tự. Để lấy kí tự ở vị trị `index` trong xâu, chúng ta dùng cú pháp:

```
c = text[index]
```

Để lấy độ dài (số kí tự) trong 1 xâu, chúng ta dùng hàm `len` :

`len(text)` → độ dài của xâu

Chúng ta có thể dùng vòng lặp `for` để duyệt qua từng kí tự trong một xâu:

```
text = 'Chào bạn'

for c in text:
    print( c )
```

Một số hàm với kí tự:

- Hàm `ord` : lấy mã thứ tự của một kí tự trong bảng mã kí tự. Ví dụ `ord('A')` → `65`, `ord('Z')` → `90`
- Hàm `chr` : chuyển ngược từ thứ tự của một kí tự trong bảng kí tự thành kí tự đó. Ví dụ `chr(65)` → `'A'`

Xâu con (substring):

Với một xâu văn bản, chúng ta có thể lấy ra từng đoạn nhỏ của xâu đó. Mỗi đoạn này gọi là một xâu con.

Cú pháp để lấy một xâu con từ một xâu văn bản có trước:

```
text[start:end]
```

Trong đó các chỉ số `start, end` là các chỉ số bắt đầu và kết thúc của các kí tự được lấy ra từ xâu gốc. Lưu ý là giống như với hàm `range` ở chương 6, các kí tự sẽ chỉ được lấy từ vị trí `start` đến `end-1`

Nếu giá trị `end` được bỏ qua thì xâu con sẽ được lấy từ vị trí `start` đến hết xâu gốc:

```
text[start:]  ~  text[start: len(text)]
```

Nếu giá trị start được bỏ qua thì xâu con sẽ được lấy từ đầu xâu gốc:

```
text[:end]  ~  text[0: end]
```

Nếu các chỉ số start, end có giá trị âm, chúng được hiểu là tính từ cuối xâu trở về, ví dụ:

```
text[-2 : -1]  ~  text[len(text)-2 : len(text)-1]
```

Ví dụ 1:

```
text = 'Chào bạn'
print(text[0:4])
print(text[5:])
print(text[:4])
print(text[-1])
print(text[-3:])
```

Các giá trị được in ra màn hình:

```
Chào
bạn
Chào
n
Bạn
```

Ví dụ 2:

Nhập vào từ bàn phím một số tự nhiên và in ra biểu diễn nhị phân của số đó.

Lời giải:

Để chuyển một số từ hệ thập phân sang nhị phân, chúng ta chia số đó cho 2 liên tiếp cho đến khi thương bằng 1. Số dư các lần chia được viết từ phải qua trái.

Chương trình :

```
x = input("Nhập số tự nhiên : ")
x = int(x)

s = ''
while x > 0:
    i = x % 2
    s = str(i) + s
    x = x // 2

print(s)
```

Ví dụ 3:

Mã hóa một bức điện.

An và Bình trao đổi qua email. Để tránh việc bị lộ nội dung trao đổi ra ngoài, An dùng chương trình để mã hóa nội dung gửi đi. Giả sử nội dung bức điện chỉ chứa các chữ cái hoa trong bảng chữ cái tiếng Anh (từ 'A' đến 'Z') và kí tự trắng (phân cách các từ). Chương trình mã hóa được thực hiện như sau:

- An chọn một số tự nhiên k trong khoảng từ 1 đến 25, dùng làm khóa hay mật mã

- Với mỗi kí tự trong xâu nội dung, nếu kí tự đó là kí tự trắng (phân cách các từ) thì giữ nguyên, nếu kí tự là chữ cái thì dịch đi k vị trí trong bảng chữ cái tiếng Anh, khi dịch qua 'Z' thì quay về 'A' trong bảng chữ cái.

Bạn hãy giúp An viết chương trình mã hóa trên. Để đơn giản, An chọn trước khóa k = 10.

Lời giải:

Giả sử chúng ta có kí tự c và cần dịch đi k vị trí trong bảng chữ cái, khi đó các bước cần thực hiện là:

```
x = ord(c) - ord('A')
x += k
x = x % 26
x = x + ord('A')
c = chr(x)
```

Giải thích:

- Dòng đầu tiên tính vị trị của kí tự c trong bảng chữ cái tiếng Anh (bắt đầu từ 'A')
- Dòng thứ 2 dịch đi k vị trị
- Dòng thứ 3 đảm bảo nếu vị trí dịch qua 'Z' sẽ quay về 'A'
- Dòng thứ 4 biến đổi vị trị được dịch tới về kí tự tương ứng trong bảng chữ cái.

Với đoạn mã hóa một kí tự trên, chúng ta xây dựng chương trình mã hóa bức điện đầy đủ như sau:

```
k = 10
msg = 'TOI NAY CO DI CHOI KHONG'

encoded_msg = ''
for c in msg:
    if c == ' ':
        encoded_msg += c
    else:
        x = ord(c) - ord('A')
        x += k
        x = x % 26
        x = x + ord('A')
        encoded_msg += chr(x)

print(encoded_msg)
```

Nội dung bức điện được mã hóa là 'DYS XKI MY NS MRYS URYXQ'

Ví dụ 4:
Giải mã bức điện.

Tiếp tục với ví dụ 2. Sau khi Bình nhận được bức điện của An, cần giải mã lại nội dung bức điện. Giả sử An cho Bình biết khóa đã mã hóa là k = 10. Bạn hãy giúp Bình giải mã bức điện.

Lời giải:

Về cơ bản, chương trình giải mã giống với chương trình mã hóa, chỉ khác là khóa giải mã khác với khóa mã hóa. Chúng ta biết nếu dịch một kí tự đi 26 vị trí trong bảng chữ cái thì sẽ trở về chính nó, do đó ta chọn khóa giải mã sao cho tổng khóa mã hóa và khóa giải mã bằng 26.

Chương trình giải mã của Bình như sau:

```
k = 10
k = 26 - k
msg = 'DYS XKI MY NS MRYS URYXQ'

decoded_msg = ''
for c in msg:
    if c == ' ':
        decoded_msg += c
    else:
        x = ord(c) - ord('A')
        x += k
        x = x % 26
        x = x + ord('A')
        decoded_msg += chr(x)

print(decoded_msg)
```

Bạn hãy kiểm tra xem bức điện Bình giải mã được có giống bức điện gốc trước khi mã hóa của An không.

Chương 9 : Dữ liệu kiểu danh sách(List) và kiểu bộ nhóm (Tuple)

Kiểu dữ liệu danh sách (List)

Kiểu dữ liệu danh sách (List) dùng để chứa một dãy nhiều phần tử. Khi khai báo giá trị, các phần từ của một danh sách được đặt trong cặp dấu [ ] , ví dụ:

```
dayso = [1, 3, 5, 7, 9]
danhsach_hocsinh = ["Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn"]
```

Các phần tử của một danh sách có thể không cùng kiểu dữ liệu:

```
diachi = [322, "Tây Sơn", "Hà Nội"]
```

Các hàm và phép tính thông dụng trên dữ liệu kiểu danh sách:

- Hàm `len` : Lấy số phần tử của một danh sách
- Hàm `append` : Thêm phần từ vào danh sách
- Hàm `remove` : Xóa một phần tử khỏi danh sách
- Phép + 2 danh sách : tạo thành một danh sách mới bằng cách ghép 2 danh sách thành phần
- Hàm `extend` : Ghép một danh sách con vào cuối một danh sách.
- Hàm reverse : Đảo ngược một danh sách
- Phép `in` : kiểm tra một giá trị có nằm trong danh sách không
- Phép `not in` : Kiểm tra một giá trị có không nằm trong danh sách không

Tương tự với kiểu dữ liệu văn bản ở chương 8, kiểu danh sách cũng cho phép truy cập đến từng phần tử qua chỉ số, đồng thời cho phép lấy ra các danh sách con theo các chỉ số đầu và cuối:

- `danhsach[i]` : phần tử thứ `i` của danh sách
- `danhsach[start:end]` : danh sách con chứa các phần tử từ chỉ số `start` đến `end-1`

- `danhsach[start:]` : danh sách con chứa các phần tử từ chỉ số `start` đến hết danh sách
- `danhsach[: end]` : danh sách con chứa các phần tử từ chỉ số `0` đến `end-1`

Ví dụ 1:

```
ds = [1, 2, 3, 4, 5]
print(len(ds))
# Kết quả : 5

ds.append(6)
print(ds)
# Kết quả : [1,2,3,4,5,6]

ds.remove(3)
print(ds)
# Kết quả : [1,2,4,5,6]

ds = ds + [7, 8]
print(ds)
# Kết quả : [1,2,4,5,6,7,8]

ds.extend([9, 10])
print(ds)
# Kết quả : [1,2,4,5,6,7,8,9,10]

if 1 in ds:
    print('1 nằm trong danh sách')

if 3 not in ds:
    print('3 không nằm trong danh sách')

print(ds[5])
# Kết quả : 7

print(ds[1:4])
# Kết quả : [2,4,5]

print(ds[:3])
# Kết quả : [1,2,3]

print(ds[6:])
# Kết quả : [8,9,10]

print(ds[-2:])
# Kết quả : [9,10]

ds.reverse()
print(ds)
# Kết quả : [10,9,8,7,6,5,4,2,1]
```

Ví dụ 2:

Viết chương trình in ra các số nguyên tố nhỏ hơn 1000

Lời giải:

```
ds = []
```

```
for x in range(2, 1000):
    nguyento = True

    for p in ds:
        if x % p == 0:
            nguyento = False
            break

        if p *p > x:
            break

    if nguyento:
        ds.append(x)

print(ds)
```

Ví dụ 3:

Tam giác Pascal.
Mỗi dòng của tam giác Pascal là các hệ số của khai triển nhị thức $(a+b)^n$:

$(a+b)^0 = 1 \rightarrow 1$
$(a+b)^1 = a + b \rightarrow 1 \quad 1$
$(a+b)^2 = a^2 + 2ab + b^2 \rightarrow 1 \quad 2 \quad 1$
$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3 \rightarrow 1 \quad 3 \quad 3 \quad 1$

Viết chương trình để in ra 10 dòng đầu của tam giác Pascal.

Lời giải:

```
N = 10
heso = []

for i in range(N):
    heso.append(1)
    for j in range(i-1, 0, -1):
        heso[j] += heso[j-1]

    print(heso)
```

Dữ liệu kiểu bộ nhóm (Tuple)

Dữ liệu kiểu bộ nhóm (Tuple) tương tự với kiểu danh sách (List), chỉ khác là các phần tử sau khi khai báo thì không sửa (thêm, xóa) được. Các phần tử của dữ liệu kiểu nhóm được đặt trong cặp dấu ( ) :

```
dayso = (1, 3, 5, 7, 9)
danhsach_hocsinh = ("Nguyễn Văn An", "Nguyễn Chí Cường", "Nguyễn Mạnh Tuấn")
```

Dữ liệu kiểu bộ nhóm có thể dùng để gán cho một bộ nhiều biến số:

```
x, y = 1, 2
```

Việc này tương đương với gán giá trị của từng thành phần trong bộ dữ liệu cho từng biến tương ứng ở bên trái câu lệnh.

Chương 10 : Dữ liệu kiểu tập hợp (Set) và kiểu từ điển (Dictionary)

Kiểu dữ liệu tập hợp (Set)

Kiểu dữ liệu tập hợp (Set) dùng để chứa các phần tử của một tập hợp. So với kiểu danh sách ở chương 9, kiểu Set có điểm khác:

- Trong tập hợp không có 2 phần tử cùng giá trị. Nếu một phần tử đã có trong tập hợp thì khi thêm mới phần tử cùng giá trị vào tập hợp, chúng chỉ được tính là một

- Không thể dùng chỉ số để truy cập các phần tử trong tập hợp như cách làm với danh sách

Các hàm và phép tính trên tập hợp:

- Hàm `add` : thêm giá trị mới vào tập
- Hàm `remove` : xóa phần tử trong tập hợp
- Phép `in` : kiểm tra giá trị có nằm trong tập hợp không
- Phep `not in` : kiểm tra giá trị có không nằm trong tập hợp không
- Hàm `union` : hợp của 2 tập hợp
- Hàm `intersection` : giao của 2 tập hợp

Ví dụ 1:

```
s1 = set([1, 2])
s1.add(3)
s1.remove(1)

if 1 not in s1:
    print("1 không nằm trong tập hợp ", s1)

if 2 in s1:
   print("2 nằm trong tập hợp ", s1)

s2 = set([3, 4])
s3 = set.union(s1, s2)
print(s3)

s4 = set.intersection(s1, s2)
print(s4)
```

Kết quả in ra màn hình:

```
1 không nằm trong tập hợp  {2, 3}
2 nằm trong tập hợp  {2, 3}
{2, 3, 4}
{3}
```

Ví dụ 2:

```
weekdays = set(['Thứ hai', 'Thứ ba', 'Thứ tư', 'Thứ năm', 'Thứ sáu'])
weekends = set(['Thứ bảy', 'Chủ nhật'])

day = "Thứ ba"

if day in weekdays:
   print(day, ' là ngày làm việc')

if day in weekends:
   print(day, ' là ngày nghỉ cuối tuần')
```

Kiểu dữ liệu từ điển (Dictionary):

Kiểu dữ liệu từ điển dùng để chứa một bảng biến đổi 1-1 giữa 2 tập hợp :

- Tập nguồn (hay tập khóa (key)) : chứa các thông tin cần tra cứu
- Tập đích (hay tập giá trị (value)) : chứa thông tin của mỗi key từ tập nguồn

Bạn hãy hình dung nó giống một cuốn từ điển để tra cứu nghĩa của các từ trong một ngôn ngữ này sang một ngôn ngữ khác.

Các phần tử của kiểu dữ liệu từ điển được khai báo theo từng cặp và đặt trong cặp dấu { }, các cặp được ngăn cách nhau bởi dấu phảy, hai giá trị trong cặp ngăn cách nhau bởi dấu hai chấm:

```
d = {"một" : 1, "hai" : 2, "ba" : 3}
```

Để lấy giá trị đích của một giá trị nguồn `key` trong từ điển d, chúng ta dùng cú pháp:

```
d[key]
```

Với cú pháp trên, nếu giá trị `key` chưa nằm trong từ điển thì chương trình sẽ báo lỗi, để khắc phục, chúng ta có thể dùng hàm:

```
d.get(key, <gia_tri_mac_dinh>)
```

Hàm này trả về giá trị đích tương ứng với `key` nếu `key` nằm trong từ điển d, nếu key không nằm trong `d` thì trả về `<gia_tri_mac_dinh>`

Ví dụ 3:
Viết chương trình đếm tần suất của các từ trong một câu. Ví dụ:

Đầu vào:

```
"Một năm có mười hai tháng, tháng hai có hai mươi tám ngày, các tháng còn
lại có ba mươi hoặc ba mươi mốt ngày"
```

Kết quả:

```
một : 1
hai : 3
tháng : 3
…
```

Lời giải:

```
text = 'Một năm có mười hai tháng, tháng hai có hai mươi tám ngày, các tháng
còn lại có ba mươi hoặc ba mươi mốt ngày'
```

```
text = text.lower()
for c in ['.', ',' , ':']:
    text = text.replace(c, ' ')

words_count = {}

for word in text.split():
    words_count[word] = words_count.get(word, 0) + 1

for word in words_count:
    print(word , ' : ', words_count[word])
```

Chương 11 : Dữ liệu kiểu tệp tin (File)

Kiểu tệp tin (File) dùng để truy xuất dữ liệu từ các file lưu trữ trên ổ cứng.

Dữ liệu lưu trong các file được chia thành 2 loại:

- Dữ liệu dạng văn bản : nội dung file là một xâu văn bản, ví dụ các loại file có tên mở rộng như txt, py, html …
- Dữ liệu dạng nhị phân : nội dung file là chuỗi byte, ví dụ như các file ảnh (bmp, jpg, png), các file âm thanh (mp3, wav), các file video (mp4, avi, wmv) …

Trong chương này, chúng ta chủ yếu quan tâm đến các file dữ liệu văn bản.

Đọc dữ liệu từ file:

Để đọc dữ liệu từ file văn bản, trước hết chúng ta cần mở file:

```
f = open('ten_file')
```

Nếu làm việc với các file văn bản chứa kí tự Unicode (như các kí tự tiếng Việt), chúng ta cần khai báo thêm thông tin về dạng mã hóa file là utf-8 (dạng mặc định của mã hóa Unicode):

```
f = open('ten_file', encoding='utf-8')
```

Lệnh trên trả về đối tượng f chứa các thông tin để truy xuất đến file có tên như đã truyền vào cho hàm open. Sau khi đã mở file, chúng ta có thể đọc toàn bộ nội dung file vào một biến dạng xâu văn bản với lệnh:

```
content = f.read()
```

Trong trường hợp không muốn đọc toàn bộ file mà duyệt qua từng dòng trong file, chúng ta có thể dùng cú pháp:

```
for line in f:
    print(line)   # line chứa nội dung mỗi dòng trong file
```

Sau khi đọc xong nội dung file, chúng ta cần đóng file lại, để các chương trình khác có thể truy xuất đến file mà không bị ảnh hưởng:

```
f.close()
```

Ghi dữ liệu vào file:

Tương tự như khi đọc file, trước hết chúng ta cần mở file với lệnh:

```
f = open('ten_file', 'w')
```

Lệnh trên tạo một file mới với nội dung trống, nếu muốn ghi dữ liệu vào cuối một file đã tồn tại, chúng ta dùng lệnh:

```
f = open('ten_file', 'a')
```

Sau khi đã mở file, chúng ta dùng lệnh sau để ghi nội dung một xâu văn bản vào file:

```
f.write('<noi_dung>')
```

Khi đã hoàn thành việc ghi file, chúng ta cần đóng file lại để đảm bảo nội dung được lưu hết vào ổ cứng:

```
f.close()
```

Ví dụ 1:

Trong thư mục của chương trình chạy có một file văn bản `input.txt`. Viết chương trình copy nội dung của file này sang file `output.txt` , nếu mỗi dòng của file đầu vào là dòng trống (không chứa kí tự) thì sẽ bỏ qua và không copy sang file đầu ra. Ví dụ:

File input.txt:

```
Một năm có 365 hoặc 366 ngày
Năm thường có 365 ngày

Năm nhuận có 366 ngày
```

File output.txt:

```
Một năm có 365 hoặc 366 ngày
Năm thường có 365 ngày
Năm nhuận có 366 ngày
```

Lời giải:

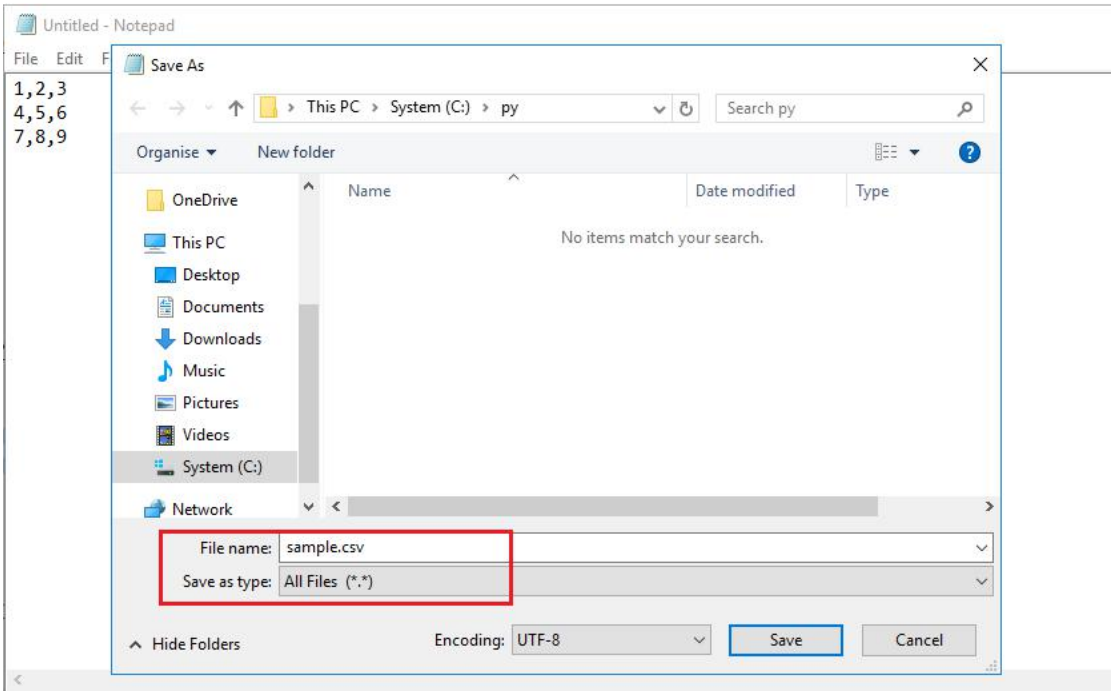
```
infile = open('input.txt', encoding='utf-8')
outfile = open('output.txt', 'w', encoding='utf-8')

for line in infile:
    if line.strip() != '':
        outfile.write(line)

infile.close()
outfile.close()
```

Ví dụ 2:

File có tên mở rộng csv (comma separted value), là file văn bản dùng để mô tả các bảng dữ liệu. Mỗi dòng của file là một hàng trong bảng, trên một dòng, các cột được ngăn cách nhau bởi dấu phảy. Bạn hãy dùng notepad tạo file `sample.csv` với nội dung như sau:

```
1,2,3
4,5,6
7,8,9
```

Khi lưu lại file, bạn nhớ chọn tên mở rộng là csv :



Lưu nội dung văn bản dưới dạng csv bằng notepad

Sau khi file `sample.csv` như trên đã được tạo ra, nếu trên máy tính của bạn có cài đặt Excel hoặc chương trình mở các bảng tài liệu, bạn có thể mở file csv vừa tạo ở trên để xem nội dung dưới dạng bảng dữ liệu :

	A	B	C	D
1	1	2	3	
2	4	5	6	
3	7	8	9	
4				

Nội dung file csv được xem ở dạng bảng dữ liệu

Sau khi đã hiểu về cơ chế hoạt động của các file csv, chúng ta hãy làm ví dụ sau:

Trong thư mục chạy chương trình có một file `input.csv` với nội dung chứa bảng điểm một môn học của một lớp học. Mỗi dòng là thông tin điểm của một học sinh, bao gồm : họ tên, điểm hệ số 1, điểm hệ số 2, điểm hệ số 3, các giá trị này ngăn cách nhau bởi dấu phảy.

Bạn hãy viết chương trình để tính điểm trung bình cho toàn bộ học sinh trong lớp. Dữ liệu được tính toán xong lưu ra file `output.csv`, nội dung tương tự với file `input.csv`, nhưng có thêm cột điểm trung bình ở cuối.

File input.csv:

Dạng văn bản (mở với notepad):

```
Nguyễn Văn An, 8, 7, 8
Nguyễn Văn Bình, 6, 6, 8
Nguyễn Thị Chi, 8, 8, 9
Lê Văn Cường, 8, 7, 9
Phạm Thu Trang, 7, 8, 8
```

Dạng bảng dữ liệu (mở với Excel):

	A	B	C	D
1	Nguyễn Văn An	8	7	8
2	Nguyễn Văn Bình	6	6	8
3	Nguyễn Thị Chi	8	8	9
4	Lê Văn Cường	8	7	9
5	Phạm Thu Trang	7	8	8

File output.csv (cần tạo ra):

Dạng văn bản (mở với notepad)

```
Nguyễn Văn An, 8, 7, 8,  7.7
Nguyễn Văn Bình, 6, 6, 8,  7.0
Nguyễn Thị Chi, 8, 8, 9,  8.5
Lê Văn Cường, 8, 7, 9,  8.2
Phạm Thu Trang, 7, 8, 8,  7.8
```

Dạng bảng dữ liệu (mở với Excel)

	A	B	C	D	E
1	Nguyễn Văn An	8	7	8	7.7
2	Nguyễn Văn Bình	6	6	8	7
3	Nguyễn Thị Chi	8	8	9	8.5
4	Lê Văn Cường	8	7	9	8.2
5	Phạm Thu Trang	7	8	8	7.8

Lời giải:

```
ds_lop = []

fi = open('input.csv', encoding='utf-8')

for line in fi:
    hoten, diemhs1, diemhs2, diemhs3 = line.split(',')
    diemhs1 = int(diemhs1)
    diemhs2 = int(diemhs2)
```

```
        diemhs3 = int(diemhs3)
        diem_tb = (diemhs1 + 2*diemhs2 + 3*diemhs3) / 6.0
        ds_lop.append((hoten, diemhs1, diemhs2, diemhs3, diem_tb))

    fi.close()

    fo = open('output.csv', 'w', encoding='utf-8')

    for hocsinh in ds_lop:
        hoten, diemhs1, diemhs2, diemhs3, diem_tb = hocsinh
        line = '{}, {}, {}, {}, {: 1.1f}'.format(
            hoten, diemhs1, diemhs2, diemhs3, diem_tb)

        fo.write(line + '\n')

    fo.close()
```

Bài tập:

1. Tìm 2 số biết tổng và tỉ số của chúng. Nhập vào từ bàn phím các giá trị S : tổng 2 số, R : tỉ số của 2 số , in ra màn hình giá trị 2 số đó.

2. Tìm 2 số khi biết tổng và tích. Nhập vào từ bàn phím các giá trị S (tổng 2 số), P(tích 2 số). In ra màn hình giá trị 2 số nếu tồn tại, nếu không thì thông báo không tồn tại 2 số.

3. Một máy bán hàng tự động cho phép người dùng đưa vào các tờ tiền có mệnh giá chia hết cho 5000. Trong trường hợp người mua hàng đưa vào nhiều tiền hơn giá trị mặt hàng mua (cũng có giá là bội số của 5000), máy sẽ trả lại tiền thừa cho người mua. Máy có 3 loại tiền để trả lại là 5 nghìn, 20 nghìn và 50 nghìn. Bạn hãy viết chương trình nhập vào từ bàn phím số tiền cần trả lại (theo đơn vị nghìn đồng), sau đó tính số tiền trả lại mỗi loại (5k, 20k, 50k) sao cho tổng số tờ tiền trả lại là nhỏ nhất.

4. Viết chương trình tính độ dài một đường gấp khúc trên mặt phẳng

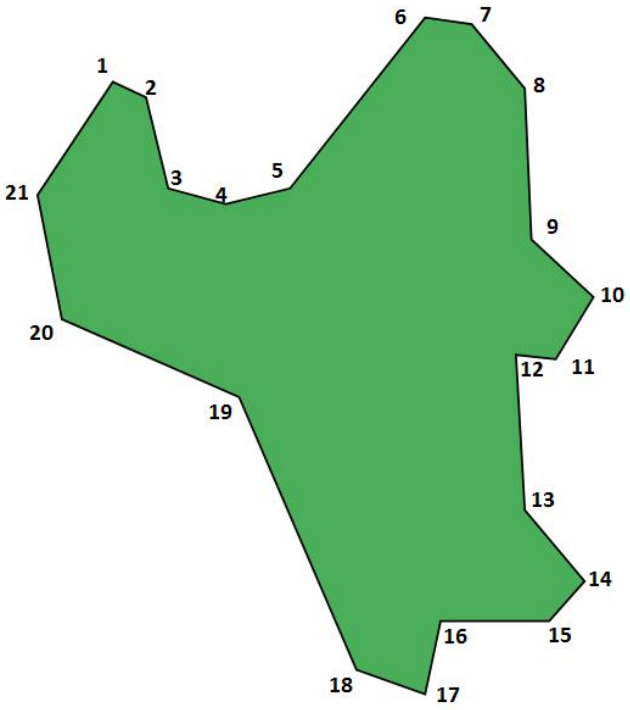
5. Viết chương trình tính diện tích của một đa giác (có thể không lồi) tạo thành từ N điểm cho trước trên mặt phẳng

6. Để định vị một điểm trên mặt đất, người ta dùng tọa độ GPS gồm kinh độ, vĩ độ, (và độ cao). Trên bản đồ Hà Nội, hồ Hoàn Kiếm có vĩ độ là 21.0259198, kinh độ 105.8444646 . Lăng Bác có vĩ độ 21.0379367, kinh độ 105.8324912. Biết bán kính trái đất là 6371 km, bạn hãy tính khoảng cách theo đường chim bay từ hồ Hoàn Kiếm đến lăng Bác.

7. Trên bản đồ (năm 2018), thành phố Hà Nội có thể được xem là một đa giác tạo thành từ các điểm có tọa độ trong hệ GPS (vĩ độ/kinh độ) như sau:

21.303867, 105.392923
21.286594, 105.436868
21.178428, 105.464334
21.157297, 105.531625
21.173946, 105.625696
21.385090, 105.800790
21.373581, 105.857095
21.291712, 105.928507
21.125276, 105.916147
21.045833, 106.019830

20.965706, 105.962152
20.973400, 105.912714
20.793128, 105.921640
20.707085, 105.998544
20.653124, 105.944986
20.642843, 105.817270
20.568288, 105.798044
20.602999, 105.703287
20.927871, 105.554971
21.020197, 105.322885
21.157297, 105.292673



Xấp xỉ gần đúng thành phố Hà Nội bằng 1 đa giác trên bản đồ

Biết bán kính trái đất là 6371 km, bạn hãy viết chương trình tính diện tích theo km^2 của thành phố Hà Nội

8. Nhập vào từ bàn phím 3 giá trị ngày, tháng, năm (trong thế kỉ 21). Kiểm tra ngày đó có tồn tại không. Ví dụ:

- Ngày 29/2/2000 : có tồn tại
- Ngày 29/2/2001 : không tồn tại
- Ngày 31/7/2010 : có tồn tại
- Ngày 31/6/2016 : không tồn tại

9. Viết chương trình nhập vào một ngày trong thế kỉ 21 và cho biết ngày đó là thứ mấy trong tuần

10. Viết chương trình kiểm tra tính hợp lệ của mật khẩu người dùng : người dùng nhập mật khẩu từ bàn phím, chương trình kiểm tra mật khẩu có hợp lệ không. Mật khẩu hợp lệ cần có độ dài tối thiểu là 6, chứa ít nhất một chữ số (0-9) và chứa ít nhất một chữ cái (a-z/A-Z)

11. Viết chương trình chuyển một số trong phạm vi 0-999 thành dạng văn bản theo phát âm tiếng Việt, ví dụ 102 → 'một trăm lẻ hai'

12. Viết chương trình chuyển một xâu chứa phát âm tiếng Việt của một số trong phạm vi 0-999 thành giá trị của số đó, ví dụ 'ba trăm hai mươi tư' → 324. Nếu xâu văn bản đầu vào không tương ứng với số nào thì thông báo không tồn tại số.

13. Một file csv chứa danh sách các mặt hàng đã được bán ra của một cửa hàng trong ngày, mỗi dòng là thông tin về một lần khách mua hàng, bao gồm mã hàng hóa, số lượng hàng hóa bán ra, giá gốc nhập về và giá bán ra. Các dòng khác nhau có thể có cùng mã hàng hóa (nhiều người mua cùng 1 mặt hàng). Viết chương trình tạo ra file csv trong đó mỗi dòng chứa thông tin về tình trạng kinh doanh của một mặt hàng, gồm các cột : mã hàng hóa (chỉ tính một lần duy nhất cho các lần mua khác nhau của cùng một mặt hàng), tổng số lượng đã bán, tổng doanh số (số tiền thu được do bán hàng), tổng lợi nhuận (số tiền thu được sau khi trừ đi giá gốc nhập về). Các mặt hàng xếp theo trình tự giảm dần của doanh số. Ví dụ :

File input.csv:

```
VINAMILK01, 5, 27000, 20000
THMILK03, 10, 28000, 20000
OMACHI05, 2, 6000, 4000
VINAMILK01, 5, 27000, 20000
VINAMILK01, 10, 27000, 20000
OMACHI05, 5, 6000, 4000
```

File output.csv:

```
VINAMILK01, 20, 540000, 140000
THMILK03, 10, 280000, 80000
OMACHI05, 7, 42000, 14000
```

Phần 2 : Lập trình đồ họa với Turtle

Giới thiệu:

Turtle là module của Python cho phép vẽ hình bằng cách điều khiển tọa độ của một bút vẽ (có biểu tượng con rùa - do đó module được gọi là turtle).

Khi khởi động module, tọa độ của bút (turtle) nằm ở vị trí (0,0). Bằng các lệnh di chuyển (đi thẳng, rẽ trái, rẽ phải, ….), bạn có thể điều khiển bút vẽ để tạo ra các khối hình học. Một cách khác, bạn có thể hình dung turtle là một robot, chương trình sẽ điều khiển robot đi theo một quỹ đạo mong muốn, vết di chuyển của robot trên màn hình chính là hình sẽ được vẽ ra.

Các chuyển động vẽ cơ bản

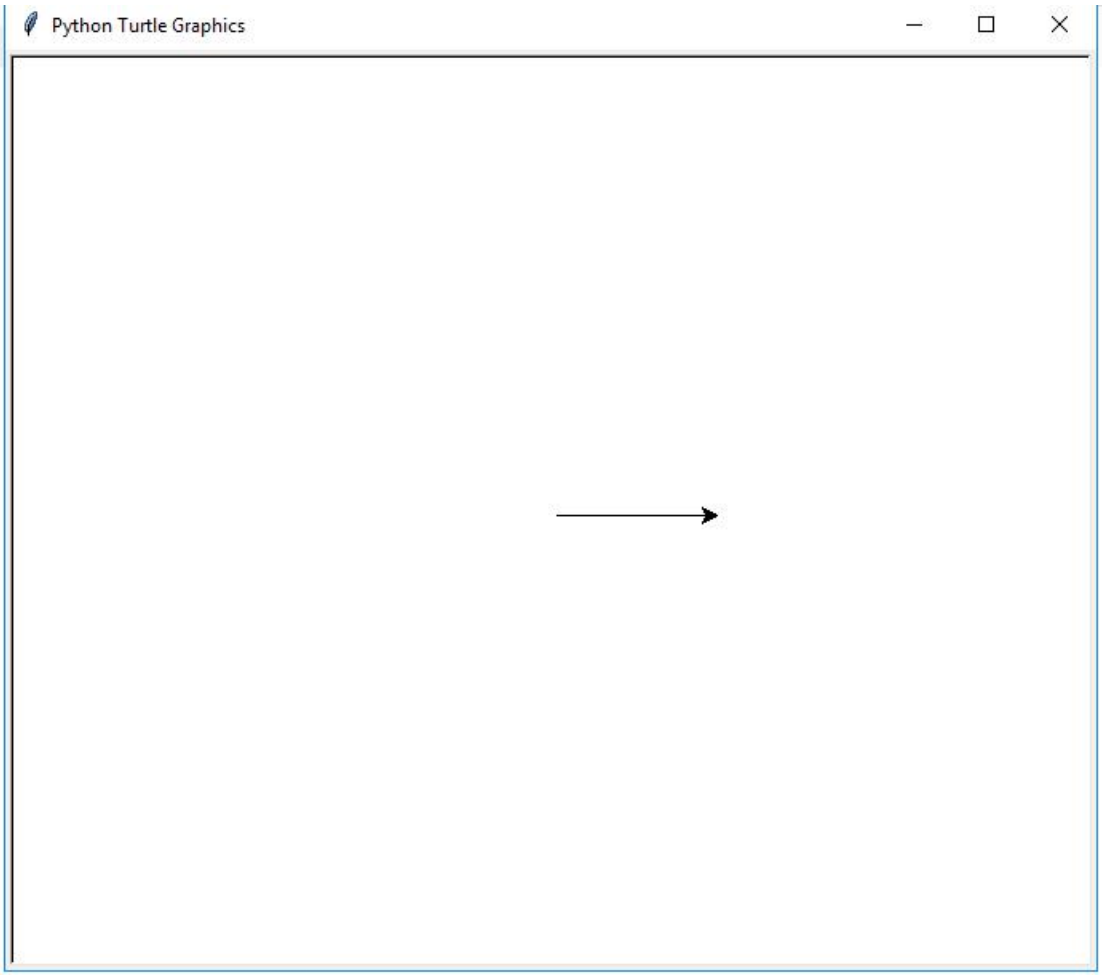
Ví dụ 1:

```
from turtle import *
forward(100)
```

`done()`

Lệnh `forward` được dùng để yêu cầu turtle di chuyển trên đường thẳng:

`forward(distance)` : di chuyển thẳng một đoạn có độ dài bằng `distance`



Chuyển động của "robot" turtle trên đường thẳng

Quan sát hình vẽ tạo ra, bạn sẽ thấy turtle xuất phát từ vị trí gốc tọa độ (0,0) sau đó đi thẳng sang phải một đoạn đường bằng 100 (pixel).

Để đổi hướng di chuyển của turtle, chúng ta có thể dùng các lệnh:

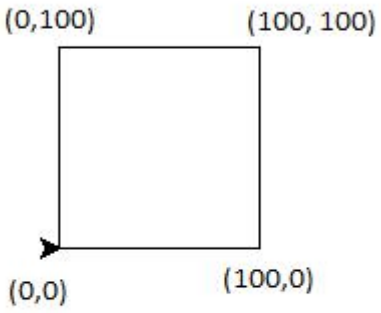
- `right(angle)` : xoay hướng di chuyển đi một góc `angle` sang phải so với hướng hiện tại
- `left(angle)` : xoay hướng di chuyển đi một góc `angle` sang trái so với hướng hiện tại
- `setheading(angle)` : đặt hướng di chuyển theo đường thẳng hợp với trục nằm ngang một góc `angle`

Sử dụng các hàm trên, chúng ta hãy vẽ một số hình cơ bản

Ví dụ 2:

Vẽ hình vuông.

```
from turtle import *

forward(100)
left(90)

forward(100)
left(90)

forward(100)
left(90)

forward(100)
done()
```



Chuyển động của turtle theo quỹ đạo hình vuông

Quan sát hình được vẽ ra, chúng ta thấy quỹ đạo chuyển động của turtle như sau:

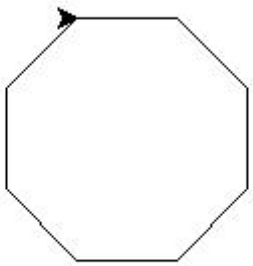
- Xuất phát (0,0)
- `forward(100)` : di chuyển theo phương ngang đến tọa độ (100, 0)
- `left(90)` : quay trái 90 độ, hướng di chuyển mới là đi lên
- `forward(100)` : di chuyển đến tọa độ (100, 100)
- `left(90)` : quay trái 90 độ, hướng di chuyển mới là sang trái
- `forward(100)` : di chuyển đến tọa độ (0, 100)
- `left(90)` : quay trái 90 độ, hướng di chuyển mới là đi xuống
- `forward(100)` : di chuyển đến tọa độ (0,0)
- `done()` : hoàn thành quá trình di chuyển

Ví dụ 3:

Vẽ đa giác đều.

```
from turtle import *

N = 8
side_length = 50
angle = 360.0 / N

for i in range(N):
    forward(side_length)
```

```
    right(angle)

done()
```

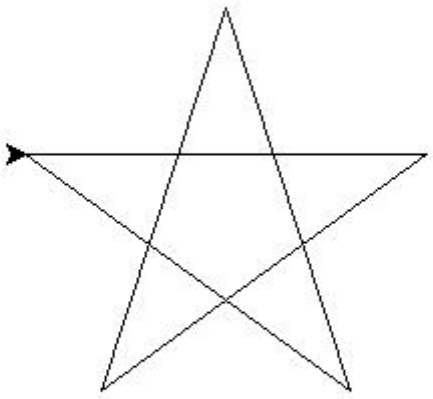


Đa giác đều được vẽ ra trong ví dụ 3

Ví dụ 4:

Vẽ ngôi sao năm cánh.

```
from turtle import *

for i in range(5):
    forward(200)
    right(144)

done()
```



Ngôi sao 5 cánh được vẽ ra trong ví dụ 4

Chọn màu vẽ và màu nền:

Để chọn màu vẽ (màu của các đường), và màu nền chúng ta dùng các lệnh:

```
pencolor('màu vẽ')
fill_color('màu nền')
color('màu vẽ', 'màu nền')
```

Các màu thông dụng:

- red
- green
- blue
- yellow
- cyan
- magenta
- black
- white

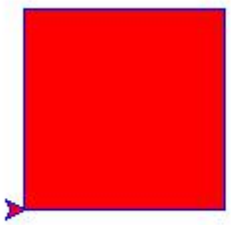
Để tô màu cho một hình, chúng ta đặt các lệnh vẽ hình đó trong một khối nằm giữa 2 lệnh `begin_fill` và `end_fill`

```
begin_fill()
# Vẽ hình
end_fill()
```

Ví dụ 5:

Vẽ một hình vuông có viền màu xanh và nền màu đỏ.

```
from turtle import *

color('blue', 'red')

begin_fill()

for i in range(4):
    forward(100)
    left(90)

end_fill()
done()
```



Hình vuông viền xanh, nền đỏ

Một số lệnh nâng cao:

- Lệnh `cirle(radius, angle)` : vẽ phần đường tròn có bán kính `radius` từ góc 0 độ đến góc `angle` (`angle=360` tương đương với toàn đường tròn)
- Lệnh `goto(x,y)` : di chuyển đến điểm có tọa độ (x,y)
- Lệnh `penup()` : ngừng vẽ trong quá trình di chuyển (nhấc bút vẽ lên khỏi màn hình)
- Lệnh `pendown()` : thực hiện vẽ trong quá trình di chuyển (đặt bút vẽ xuống)
- Lệnh `getx(), gety()` : lấy tọa độ hiện tại của turtle
- Lệnh `distance(x,y)` : trả về khoảng cách từ vị trí hiện tại đến điểm có tọa độ (x,y)
- Lệnh `towards(x,y)` : trả về góc giữa hướng chuyển động hiện tại của turtle với đường thẳng nối vị trí hiện tại và điểm có tọa độ (x,y)
- Lệnh `pensize()` : lấy giá trị độ dày đường vẽ hiện tại
- Lệnh `pensize(width)` : đặt độ dày đường vẽ
- Lệnh `stamp()` : "Đóng dấu" biểu tượng của turtle lên vị trí hiện tại
- Lệnh `clear()` : Xóa màn hình
- Lệnh `speed(s)` : Chọn tốc độ vẽ, các giá trị tốc độ là `'fatest', 'fast', 'normal', 'slow' , 'slowest'`

Ví dụ 6:

Chương trình vẽ tên lửa đuổi theo máy bay.

Máy bay trên bầu trời ở tọa độ (100,100) và đang bay theo hướng nằm ngang sang phải. Tên lửa xuất phát trên mặt đất tại vị trí (0,0) và đuổi theo máy bay. Tên lửa có radar phát hiện vị trí của máy bay, do đó trong quá trình di chuyển, tên lửa luôn được xoay để hướng vào vị trị của máy bay. Tốc độ chuyển động của tên lửa gấp đôi tốc độ máy bay. Bạn hãy vẽ hình minh họa chuyển động của 2 đối tượng.

```
import turtle

def createTurtle(x,y):
    t = turtle.Turtle()
    t.penup()
    t.goto(x, y)
    t.pendown()
    t.speed('fastest')
    return t

def distance(t1, t2):
    x1, y1 = t1.pos()
    x2, y2 = t2.pos()
    return abs(x2-x1) + abs(y2-y1)

airplane = createTurtle(100, 100)
missile = createTurtle(0, 0)
```

```
while distance(airplane, missile) > 2:
    alpha = missile.towards(airplane.pos())
    missile.setheading(alpha)

    airplane.forward(1)
    missile.forward(2)

turtle.done()
```

Mô phỏng chuyển động của tên lửa đuổi theo máy bay

Ví dụ 7:

Bài toán chuyển động của 3 robot.

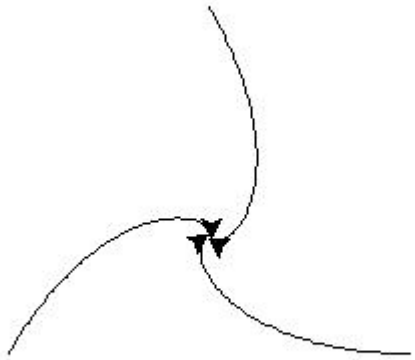
Ba robot xuất phát tại 3 đỉnh của một tam giác đều, và cùng bắt đầu chuyển động. Trong quá trình chuyển động các robot được điều khiển sao cho phương chuyển động luôn hướng tới "mục tiêu" của mình :

- Robot 1 luôn giữ hướng chuyển động tới vị trí Robot 2
- Robot 2 luôn giữ hướng chuyển động tới vị trí Robot 3
- Robot 3 luôn giữ hướng chuyển động tới vị trí Robot 1

Hãy vẽ minh họa chuyển động của 3 robot.

```
import turtle

def createTurtle(x,y):
  t = turtle.Turtle()
  t.penup()
  t.goto(x, y)
  t.pendown()
  t.speed(0)
  return t

def distance(t1, t2):
  x1, y1 = t1.pos()
  x2, y2 = t2.pos()
  return abs(x2-x1) + abs(y2-y1)

t1 = createTurtle(-100, 0)
t2 = createTurtle(0, 173)
t3 = createTurtle(100, 0)
```

```
    while distance(t1, t2) > 4:
        alpha1 = t1.towards(t2.pos())
        t1.setheading(alpha1)

        alpha2 = t2.towards(t3.pos())
        t2.setheading(alpha2)

        alpha3 = t3.towards(t1.pos())
        t3.setheading(alpha3)

        t1.forward(2)
        t2.forward(2)
        t3.forward(2)

    turtle.done()
```



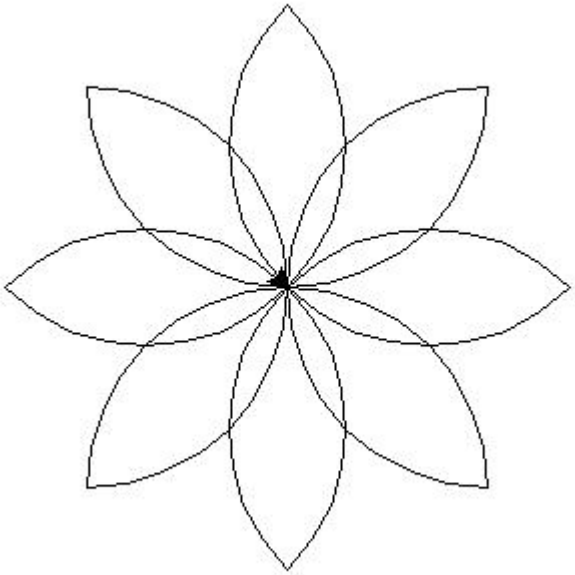
Quỹ đạo chuyển động của 3 robot

Ví dụ 8:

Vẽ bông hoa.

```
    # Chương trình vẽ bông hoa bằng turtle
    # Dựa trên chương trình gốc tại : http://nghiaho.com/?p=1603

    from turtle import *

    def petal(radius,steps):
        circle(radius,90,steps)
        left(90)
        circle(radius,90,steps)

    num_petals = 8
    steps = 8
    radius = 100

    speed('fastest')
```

```
for i in range(num_petals):
    setheading(0)
    right(360*i/num_petals)
    petal(radius,steps)

done()
```



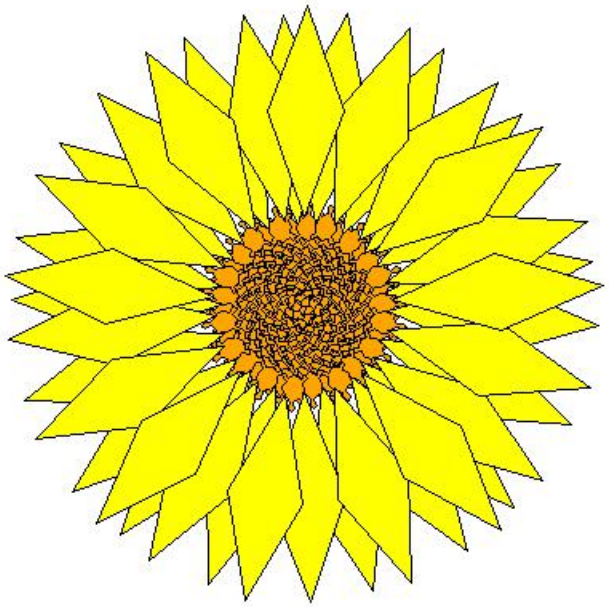
Bông hoa 8 cánh trong ví dụ 8

Ví dụ 9:

Vẽ bông hoa hướng dương.

```
# Chương trình vẽ bông hoa hướng dương bằng turtle
# Dựa trên chương trình gốc tại:
#     http://www.deborahrfowler.com/PythonResources/PythonTurtle.html

import math
from turtle import *

shape("turtle")
color("black")
speed("fastest")

def drawSunflower(numseeds, numpetals, angle, cspread):
    fillcolor("orange")
    phi = angle * (math.pi / 180.0)

    for i in range (numseeds + numpetals):
      # figure out the next x, y position
      r = cspread * math.sqrt(i)
      theta = i * phi
      x = r * math.cos(theta)
```

```
      y = r * math.sin(theta)

      # move the turtle and orient it correctly
      penup()
      goto(x, y)
      setheading(i * angle)
      pendown()

      if i <  numseeds:
        stamp()
      else:
        drawPetal()

def drawPetal():
    fillcolor("yellow")
    begin_fill()
    right(20)
    forward(70)
    left(40)
    forward(70)
    left(140)
    forward(70)
    left(40)
    forward(70)
    end_fill()

drawSunflower(160, 40, 137.508, 4)

hideturtle()
done()
```



Bông hoa hướng dương trong ví dụ 9

Hình học fractal

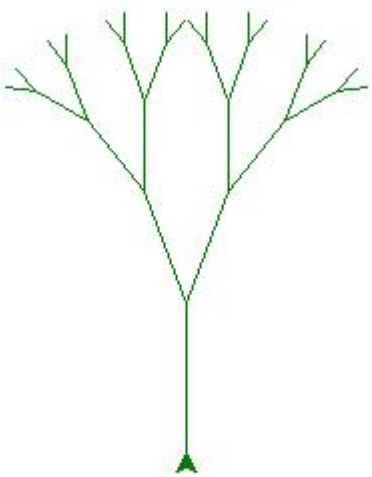
Hình học fractal là cách mô tả các đối tượng trong tự nhiên bằng các hình toán học. Các hình fractal thường là các hình tự đồng dạng, tức hình đó được cấu tạo từ các hình nhỏ mà bản thân mỗi phần tử nhỏ đó là một hình đồng dạng với chính hình gốc đó.

Turtle có thể được dùng để tạo ra các hình fractal mô phỏng các đối tượng trong tự nhiên.

Ví dụ 10:

Dùng hình học fractal để vẽ mô phỏng một hình cây.

```
# Chương trình vẽ cây bằng hình học fractal
# Dựa trên chương trình gốc tại: https://trinket.io/python/62ca928524

from turtle import *

def tree(branchLen):
    if branchLen > 5:
        forward(branchLen)
        right(20)
        tree(branchLen-15)
        left(40)
        tree(branchLen-15)
        right(20)
        backward(branchLen)

left(90)
up()
backward(100)
down()
color("green")
tree(75)

done()
```



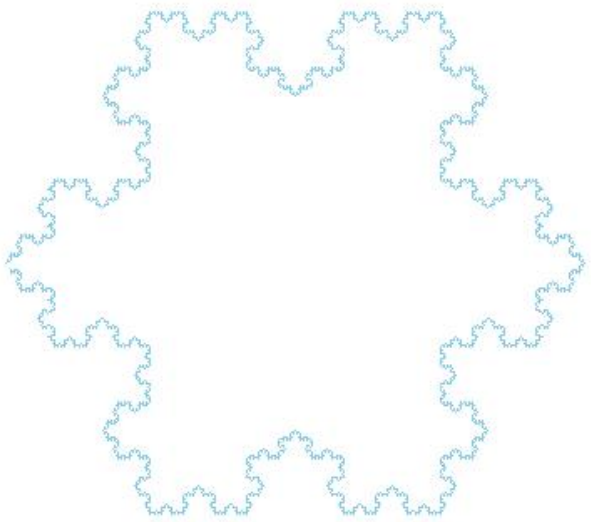
Hình fractal mô phỏng cây ở ví dụ 10

Ví dụ 11:

Dùng hình học fractal để vẽ mô phỏng bông tuyết

```
# Chương trình vẽ bông tuyết bằng hình học fractal
# Dựa trên chương trình gốc tại :
# http://python-with-science.readthedocs.io/en/latest/koch_fractal/koch_fractal.html

from turtle import *

def koch(a, order):
    if order > 0:
        for t in [60, -120, 60, 0]:
            koch(a/3., order-1)
            left(t)
    else:
        forward(a)

color("sky blue")
size = 300
order = 5

penup()
backward(size/1.732)
left(30)
pendown()

tracer(100)
hideturtle()

for i in range(3):
    koch(size, order)
    right(120)
```

```
done()
```



Hình fractal mô phỏng bông tuyết ở ví dụ 11

Phần 3 : Lập trình trò chơi với Pygame

Pygame là module cho phép lập trình trò chơi và các ứng dụng đồ họa tương tác một cách dễ dàng

1. Cài đặt Pygame

Để cài đặt Pygame bạn hãy chạy đoạn chương trình sau :

```
from pip._internal import main
main(['install', 'pygame'])
```

Sau khi chạy xong, hãy kiểm tra lại xem cài đặt của bạn đã thành công chưa với đoạn chương trình sau:

```
import pygame
print(pygame.__version__)
```

Nếu đoạn chương trình trên in ra phiên bản của pygame thì có nghĩa bạn đã cài đặt thành công.

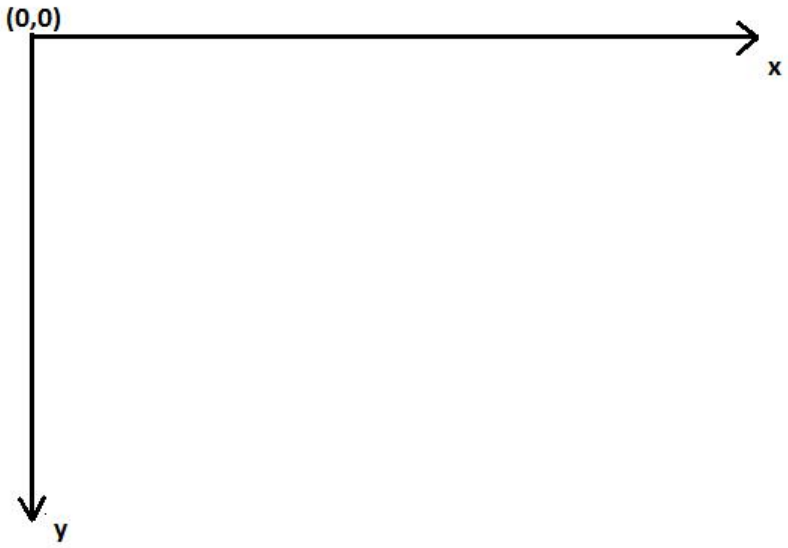
2. Cơ bản về Pygame

2.1. Một số khái niệm về đồ họa

Màn hình đồ họa

Ở chế độ đồ họa, màn hình máy tính bao gồm các điểm ảnh (pixel) nằm liền kề nhau trên một lưới chữ nhật. Kích thước màn hình (theo pixel) được gọi là độ phân giải. Ví dụ về các độ phân giải hay dùng như : 640x480, 800x600, 1024x768, 1280x1024, 1366x768 …

Mỗi điểm trên màn hình đồ họa được xác định bằng tọa độ (x,y). Tọa độ x là tọa độ theo phương ngang, tính từ trái qua phải. Tọa độ y là tọa độ theo phương thẳng đứng, tính từ trên xuống dưới. Góc tọa độ (0,0) nằm ở góc trái, phía trên của màn hình.



Tọa độ trên màn hình độ họa

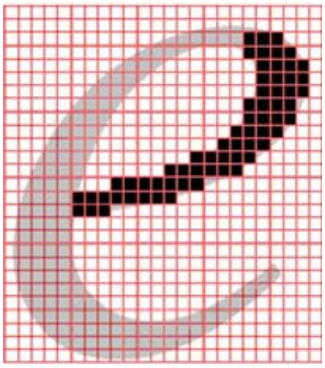
Màu sắc

Trên màn hình máy tính, mỗi điểm ảnh được tạo thành từ 3 điểm sáng có các màu : đỏ (R), xanh lá cây (G), xanh da trời (B), nằm cạnh nhau. Khi độ sáng của mỗi thành phần R,G,B thay đổi, chúng ta sẽ có các màu sắc khác nhau. Như vậy, mỗi màu sắc được thể hiện bằng 3 giá trị cường độ R, G, B. Với Pygame, và đa phần các chương trình đồ họa, khoảng giá trị cho mỗi thành phần R,G, B là từ 0 đến 255. Các màu thông dụng là:

- BLACK : (0, 0, 0)
- RED : (255, 0, 0)
- GREEN : (0, 255, 0)
- BLUE : (0, 0, 255)
- YELLOW : (255, 255, 0)
- MAGENTA : (255, 0, 255)
- CYAN: (0, 255, 255)
- WHITE : (255, 255, 255)

Ảnh bitmap

Ảnh bitmap được dùng để thể hiện hình ảnh của một đối tượng trên màn hình đồ họa. Một ảnh bitmap là một ma trận các điểm ảnh nằm liền kề nhau trong một hình chữ nhật, mỗi điểm ảnh có giá trị màu săc (R,G,B) riêng.



Ví dụ về ảnh bitmap

2.2. Bắt đầu lập trình với Pygame

Bạn hãy làm quen với Pygame bắt đầu từ ví dụ sau:

```
import pygame, sys

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Ứng dụng đồ họa trên Pygame không sử dụng toàn bộ màn hình, mà chỉ sử dụng một cửa sổ có kích thước mong muốn. Ở ví dụ trên, cửa sổ Pygame được tạo ra có kích thước 640x480 (pixel). Nếu từng chơi các game cổ điển trên DOS, bạn sẽ biết kích thước màn hình 640x480 được dùng trong rất nhiều trong các trò chơi ở những năm 90 của thế kỉ trước.

Với Pygame, việc tạo cửa sổ với kích thước mong muốn được thực hiện bằng lệnh:

```
screen = pygame.display.set_mode((screenX, screenY))
```

Sau khi tạo cửa sổ làm việc, các chương trình trò chơi thường sẽ có một vòng lặp chính (bắt đầu với lệnh "`while True`" như ở ví dụ trên) , bên trong vòng lặp, chương trình sẽ xử lý các sự kiện tương tác với người chơi (sự kiện từ chuột, bàn phím, sự kiện định kỳ theo thời gian …). Ngoài ra còn một sự kiện luôn phải xử lý, đó là sự kiện người chơi tắt ứng dụng :

```
if event.type == pygame.QUIT:
    sys.exit()
```

Nếu không xử lý sự kiện này, chương trình trò chơi của bạn sẽ không thể đóng lại được.

2.3. Vẽ hình trong Pygame:

Pygame, giống như các thư việc trò chơi khác, cung cấp các hàm vẽ đồ họa thông dụng.

Vẽ đường thẳng:

```
pygame.draw.line(screen, color, start_point, end_point, width=1)
```

Các tham số :

- screen : vùng cửa sổ vẽ, tạo ra từ hàm `pygame.display.set_mode` lúc khởi động
- color : màu của đường cần vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- start_point : điểm bắt đầu của đường thẳng, thể hiện bằng bộ 2 giá trị (x,y)
- end_point : điểm cuối của đường thẳng, thể hiện bằng bộ 2 giá trị (x,y)
- width : độ dày đường vẽ tính theo pixel, giá trị mặc định là 1

Vẽ đường gấp khúc:

```
pygame.draw.lines(screen, color, closed, pointlist, width=1)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- closed : True/False, vẽ đường đóng (nối điểm cuối với điểm đầu) hay mở
- pointlist : danh sách các điểm, mỗi điểm thể hiện bằng một bộ 2 giá trị (x,y)
- width : độ dày đường vẽ tính theo pixel, giá trị mặc định là 1

Vẽ hình chữ nhật:

```
pygame.rect(screen, color, Rect, width=0)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- Rect : bộ 4 giá trị (left,top,width,height), thể hiện tọa độ góc trái phía trên và 2 kích thước hình chữ nhật
- width : độ dày đường viền hình chữ nhật, nếu giá trị này bằng 0 thì hình chữ nhật được vẽ là hình đặc.

Vẽ đa giác:

```
pygame.draw.polygon(screen, color, pointlist, width=0)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- pointlist : danh sách tạo độ các đỉnh của đa giác
- width : độ dày đường viền đa giác, nếu giá trị này bằng 0 thì đa giác được vẽ là hình đặc.

Vẽ hình tròn:

```
pygame.circle(screen, color, pos, radius, width=0)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- pos: tọa độ (x,y) tâm hình tròn
- radius : bán kính hình tròn
- width : độ dày đường viền, nếu giá trị này bằng 0 thì hình tròn được vẽ là hình đặc.

Vẽ hình ellipse:

```
pygame.ellipse(screen, color, Rect, width=0)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- Rect: bộ 4 giá trị (left, top, width, height) của hình chữ nhật bao ellipse
- width : độ dày đường viền, nếu giá trị này bằng 0 thì ellipse được vẽ là hình đặc.

Vẽ cung đường tròn:

```
pygame.arc(screen, color, Rect, start_angle, stop_angle, width=1)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu vẽ, thể hiện bằng bộ 3 giá trị (R,G,B)
- Rect: bộ 4 giá trị (left, top, width, height) của hình chữ nhật bao đường tròn
- start_angle: góc bắt đầu của cung (hệ Radian)
- stop_angle : góc kết thúc của cung (hệ Radian)
- width : độ dày đường vẽ, giá trị mặc định là 1

Xóa màn hình:

```
screen.fill(color)
```

Các tham số:

- screen : vùng cửa sổ vẽ
- color : màu nền, thể hiện bằng bộ 3 giá trị (R,G,B)

Lưu ý:

Tất cả các lệnh vẽ của Pygame đều được thực hiện trên bộ đệm, để đưa những gì đã được vẽ ra màn hình, chúng ta cần sử dụng lệnh:

```
pygame.display.flip()
```

Ví dụ 2.3.1:

Vẽ ngôi nhà.

```
import pygame, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0
YELLOW = 255, 255, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

screen.fill(WHITE)
```

```
pygame.draw.polygon(screen, YELLOW, [
    (50,100),
    (100, 50),
    (250, 50),
    (300, 100)
])

pygame.draw.rect(screen, GREEN, pygame.Rect(100, 100, 150, 100))

pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Hình ngôi nhà trong ví dụ 2.3.1

Ví dụ 2.3.2:

Vẽ biểu tượng khuôn mặt cười .

```
import pygame, sys

BLACK = 0, 0, 0
WHITE = 255, 255, 255
YELLOW = 255, 255, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))
screen.fill(WHITE)

# Face
pygame.draw.circle(screen, YELLOW, (320, 240),  70, 0)
pygame.draw.circle(screen, BLACK, (320, 240),  70, 2)

# Eyes
pygame.draw.circle(screen, BLACK, (295, 220),  10)
pygame.draw.circle(screen, BLACK, (345, 220),  10)

# Mouth
rect = pygame.Rect(270, 190, 100, 100)
pygame.draw.arc(screen, BLACK, rect, -3, -0.1, 3)
```

```
pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Biểu tượng khuôn mặt cười trong ví dụ 2.3.2

2.4. Vẽ ảnh trong Pygame

Ảnh trong Pygame (và đồ họa nói chung) được biểu diễn bằng 1 bitmap, đặc trưng bởi 2 kích thước dài (width) và rộng (height)

Để tải ảnh từ một file ảnh từ đĩa cứng, pygame sử dụng lệnh:

```
image = pygame.image.load('tên file ảnh')
```

Ảnh được tạo ra sẽ có kích thước bằng kích thước của ảnh trong file gốc, thông thường kích thước này không bằng với kích thước chúng ta mong muốn vẽ lên cửa sổ pygame. Để đưa ảnh về kích thước mong muốn, chúng ta dùng lệnh :

```
image = pygame.transform.scale(image, (new_width, new_height))
```

Trong đó (new_width, new_height) là kích thước mong muốn của ảnh.

Để vẽ một ảnh lên cửa sổ pygame, chúng ta dùng lệnh:

```
screen.blit(image, Rect)
```

Trong đó `Rect` là hình chữ nhật chứa vùng chúng ta muốn đặt ảnh vào.

Ví dụ 2.4.1:

Trong thư mục của chương trình chạy có file `ball.jpg` chứa ảnh một quả bóng. Hãy viết chương trình tải ảnh từ file và vẽ lên màn hình.

```
import pygame, sys
import math

WHITE = 255, 255, 255

screenX, screenY = 640, 480
```

```
screen = pygame.display.set_mode((screenX, screenY))
screen.fill(WHITE)

ball_size = 80
x = (screenX - ball_size)/2
y = (screenY - ball_size)/2

image = pygame.image.load('ball.jpg')
image = pygame.transform.scale(image, (ball_size, ball_size))
screen.blit(image, pygame.Rect(x, y, ball_size, ball_size))

pygame.display.flip()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Tải ảnh quả bóng từ file và vẽ lên cửa sổ pygame

2.5. Tạo ảnh chuyển động

Để tạo ảnh chuyển động, chúng ta dùng một đồng hồ định thời để vẽ liên tiếp các frame theo thời gian. Số frame vẽ trong một giây phải từ 25 trở lên để ảnh không bị giật.

Cách sử dụng đồng hồ định thời để tạo ra ảnh chuyển động trong pygame như sau:

```
frame_rate = 50  # Số lần vẽ trong 1 giây
clock = pygame.time.Clock()

while True:
    clock.tick(frame_rate )

    # Thực hiện vẽ

    pygame.display.flip()

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Ví dụ 2.5.1:

Vẽ quả bóng chuyển động từ trên xuống dưới màn hình, khi chạm tới biên dưới cửa sổ thì bật ngược trở lại và chuyển động lên trên, khi tới biên trên cửa sổ thì lại chuyển động xuống dưới.

```
import pygame, sys

WHITE = 255, 255, 255
RED = 255, 0, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

radius = 40
x = screenX/2
y = radius
vy = 1

clock = pygame.time.Clock()

while True:
    clock.tick(100)

    screen.fill(WHITE)
    pygame.draw.circle(screen, RED, (int(x), int(y)), radius)
    pygame.display.flip()

    y += vy

    if (y < radius) or (y + radius > screenY):
        vy = -vy

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Thực hiện chạy chương trình ở ví dụ trên, bạn sẽ thấy quả bóng màu đỏ chuyển động tuần hoàn theo phương thẳng đứng. Chương trình đặt định kì vẽ 100 lần trong một giây, tại mỗi lần vẽ, tọa độ y của quả bóng được cộng thêm một lượng thay đổi (bằng vy , tốc độ chuyển động). Khi tọa độ y chạm tới biên trên hoặc biên dưới của cửa sổ thì chiều chuyển động được đảo ngược (đổi dấu vận tốc chuyển động)

Ví dụ 2.5.2:

Viết chương trình mô phỏng quả bóng rơi tự do trong trọng trường, khi rơi xuống sàn thì nảy lên, mỗi lần va chạm với sàn, vận tốc quả bóng giảm đi 10%.

```
import pygame, sys

WHITE = 255, 255, 255
RED = 255, 0, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

radius = 40
x = screenX/2
y = radius
```

```
vy = 0
a = 0.05

clock = pygame.time.Clock()

while True:
    clock.tick(100)

    screen.fill(WHITE)
    pygame.draw.circle(screen, RED, (int(x), int(y)), radius)
    pygame.display.flip()

    vy += a
    y += vy

    if y + radius > screenY:
        y = screenY - radius
        vy = -0.9 * vy

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
```

Quan sát chương trình ở ví dụ 5, bạn sẽ thấy nó chỉ khác với ví dụ 4 một chút:

- Ở mỗi bước vẽ, không chỉ tọa độ y mà cả vận tốc vy cũng được cập nhật. Giá trị a thể hiện độ lớn của gia tốc trọng trường:

```
vy += a
y += vy
```

- Khi quả bóng chạm tới sàn thì vận tốc đổi chiều, đồng thời độ lớn giảm còn bằng 0.9 giá trị cũ :

```
if y + radius > screenY:
    y = screenY - radius
    vy = -0.9 * vy
```

2.6. Tương tác với người chơi qua bàn phím

Để tương tác với người chơi qua bàn phìm, tại mỗi bước trong vòng lặp chính, chương trình cần kiểm tra các sự kiện từ bàn phím. Hai sự kiện cơ bản là phím được nhấn (pygame.KEYDOWN) và phím được nhả (pygame.KEYUP)

Ví dụ 2.6.1:

Viết chương trình kiểm tra xem người chơi nhấn và nhả phím nào. In ra màn hình mã của phím được nhấn/nhả.

```
import pygame, sys

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

while True:
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.KEYDOWN:
            print('Phím có mã ', event.key, ' được nhấn')

        if event.type == pygame.KEYUP:
            print('Phím có mã ', event.key, ' được nhả')
```

Khi chạy chương trình trên, quan sát ở cử sổ console, bạn sẽ thấy mỗi khi một phím được nhấn hay nhả thì mã của phím đó lại được in ra màn hình. Các phím hay được dùng trong các trò chơi :

- Phím 'A' đến 'Z' : tương ứng với mã từ 97 (hay `ord('a')`) đến 122 (hay `ord('z')`)
- Phím '0' đến '9' : tương ứng với mã từ 48 (hay `ord('0')`) đến 57 (hay `ord('9')`)
- Phím mũi tên lên : mã 273 (hay `pygame.K_UP`)
- Phím mũi tên xuống : mã 274 (hay `pygame.K_DOWN`)
- Phím mũi tên phải : mã 275 (hay `pygame.K_RIGHT`)
- Phím mũi tên trái : mã 276 (hay `pygame.K_LEFT`)
- Phím SPACE : mã 32 (hay `pygame.SPACE`)
- Phím SHIFT phải : mã 303 (hay `pygame.K_RSHIFT`)
- Phím SHIFT trái : mã 304 (hay `pygame.K_LSHIFT`)
- Phím CTRL phải : mã 305 (hay `pygame.K_RCTRL`)
- Phím CTRL trái : mã 306 (hay `pygame.K_LCTRL`)
- Phím ALT phải : mã 307 (hay `pygame.K_RALT`)
- Phím ALT trái : mã 308 (hay `pygame.K_LALT`)

Ví dụ 2.6.2:

Viết chương trình vẽ quả bóng trên màn hình, dùng 4 phím mũi tên để di chuyển quả bóng đến các vị trị khác nhau.

```
import pygame, sys

WHITE = 255, 255, 255
RED = 255, 0, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

radius = 40
x, y = screenX/2, screenY/2
vx, vy = 0, 0

clock = pygame.time.Clock()

while True:
    clock.tick(100)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                vy = -1
```

```
            if event.key == pygame.K_DOWN:
                vy = 1

            if event.key == pygame.K_LEFT:
                vx = -1

            if event.key == pygame.K_RIGHT:
                vx = 1

        if event.type == pygame.KEYUP:
            vx, vy = 0, 0

    x += vx
    y += vy

    x = min(max(x, radius), screenX-radius)
    y = min(max(y, radius), screenY-radius)

    screen.fill(WHITE)
    pygame.draw.circle(screen, RED, (int(x), int(y)), radius)
    pygame.display.flip()
```

2.7. Tương tác với người chơi qua chuột

Để tương tác với người chơi qua chuột, trong vòng lặp chính, chương trình cần kiểm tra các sự kiện về chuột :

- `pygame.MOUSEBUTTONDOWN` : chuột được nhấn
- `pygame.MOUSEBUTTONUP` : chuột được nhả
- `pygame.MOUSEMOTION` : chuột di chuyển

Ngoài ra, muốn biết tọa độ hiện tại của chuột, chúng ta dùng hàm:

```
pygame.mouse.get_pos()
```

Hàm này trả về tọa độ của chuột dưới dạng một bộ 2 giá trị (x,y)

Ví dụ 2.7.1:

Viết chương trình vẽ quả bóng trên màn hình, dùng chuột để kéo quả bóng đến các vị trí khác nhau.

```
import pygame, sys

WHITE = 255, 255, 255
RED = 255, 0, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

radius = 40
x, y = screenX/2, screenY/2

moving = False
clock = pygame.time.Clock()

while True:
```

```
clock.tick(100)

screen.fill(WHITE)
pygame.draw.circle(screen, RED, (int(x), int(y)), radius)
pygame.display.flip()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        sys.exit()

    if event.type == pygame.MOUSEBUTTONDOWN:
        mouse_x, mouse_y = pygame.mouse.get_pos()
        dx = mouse_x - x
        dy = mouse_y - y

        if dx*dx + dy*dy < radius*radius:
            moving = True

    if event.type == pygame.MOUSEMOTION and moving:
        x, y = pygame.mouse.get_pos()

    if event.type == pygame.MOUSEBUTTONUP:
        moving = False
```

3. Trò chơi Flappy Bird

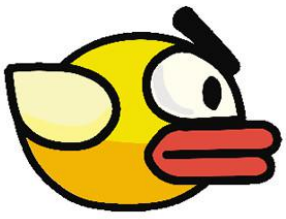
Trò chơi Flappy Bird của tác giả Nguyễn Hà Đông đã từng rất nổi tiếng trên điện thoại vào năm 2013. Trong trò chơi này, người chơi phải điều khiển một chú chim bay qua các cột ống nước mà không chạm vào thành ống.

Trong phần này, chúng ta sẽ tìm hiểu cách viết một trò chơi tương tự bằng Pygame. Để hiểu được nguyên lý viết trò chơi, chúng ta chỉ tập trung vào các chi tiết chính và sẽ đơn giản hóa nhiều phần.

Chuẩn bị ảnh:

Hầu hết các trò chơi đều cần có ảnh của các nhân vật và các đối tượng trong trò chơi. Với trò chơi Flappy Bird chúng ta sẽ xây dựng trong phần này, bạn chỉ cần có ảnh của chú chim, còn các cột ống nước sẽ được vẽ bằng các thanh hình chữ nhật.

Bạn có thể tìm ảnh của Flappy Bird bằng công cụ tìm kiếm hình ảnh của google. Hãy chọn ảnh có kích thước hình vuông, hoặc 2 chiều kích thước gần bằng nhau, vì trong chương trình chúng ta sẽ vẽ chú chim vào một vùng có kích thước 40x40 pixel



Một ảnh Flappy Bird lấy từ công cụ tìm kiếm hình ảnh của google.

Sau khi đã có ảnh chú chim, bạn hãy đổi tên file ảnh thành **bird.jpg** và đặt trong cùng thư mục với file chạy chương trình.

Bước 1: Vẽ chú chim lên chính giữa màn hình.

```
import sys, pygame

WHITE = 255, 255, 255
screenSize = screenX, screenY = 640, 480
screen = pygame.display.set_mode(screenSize)

bird_size = 40
image = pygame.image.load("bird.jpg")
image = pygame.transform.scale(image, (bird_size, bird_size))

x = (screenX - bird_size)/2
y = (screenY - bird_size)/2

clock = pygame.time.Clock()

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    screen.fill(WHITE)
    screen.blit(image, pygame.Rect(x, y, bird_size, bird_size))
    pygame.display.flip()
```

Vẽ chú chim tại chính giữa màn hình

Bước 2 : Mô phỏng chuyển động rơi của chú chim trong trọng trường.

Bạn hãy xem lại ví dụ 2.5.2 trong chương 2, mô phỏng chuyển động rơi của quả bóng trong trọng trường. Xem lại chương trình đó, bạn sẽ thấy chúng ta cần các biến để lưu tọa độ theo phương y, vận tốc theo phương y và gia tốc trọng trường. Tại mỗi bước trong vòng lặp chính, chúng ta cần cập nhật lại các giá trị vận tốc và tọa độ theo phương y cho chú chim.

Gia tốc trọng trường có giá trị hướng xuống dưới, tuy nhiên khi người dùng click chuột, chúng ta thay đổi chiều gia tốc hướng lên trên, tương đương với việc người chơi dùng lực để nâng chú chim lên.

```
import sys, pygame

WHITE = 255, 255, 255
screenSize = screenX, screenY = 640, 480
screen = pygame.display.set_mode(screenSize)

bird_size = 40
image = pygame.image.load("bird.jpg")
image = pygame.transform.scale(image, (bird_size, bird_size))

x = (screenX - bird_size)/2
y = (screenY - bird_size)/2
vy, a = 0, 0.05

clock = pygame.time.Clock()

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            a = -0.2

        if event.type == pygame.MOUSEBUTTONUP:
            a = 0.05

    vy += a
    y += vy

    if y + bird_size > screenY:
        vy, y = 0, screenY - bird_size

    if y < 0:
        vy, y = 0, 0


    screen.fill(WHITE)
    screen.blit(image, pygame.Rect(x, y, bird_size, bird_size))
    pygame.display.flip()
```

Để tiết kiệm công soạn thảo, các phần thay đổi trong chương trình sẽ được bôi đậm, bạn hãy bổ sung các phần này vào chương trình đã có ở bước 1.

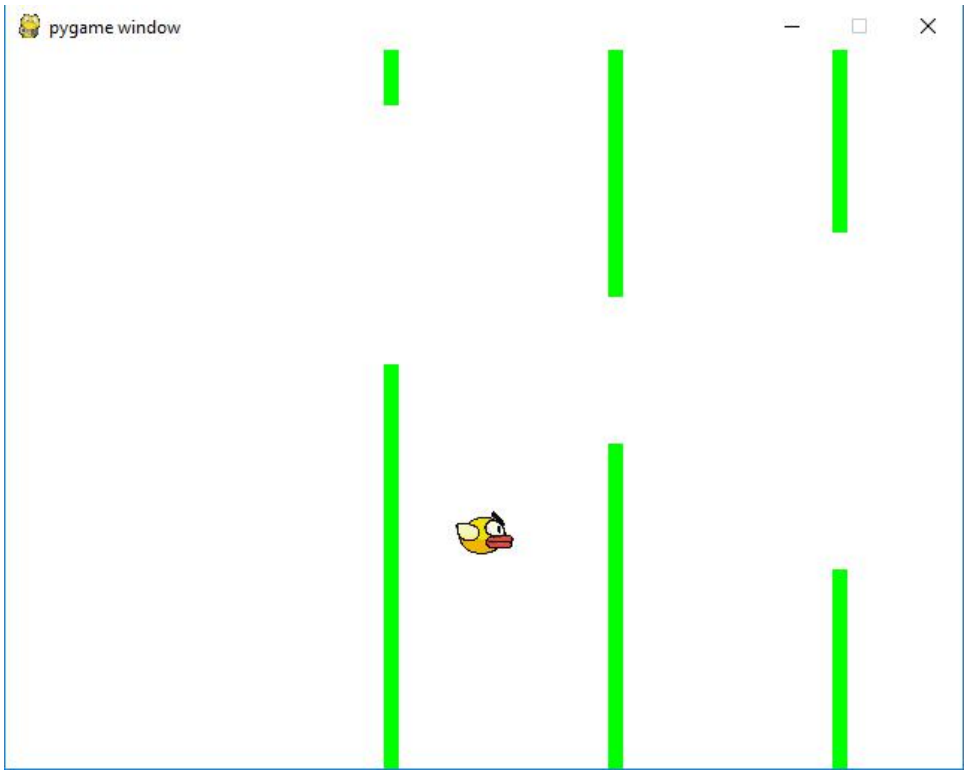
Sau khi soạn thảo chương trình xong, bạn hãy chạy chương trình và dùng chuột để điều khiển chú chim trên màn hình.

Bước 3: Tạo các ống nước.

Các ống nước được vẽ bằng các thanh chữ nhật. Mỗi cặp ống nước gồm 2 nửa : nửa trên và nửa dưới, nhiệm vụ của chú chim là phải bay qua khe hở giữa 2 nửa của từng cặp ống. Các cặp ống xuất hiện từ phía ngoài cùng bên phải màn hình và chuyển động dần sang trái, tạo ra cảm giác chú chim đang bay sang phải.

```
import sys, pygame, random

WHITE = 255, 255, 255
screenSize = screenX, screenY = 640, 480
screen = pygame.display.set_mode(screenSize)

bird_size = 40
image = pygame.image.load("bird.jpg")
image = pygame.transform.scale(image, (bird_size, bird_size))

x = (screenX - bird_size)/2
y = (screenY - bird_size)/2
vy, a = 0, 0.05

pipes = []
frameNo = 0

clock = pygame.time.Clock()

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            a = -0.2

        if event.type == pygame.MOUSEBUTTONUP:
            a = 0.05

    frameNo += 1
    if frameNo % 150 == 0:
        h1 = random.randint(25, 250)
        h2 = h1 + random.randint(75, 250)
        pipes.append(pygame.Rect(screenX, 0, 10, h1))
        pipes.append(pygame.Rect(screenX, h2, 10, screenY-h2))

    vy += a
    y += vy

    if y + bird_size > screenY:
        vy, y = 0, screenY - bird_size

    if y < 0:
        vy, y = 0, 0
```

```
    screen.fill(WHITE)
    screen.blit(image, pygame.Rect(x, y, bird_size, bird_size))

    for pipe in pipes:
        pipe.left -= 1
        pygame.draw.rect(screen, (0, 255, 0), pipe)

    pipes = [pipe for pipe in pipes if pipe.left >= 0]

    pygame.display.flip()
```



Vẽ các ống nước chuyển động sang trái.

Quan sát phần thay đổi trong chương trình, bạn sẽ thấy chúng thực hiện các chức năng sau:

- Khởi tạo ngẫu nhiên các ống nước:

```
frameNo += 1
if frameNo % 150 == 0:
    h1 = random.randint(25, 250)
    h2 = h1 + random.randint(75, 250)
    pipes.append(pygame.Rect(screenX, 0, 10, h1))
    pipes.append(pygame.Rect(screenX, h2, 10, screenY-h2))
```

Biến `frameNo` là thứ tự của các bước vẽ theo thời gian (số lần vẽ là 50 lần/giây). Khi `frameNo` chia hết cho 150 (cứ 3 giây một), một cặp ống nước với 2 độ cao ngẫu nhiên h1, h2 được tạo ra và bổ sung vào danh sách các ống nước.

- Cho các ống nước chuyển động dần dần sang trái:

```
for pipe in pipes:
```

```
            pipe.left -= 1
            pygame.draw.rect(screen, (0, 255, 0), pipe)

        pipes = [pipe for pipe in pipes if pipe.left >= 0]
```

Mỗi bước vẽ, các ống nước lại được dịch sang trái 1 pixel. Các ống nước đã chuyển động hết sang trái và ra khỏi màn hình sẽ được bỏ ra khỏi danh sách ống nước.

Chạy đoạn chương trình trên, bạn sẽ thấy các ống nước chuyển động xuyên qua chú chim. Chúng ta sẽ xử lý việc kiểm tra va chạm giữa chú chim và các ống nước ở bước sau.

Bước 4 : Kiểm tra va chạm giữa chú chim và ống nước

Nếu người chơi điều khiển chú chim để chạm vào các ống nước thì chúng ta thông báo kết thúc trò chơi.

```
import sys, pygame, random

WHITE = 255, 255, 255
screenSize = screenX, screenY = 640, 480
screen = pygame.display.set_mode(screenSize)

pygame.font.init()
font = pygame.font.SysFont('Calibri', 36, bold=True)

bird_size = 40
image = pygame.image.load("bird.jpg")
image = pygame.transform.scale(image, (bird_size, bird_size))

x = (screenX - bird_size)/2
y = (screenY - bird_size)/2
vy, a = 0, 0.05

pipes = []
frameNo = 0

finished = False
clock = pygame.time.Clock()

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            a = -0.2

        if event.type == pygame.MOUSEBUTTONUP:
            a = 0.05

    if finished:
        continue

    frameNo += 1
```

```
        if frameNo % 150 == 0:
            h1 = random.randint(25, 250)
            h2 = h1 + random.randint(75, 250)
            pipes.append(pygame.Rect(screenX, 0, 10, h1))
            pipes.append(pygame.Rect(screenX, h2, 10, screenY-h2))

        vy += a
        y += vy

        if y + bird_size > screenY:
            vy, y = 0, screenY - bird_size

        if y < 0:
            vy, y = 0, 0

        screen.fill(WHITE)
        screen.blit(image, pygame.Rect(x, y, bird_size, bird_size))

        bird = pygame.Rect(x+2, y+2, bird_size-4, bird_size-4)

        for pipe in pipes:
            pipe.left -= 1
            pygame.draw.rect(screen, (0, 255, 0), pipe)

            if bird.colliderect(pipe):
                finished = True

        pipes = [pipe for pipe in pipes if pipe.left >= 0]

        if finished:
            text = font.render('Game Over!', False, (255, 0, 0))
            screen.blit(text, (280, 200))

        pygame.display.flip()
```

Việc kiểm tra va chạm giữa chú chim và các ống nước được thực hiện qua lệnh:

```
            if bird.colliderect(pipe):
                finished = True
```

Khi trò chơi kết thúc, chúng ta đưa ra thông báo ở giữa màn hình:

```
        if finished:
            text = font.render('Game Over!', False, (255, 0, 0))
            screen.blit(text, (280, 220))
```

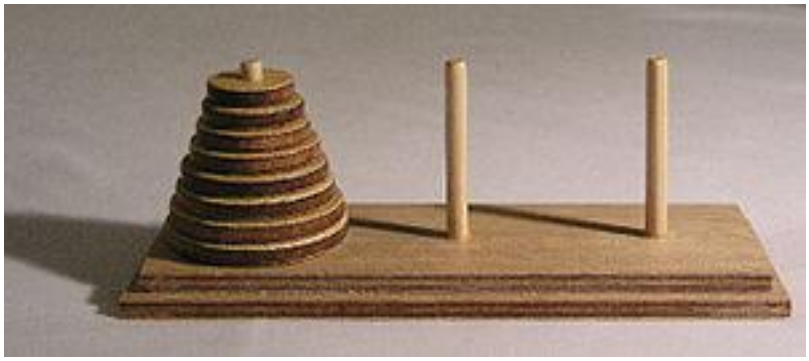
Mở rộng:

Bạn hãy thêm phần đo khoảng cách chú chim đã bay được và thông báo điểm số này cho người chơi sau mỗi lần chơi.

4. Trò chơi tháp Hà Nội

Bài toán Tháp Hà Nội do nhà toán học người Pháp Édouard Lucas đặt ra vào năm 1883 khi quan sát các tầng của tháp rùa Hà Nội.

Trong bài toán này, chúng ta có 3 cọc và N đĩa với độ rộng khác nhau. Ban đầu toàn bộ N đĩa nằm ở cọc số 1, các đĩa to nằm dưới đĩa con. Bạn cần dùng cọc trung gian số 2 để chuyển N đĩa sang cọc thứ 3. Trong quá trình chuyển, các đĩa to không được đặt lên trên đĩa con.



Minh họa bài toán tháp Hà Nội (nguồn : wikipedia)

Trong chương này chúng ta sẽ xây dựng trò chơi dựa trên bài toán tháp Hà Nội. Trò chơi sẽ yêu cầu người chơi chuyển các đĩa từ cọc số 1 sang cọc số 3, sử dụng cọc số 2 làm trung gian.

4.1. Lời giải của bài toán tháp Hà Nội

Bài toán tháp Hà Nội được giải bằng phương pháp đệ quy. Đệ quy là cách lập trình trong đó một hàm gọi chính bản thân nó nhưng với giá trị tham số truyền vào nhỏ hơn so với ban đầu. Một ví dụ về lập trình đệ quy là hàm tính giai thừa một số tự nhiên:

```
def factorial(n):
    if n <= 0:
        return 1

    return n*factorial(n-1)
```

Ý nghĩa của phương pháp đệ quy là ban đầu chúng ta tìm cách giải bài toán ở quy mô nhỏ nhất (N=0, N= 1, …), sau đó với các giá trị tham số lớn hơn, chúng ta tìm cách đưa bài toán về quy mô nhỏ hơn đã được giải quyết.

Sử dụng phương pháp đệ quy chúng ta giải bài toán tháp Hà Nội như sau:

- N = 1: có 1 đĩa, để chuyển từ cọc 1 sang cọc 3 chúng ta di chuyển trực tiếp, không cần trung gian
- N > 1 : chúng ta tìm cách chuyển N-1 đĩa trên cùng từ cọc 1 sang cọc 2, sử dụng cọc 3 làm trung gian. Sau khi chuyển xong, chúng ta chuyển đĩa còn lại ở cọc 1 sang cọc 3. Cuối cùng chuyển N-1 đĩa từ cọc 2 sang cọc 3, sử dụng cọc 1 là trung gian.

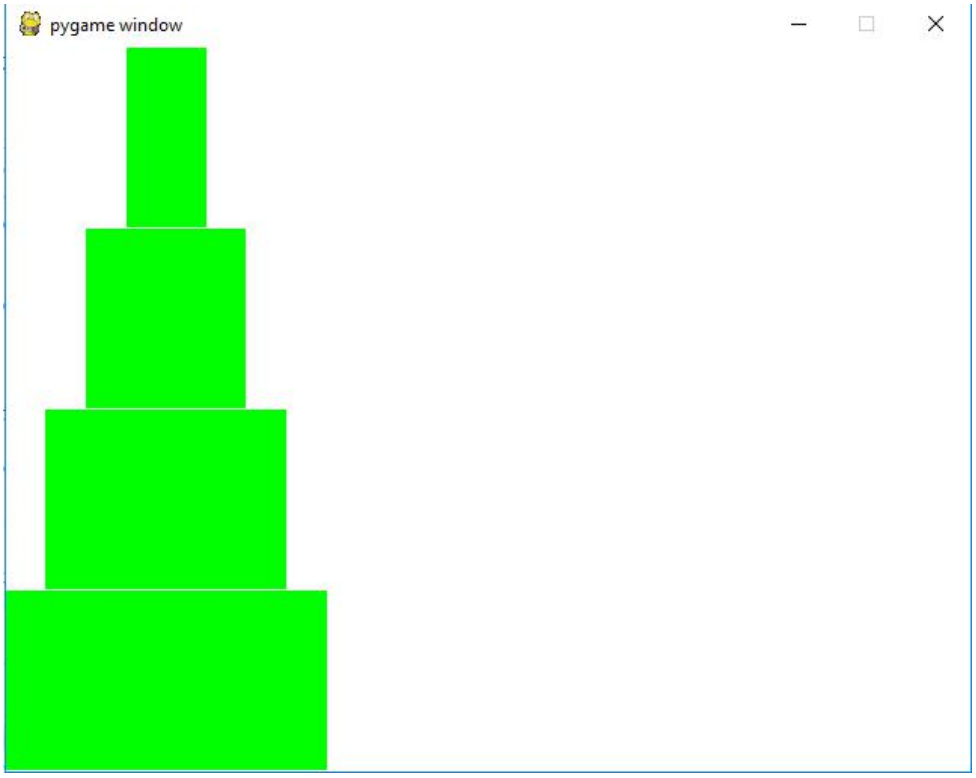
Chương trình giải bài toán tháp Hà Nội:

```
def hanoi(src, dest, N):
    if N > 0:
        hanoi(src, 3-src-dest, N-1)
        print('Chuyển đĩa từ cọc ', src+1, ' sang cọc ', dest+1)
        hanoi(3-src-dest, dest, N-1)

N = 4
hanoi(0, 2, N)
```

Để hiểu hơn về cách giải bài toán tháp Hà Nội, chúng ta hãy xây dựng chương trình mô phỏng bằng Pygame.

Bước 1: Vẽ N đĩa ở cọc 1

```
# Chương trình minh họa bài toán tháp Hà Nội
# Dựa trên chương trình gốc tại :
#      http://python3.codes/animated-tower-of-hanoi

import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480

column_width = screenX/3
disk_height = screenY/N

disks = [[], [], []]

screen = pygame.display.set_mode((screenX, screenY))
screen.fill(WHITE)

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

for i in range(N):
    disk = {'width' : column_width * (N - i) / N}
    pushDisk(disk, 0)
    pygame.draw.rect(screen, GREEN, getRect(disk))

pygame.display.flip()
time.sleep(3)
```



Vẽ N đĩa ở cọc số 1

Quan sát chương trình, bạn sẽ thấy mỗi đĩa được vẽ bằng một hình chữ nhật. Một số chi tiết về các biến và hàm trong chương trình:

- Biến `disks` : một danh sách gồm 3 phần tử, mỗi phần tử lại là một danh sách chứa các đĩa ở các cọc 1, 2, 3 tương ứng. Mỗi đĩa là một đối tượng với các thông tin:
  - `'width'` : độ rộng của đĩa, giá trị này không thay đổi sau khi đĩa được tạo ra
  - `'column'` : cọc hiện tại của đĩa, giá trị này thay đổi khi đĩa được di chuyển
  - `'level'` : thứ tự của đĩa ở cọc đĩa đang nằm, đĩa nằm dưới cùng có thứ tự bằng 1, các đĩa nằm trên có thứ tự tăng dần
- Hàm `getRect` : lấy hình chữ nhật bao của một đĩa
- Hàm `pushDisk` : đặt một đĩa vào một cọc
- Hàm `popDisk` : lấy đĩa trên cùng của một cọc ra khỏi cọc đó

Bước 2 : Mô phỏng quá trình chuyển đĩa từ cọc này sang cọc khác

```
# Chương trình minh họa bài toán tháp Hà Nội
# Dựa trên chương trình gốc tại :
#      http://python3.codes/animated-tower-of-hanoi

import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480
```

```
column_width = screenX/3
disk_height = screenY/N

disks = [[], [], []]

screen = pygame.display.set_mode((screenX, screenY))
screen.fill(WHITE)

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

def animateMove(old_rect, new_rect):
    pygame.draw.rect(screen, WHITE, old_rect)
    pygame.draw.rect(screen, GREEN, new_rect)
    pygame.display.flip()
    time.sleep(0.5)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

def moveDisk(disk, column):
    popDisk(disk['column'])

    rect1 = getRect(disk)
    rect2 = pygame.Rect(rect1)
    rect2.top = 0

    pushDisk(disk, column)
    rect4 = getRect(disk)
    rect3 = pygame.Rect(rect4)
    rect3.top = 0

    animateMove(rect1, rect2)
    animateMove(rect2, rect3)
    animateMove(rect3, rect4)

for i in range(N):
    disk = {'width' : column_width * (N - i) / N}
    pushDisk(disk, 0)
    pygame.draw.rect(screen, GREEN, getRect(disk))

pygame.display.flip()

time.sleep(1)
moveDisk(disks[0][-1], 2)
time.sleep(3)
```

Đoạn chương trình trên mô phỏng quá trình chuyển một đĩa từ cọc này sang cọc khác. Bên trong hàm `moveDisk`, chúng ta sẽ thấy quá trình di chuyển đĩa gồm 3 bước:

```
animateMove(rect1, rect2)
animateMove(rect2, rect3)
animateMove(rect3, rect4)
```

Các bước này tương ứng với các hành động:

- Nhấc đĩa lên khỏi cọc ban đầu (di chuyển thẳng lên trên)
- Di chuyển sang cọc mới (phương ngang)
- Đặt đĩa xuống cọc mới (di chuyển thẳng xuống dưới)

Bước 3 : Hoàn thành chương trình mô phỏng tháp Hà Nội

Sau khi đã có các hàm vẽ, chúng ta ghép với lời giải bài toán tháp Hà Nội bằng phương pháp đệ quy để có chương trình mô phỏng đầy đủ.

```
# Chương trình minh họa bài toán tháp Hà Nội
# Dựa trên chương trình gốc tại :
#      http://python3.codes/animated-tower-of-hanoi

import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480

column_width = screenX/3
disk_height = screenY/N

disks = [[], [], []]

screen = pygame.display.set_mode((screenX, screenY))
screen.fill(WHITE)

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

def animateMove(old_rect, new_rect):
    pygame.draw.rect(screen, WHITE, old_rect)
    pygame.draw.rect(screen, GREEN, new_rect)
    pygame.display.flip()
```

```
        time.sleep(0.5)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

    def moveDisk(disk, column):
        popDisk(disk['column'])

        rect1 = getRect(disk)
        rect2 = pygame.Rect(rect1)
        rect2.top = 0

        pushDisk(disk, column)
        rect4 = getRect(disk)
        rect3 = pygame.Rect(rect4)
        rect3.top = 0

        animateMove(rect1, rect2)
        animateMove(rect2, rect3)
        animateMove(rect3, rect4)

    for i in range(N):
        disk = {'width' : column_width * (N - i) / N}
        pushDisk(disk, 0)
        pygame.draw.rect(screen, GREEN, getRect(disk))

    pygame.display.flip()

    def hanoi(src, dest, N):
        if N > 0:
            hanoi(src, 3-src-dest, N-1)
            moveDisk(disks[src][-1], dest)
            hanoi(3-src-dest, dest, N-1)

    hanoi(0, 2, N)
```

4.2. Xây dựng trò chơi dựa trên bài toán tháp Hà Nội

Trong phần này, chúng ta xây dựng trò chơi cho phép người chơi dùng chuột để chuyển các đĩa qua các cọc. Khi người chơi chuyển toàn bộ N đĩa từ cọc 1 sang cọc 3 thì trò chơi kết thúc.

Bước 1: Vẽ N đĩa ở cọc số 1

Tương tự như ở phần mô phỏng, đầu tiên chúng ta vẽ N đĩa ở cọc số 1

```
import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480

column_width = screenX/3
disk_height = screenY/N
```

```
disks = [[], [], []]

screen = pygame.display.set_mode((screenX, screenY))

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

for i in range(N):
    disk = {'width' : column_width * (N - i) / N}
    pushDisk(disk, 0)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    screen.fill(WHITE)

    for i in range(3):
        for disk in disks[i]:
            pygame.draw.rect(screen, GREEN, getRect(disk))

    pygame.display.flip()

    time.sleep(0.01)
```

Bước 2: Cho phép người chơi dùng chuột để di chuyển các đĩa ở trên cùng của các cọc

Việc dùng chuột để di chuyển các đĩa tương tự như ví dụ 2.7.1 trong chương 2, trong đó người chơi dùng chuột để di chuyển quả bóng. Việc này được thực hiện nhờ việc kiểm tra các sự kiện về chuột : `pygame.MOUSEBUTTONDOWN` (chuột nhấn), `pygame.MOUSEBUTTONUP` (chuột nhả), `pygame.MOUSEMOTION` (chuột chuyển động) :

```
import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480

column_width = screenX/3
disk_height = screenY/N

disks = [[], [], []]
```

```
screen = pygame.display.set_mode((screenX, screenY))

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

for i in range(N):
    disk = {'width' : column_width * (N - i) / N}
    pushDisk(disk, 0)

move_disk = None

while True:
    mouse_pos = pygame.mouse.get_pos()

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            for i in range(3):
                if len(disks[i]) == 0 :
                    continue

                top_disk = disks[i][-1]
                top_disk_rect = getRect(top_disk)

                if top_disk_rect.collidepoint(mouse_pos):
                    move_disk = top_disk
                    move_rect = top_disk_rect
                    break

        if event.type == pygame.MOUSEMOTION and move_disk != None:
            move_rect.topleft = mouse_pos

        if event.type == pygame.MOUSEBUTTONUP and move_disk != None:
            move_disk = None

    screen.fill(WHITE)

    for i in range(3):
        for disk in disks[i]:
            if disk != move_disk:
                pygame.draw.rect(screen, GREEN, getRect(disk))

    if move_disk != None:
        pygame.draw.rect(screen, GREEN, move_rect)
```

```
        pygame.display.flip()

        time.sleep(0.01)
```

Việc dùng chuột di chuyển các đĩa thực hiện theo nguyên tắc sau : mỗi khi chuột được nhấn (sự kiện pygame.MOUSEBUTTONDOWN), chương trình kiểm tra tọa độ chuột có nằm trong đĩa trên cùng của một trong 3 cọc không, nếu có thì bắt đầu đánh dấu đĩa đó là đĩa được kéo đi (move_disk):

```
            if event.type == pygame.MOUSEBUTTONDOWN:
                for i in range(3):
                    if len(disks[i]) == 0 :
                        continue

                    top_disk = disks[i][-1]
                    top_disk_rect = getRect(top_disk)

                    if top_disk_rect.collidepoint(mouse_pos):
                        move_disk = top_disk
                        move_rect = top_disk_rect
                        break
```

Khi có một đĩa đang được kéo đi thì mỗi khi chuột di chuyển (sự kiện pygame.MOUSEMOTION), chương trình sẽ cập nhật tọa độ mới cho đĩa tại vị trí mới của chuột:

```
            if event.type == pygame.MOUSEMOTION and move_disk != None:
                move_rect.topleft = mouse_pos
```

Khi chuột nhả (sự kiện pygame.MOUSEBUTTONUP) thì quá trình di chuyển đĩa kết thúc:

```
            if event.type == pygame.MOUSEBUTTONUP and move_disk != None:
                move_disk = None
```

Nếu chạy đoạn chương trình trên, bạn sẽ thấy khi chuột được nhả, đĩa đang di chuyển sẽ trở về cọc mà nó đã nằm trước khi được kéo đi. Chúng ta sẽ thêm phần xử lý cập nhật cọc mới cho đĩa sau khi di chuyển xong ở bước tiếp.

Bước 3 : Hoàn thành trò chơi.

```
import pygame, time, sys

WHITE = 255, 255, 255
GREEN = 0, 255, 0

N = 4
screenX, screenY = 640, 480

column_width = screenX/3
disk_height = screenY/N

disks = [[], [], []]

screen = pygame.display.set_mode((screenX, screenY))

pygame.font.init()
font = pygame.font.SysFont('Calibri', 36, bold=True)

def getRect(disk):
    xcenter = (disk['column'] + 0.5) * column_width
```

```
    left = xcenter - disk['width']/2
    top = screenY - disk_height * disk['level']
    return pygame.Rect(left, top, disk['width'], disk_height-1)

def canPush(disk, column):
    if len(disks[column]) == 0:
        return True

    top_disk = disks[column][-1]
    return top_disk['width'] > disk['width']

def pushDisk(disk, column):
    disk['column'] = column
    disk['level'] = len(disks[column]) + 1
    disks[column].append(disk)

def popDisk(column):
    disks[column].pop()

for i in range(N):
    disk = {'width' : column_width * (N - i) / N}
    pushDisk(disk, 0)

move_disk = None

while True:
    mouse_pos = pygame.mouse.get_pos()

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            for i in range(3):
                if len(disks[i]) == 0 :
                    continue

                top_disk = disks[i][-1]
                top_disk_rect = getRect(top_disk)

                if top_disk_rect.collidepoint(mouse_pos):
                    move_disk = top_disk
                    move_rect = top_disk_rect
                    break

        if event.type == pygame.MOUSEMOTION and move_disk != None:
            move_rect.topleft = mouse_pos

        if event.type == pygame.MOUSEBUTTONUP and move_disk != None:
            column = int(mouse_pos[0]/column_width)

            if canPush(move_disk, column):
                popDisk(move_disk['column'])
                pushDisk(move_disk, column)

            move_disk = None

    screen.fill(WHITE)
```

```
        for i in range(3):
            for disk in disks[i]:
                if disk != move_disk:
                    pygame.draw.rect(screen, GREEN, getRect(disk))

        if move_disk != None:
            pygame.draw.rect(screen, GREEN, move_rect)

        if len(disks[2]) == N:
            text = font.render('Well done!', False, (255, 0, 0))
            screen.blit(text, (280, 200))

        pygame.display.flip()

        time.sleep(0.01)
```

So sánh với chương trình ở bước 2, chúng ta thấy ở chương trình này có thêm phần cập nhật cọc mới cho đĩa sau khi di chuyển. Khi một đĩa được kéo sang vị trí của một cọc mới, chương trình sẽ kiểm tra xem nó có thể đặt lên đầu cọc đó không. Điều kiện đặt được là đĩa đó phải nhỏ hơn đĩa trên cùng của cọc tới:

```
            column = int(mouse_pos[0]/column_width)

            if canPush(move_disk, column):
                popDisk(move_disk['column'])
                pushDisk(move_disk, column)
```

Ở cuối chương trình, có đoạn kiểm tra xem người chơi đã chuyển xong hết các đĩa chưa. Điều kiện để trò chơi hoàn thành là số đĩa ở cọc 3 bằng N:

```
        if len(disks[2]) == N:
            text = font.render('Well done!', False, (255, 0, 0))
            screen.blit(text, (280, 200))
```

5. Trò chơi đào vàng

Trò chơi đào vàng được người chơi Việt Nam chơi khá phổ biến trên máy tính (trong những năm đầu của thế kỉ 21), và trên điện thoại (từ khi smartphone trở nên phổ biến). Trong trò chơi này, người chơi phải điều khiển một cần để lấy các cục vàng trên màn hình về.

Trong chương này, chúng ta sẽ xây dựng một trò chơi tương tự.

Chuẩn bị hình ảnh:

Thay vì chuẩn bị hình ảnh các cục vàng với kích thước khác nhau, chúng ta dùng một ảnh chung cho viên kim cương. Mặc dù tên trò chơi là đào vàng, việc dùng kim cương không làm thay đổi ý tưởng trò chơi.

Bạn hãy dùng công cụ tìm kiếm hình ảnh của google để tìm ảnh của một viên kim cương. Sau đó, hãy lưu ảnh này dưới tên **`diamond.jpg`** trong thư mục cùng với file chạy chương trình.

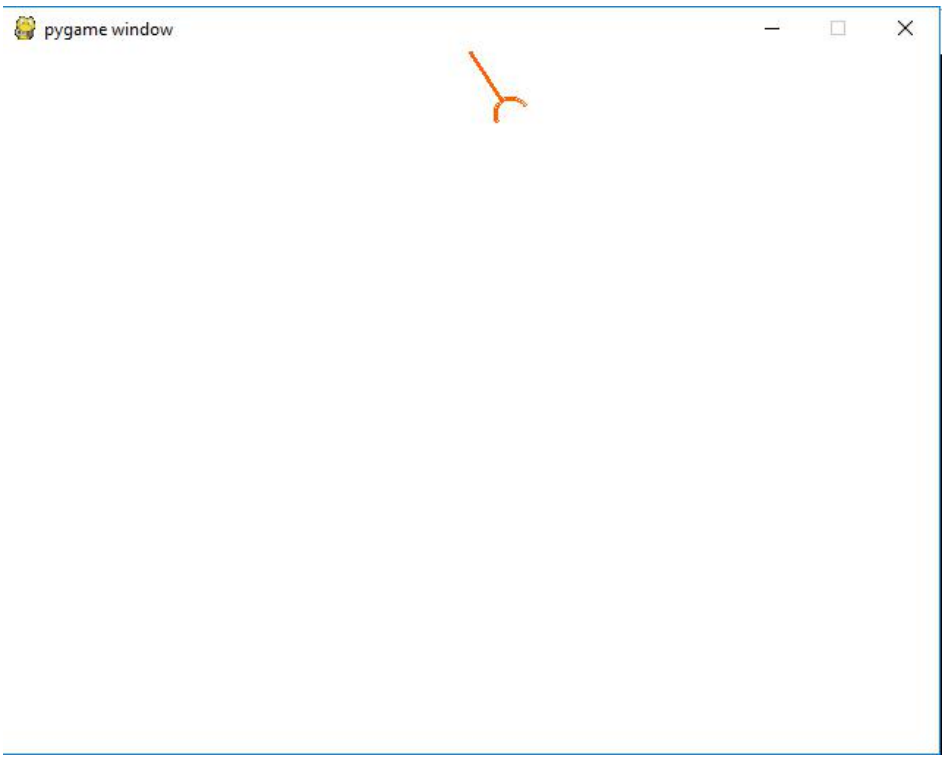
Một ảnh kim cương lấy từ công cụ tìm kiếm hình ảnh của google

Bước 1: Vẽ chiếc cần

Trong trò chơi, ở trên cùng là một chiếc cần quay liên tục qua lại màn hình để tìm kiếm vàng/kim cương. Đầu tiên, chúng ta hãy vẽ chiếc cần chuyển động này.

```
import pygame, sys, math

WHITE = 255, 255, 255
ORANGE = 255, 96, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

rect_size = 24
rod = rod_min = 40
rod_max = 500

angle = 0
clock = pygame.time.Clock()

def draw_rod(x, y, phi):
    pygame.draw.line(screen, ORANGE, (screenX/2, 0), (x, y), 3)

    cx = x + rect_size/2 * math.cos(phi)
    cy = y + rect_size/2 * math.sin(phi)

    rect = pygame.Rect(cx - rect_size/2, cy - rect_size/2, rect_size, rect_size)

    pygame.draw.arc(screen, ORANGE, rect, math.pi/2 - phi, 3*math.pi/2 - phi, 3)

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    angle += 1
    phi = abs(angle%360 - 180) * math.pi/180
    x = screenX/2 + rod * math.cos(phi)
    y = rod * math.sin(phi)
```

```
screen.fill(WHITE)
draw_rod(x, y, phi)
pygame.display.flip()
```



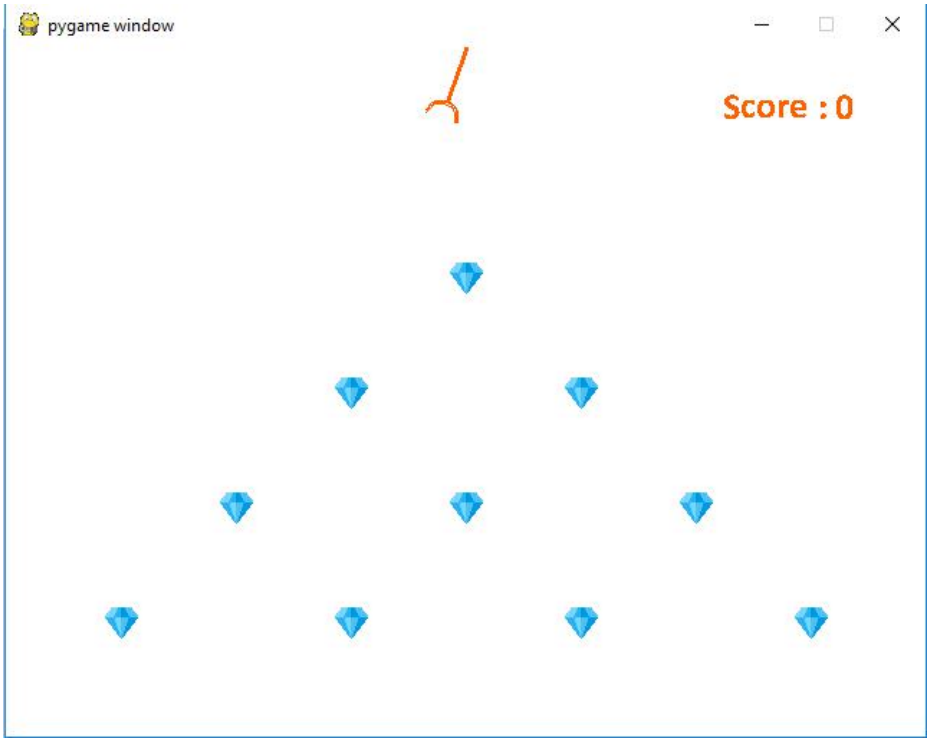
Vẽ chiếc cần chuyển động qua lại trên màn hình

Quan sát đoạn chương trình trên, chúng ta thấy chiếc cần được vẽ từ 2 phần : một đoạn thẳng nối với một nửa đường tròn.

Bước 2: Vẽ các viên kim cương

```
import pygame, sys, math

WHITE = 255, 255, 255
ORANGE = 255, 96, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

rect_size = 24
rod = rod_min = 40
rod_max = 500

pygame.font.init()
font = pygame.font.SysFont('Calibri', 26, bold=True)

image = pygame.image.load("diamond.png")
image = pygame.transform.scale(image, (rect_size, rect_size))

diamonds = [(80, 400), (240, 400), (400,400), (560, 400), (160,320),
```

```
                    (320,320), (480, 320), (240, 240), (400,240), (320, 160)]
score = 0
angle = 0
clock = pygame.time.Clock()

def draw_rod(x, y, phi):
    pygame.draw.line(screen, ORANGE, (screenX/2, 0), (x, y), 3)

    cx = x + rect_size/2 * math.cos(phi)
    cy = y + rect_size/2 * math.sin(phi)

    rect = pygame.Rect(cx - rect_size/2, cy - rect_size/2, rect_size, rect_size)

    pygame.draw.arc(screen, ORANGE, rect, math.pi/2 - phi, 3*math.pi/2 - phi, 3)

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    phi = abs(angle%360 - 180) * math.pi/180
    x = screenX/2 + rod * math.cos(phi)
    y = rod * math.sin(phi)

    angle += 1
    screen.fill(WHITE)
    draw_rod(x, y, phi)

    for (xi, yi) in diamonds:
        screen.blit(image, pygame.Rect(xi - rect_size/2, yi - rect_size/2, rect_size, rect_size))

    score_text = font.render("Score : " + str(score), False, ORANGE)
    screen.blit(score_text, (500, 30))

    pygame.display.flip()
```



Vẽ các viên kim cương

Các viên kim cương được vẽ ở các vị trí cố định trên màn hình. Ngoài ra ở góc phải phía trên, chúng ta hiển thị điểm số của người chơi, được tính bằng số viên kim cương đã lấy được.

Bước 3: Cho phép người chơi dùng chuột hoặc bàn phím để điều khiển chiếc cần.

Chúng ta thêm phần xử lý sự kiện chuột và bàn phím để cho phép người chơi điều khiển chiếc cần. Khi người chơi nhấn phím mũi tên xuống hoặc nhấn chuột, chiếc cần sẽ chuyển động xuống dưới.

```
import pygame, sys, math

WHITE = 255, 255, 255
ORANGE = 255, 96, 0

screenX, screenY = 640, 480
screen = pygame.display.set_mode((screenX, screenY))

rect_size = 24
rod = rod_min = 40
rod_max = 500

pygame.font.init()
font = pygame.font.SysFont('Calibri', 26, bold=True)

image = pygame.image.load("diamond.jpg")
image = pygame.transform.scale(image, (rect_size, rect_size))

diamonds = [(80, 400), (240, 400), (400,400), (560, 400), (160,320),
                        (320,320), (480, 320), (240, 240), (400,240), (320, 160)]
score = 0
```

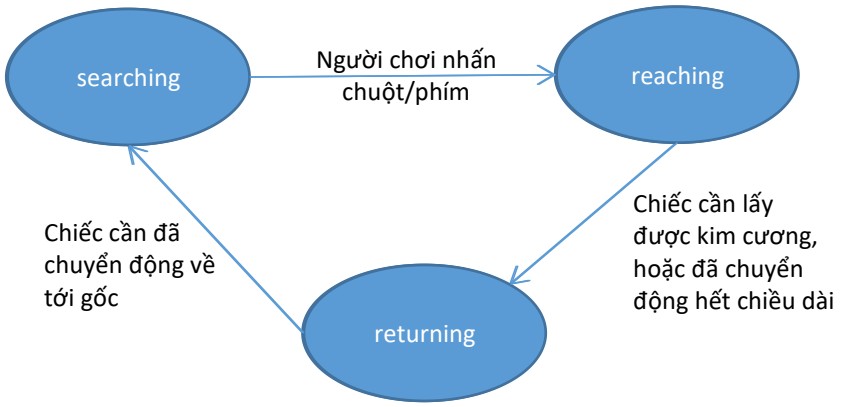
```
angle = 0
state = 'searching'
clock = pygame.time.Clock()

def draw_rod(x, y, phi, caught):
    pygame.draw.line(screen, ORANGE, (screenX/2, 0), (x, y), 3)

    cx = x + rect_size/2 * math.cos(phi)
    cy = y + rect_size/2 * math.sin(phi)

    rect = pygame.Rect(cx - rect_size/2, cy - rect_size/2, rect_size, rect_size)

    if caught:
        screen.blit(image, rect)

    pygame.draw.arc(screen, ORANGE, rect, math.pi/2 - phi, 3*math.pi/2 - phi, 3)

while True:
    clock.tick(50)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

        if (event.type == pygame.KEYDOWN and event.key == pygame.K_DOWN
            or event.type == pygame.MOUSEBUTTONDOWN):
            if state == 'searching':
                state = 'reaching'

    phi = abs(angle%360 - 180) * math.pi/180
    x = screenX/2 + rod * math.cos(phi)
    y = rod * math.sin(phi)

    if state == 'searching':
        caught = False
        angle += 1

    elif state == 'reaching':
        for point in diamonds:
            if abs(x - point[0]) + abs(y - point[1]) <= rect_size/2:
                caught = True
                diamonds.remove(point)
                break

        rod += 5
        if rod >= rod_max or caught:
            state = 'returning'

    elif state == 'returning':
        rod -= 5
        if rod <= rod_min:
            if caught:
                score += 1

            rod = rod_min
            state = 'searching'
```

```
# angle += 1
screen.fill(WHITE)

draw_rod(x, y, phi, caught)

for (xi, yi) in diamonds:
    screen.blit(image, pygame.Rect(xi - rect_size/2, yi - rect_size/2, rect_size, rect_size))

score_text = font.render("Score : " + str(score), False, ORANGE)
screen.blit(score_text, (500, 30))

pygame.display.flip()
```

Trong đoạn chương trình trên, chúng ta bổ sung biến `state` để lưu trạng thái trò chơi. Trò chơi có 3 trạng thái:

- searching : người chơi đang quan sát, chiếc cần chuyển động qua lại trên màn hình
- reaching : người chơi đã nhấn chuột/ phím mũi tên xuống, chiếc cần chuyển động xuống dưới
- returning : chiếc cần đã lấy được viên kim cương, hoặc chuyển động hết chiều dài và quay về

Quá trình chuyển giữa các trạng thái như sau:



Quá trình chuyển qua lại giữa các trạng thái của trò chơi. Điều kiện chuyển được viết trên mũi tên giữa các trạng thái

Mở rộng:

Bạn hãy giới hạn chò trơi trong thời gian một phút, sau một phút hãy đưa ra thông báo điểm số cuối cùng của người chơi.