[Bài đọc] Tuples

Tuples là gì?

Tuples được sử dụng trong Python để lưu nhiều giá trị một một biến. Các phần tử trong tuple có thứ tự nhất định và không thể thay đổi, điều này có nghĩa là sau khi tuple được tạo, nó không thể thay đổi, thêm mới hoặc xóa phần tử. Các phần tử của tuple có thể có cùng giá trị (trùng nhau) và chúng được đánh chỉ số tương ứng, phần tử đầu tiên có chỉ số là 0.

Cũng giống như list, các phần tử của tuple có thể là bất kỳ kiểu dữ liệu nào và trong 1 tuple có thể có các phần tử có kiểu dữ liệu khác nhau.

Cách tạo tuple: Tuple được tạo sử dụng dấu ngoặc tròn để khai báo các phần tử của tuple.

```
languages = ("PHP", "Python", "C#", "Java", "Android")

print("Tuple = ", languages)
```

Chú ý: Khi tạo tuple với chỉ một phần tử, cần phải sử dụng dấu phẩy sau phần tử, nếu không, biến khai báo không được coi là 1 tuple

```
# Khai báo tuple với 1 phần tử
languages = ("Python",)
print("Kiểu dữ liệu = ", type(languages))

# Trong trường hợp này languages là
# 1 string
languages = ("Python")
print("Kiểu dữ liệu = ", type(languages))
```

Ngoài cách tạo tuple như trên, chúng ta có thể tạo tuple bằng cách sử dụng hàm khởi tạo (constructor) tuple như sau:

```
languages = tuple(("PHP", "Python", "C#", "Java", "Android"))

print("Tuple = ", languages)
```

Truy xuất các phần tử của tuple: Tương tự như list, chỉ số được dùng để truy xuất các phần tử của tuple.

```
languages = tuple(("PHP", "Python", "C#", "Java", "Android"))

print("Tuple = ", languages)

# Truy xuất đến phần tử thứ 2 của tuple
print("Phần tử thứ 2 của tuple = ", languages[1])
```

```
# Truy xuất đến phần tử cuối cùng của tuple
# dùng chỉ số âm
print("Phần tử cuối cùng của tuple = ", languages[-1])

# Lấy nhiều phần tử của tuple dùng slicing
print("Phần tử thứ 2 đến phần tử thứ 4 của tuple", languages[1:4])
```

Cập nhật các phần tử của tuple: Sau khi tuple được tạo, các phần tử trong tuple là không thể thay đổi. Tuy nhiên, vẫn có cách để cập nhật (Thay đổi phần tử, thêm phần tử, xóa phần tử) trong tuple. Để làm được điều đó, đầu tiên tuple được chuyển đổi sang list, sau đó cập nhật list đó theo mong muốn. Cuối cùng lại chuyển đổi từ list đã cập nhật sang tuple. Xem ví dụ bên dưới:

```
languages = tuple(("PHP", "Python", "C#", "Java", "Android"))

print("Tuple = ", languages)

# Chuyển đổi từ tuple sang list
languages_list = list(languages)

# Thêm phần tử trên list
languages_list.append("Swift")

# Thay đổi phần tử thứ nhất
languages_list[0] = "C++"

# Xóa phần tử thứ 3
languages_list.pop(2)

# Chuyển đổi từ list sang tuple
languages = tuple(languages_list)

print("Tuple sau khi cập nhật = ", languages)
```

Mở rộng tuple bằng toán tử +: Ta có thể mở rộng tuple bằng cách thêm các phần tử của một tuple khác bằng toán tử +

```
languages = tuple(("PHP", "Python", "C#", "Java", "Android"))

print("Tuple trước khi mở rộng", languages)
languages2 = ("Swift", )

languages = languages+languages2

print("Tuple sau khi mở rộng", languages)
```

Ngoài ra, toán tử * cũng được dùng để nhân tạo số lần các phần tử của tuple

```
languages = tuple(("PHP", "Python", "C#", "Java", "Android"))

print("Tuple trước khi *", languages)
```

```
languages = languages * 2

print("Tuple sau khi *", languages)
```

Unpacking: Khi tuple được tạo, chúng ta gói các giá trị (phần tử) vào trong 1 biến kiểu tuple (quá trình này gọi là packing). Ở chiều ngược lại, khi ta lấy các giá trị của tuple lưu vào các biến được gọi là unpacking.

```
languages = ("PHP", "Python", "C#")

(php, python, c_sharp) = languages
print(php)
print(python)
print(c_sharp)
```

Lưu ý, số lượng biến dùng để unpack tuple phải bằng số lượng phần tử trong tuple. Trong trường hợp số lượng biến nhỏ hơn, ta phải thêm dấu * trước 1 biến nào đó, lúc đó biến chứa dấu * sẽ chứa tất cả những phần tử còn lại trong tuple.

```
languages = ("PHP", "Python", "C#", "Java", "Visual Basic")

(php, python, *other) = languages
print(php)
print(python)
print(other)
```

Phương thức với tuple: Hai phương thức có sẵn để làm việc với tuple: count() kiểm tra số lần xuất hiện của một giá trị nào đó trong tuple. index() là phương thức tìm kiếm một giá trị nào đó trong tuple và trả về chỉ số của phần tử đó. Ngược lại, khi tìm kiếm một phần tử không tồn tại trong tuple, chương trình sẽ phát sinh lỗi.

```
languages = ("PHP", "Python", "C#", "Java", "Visual Basic", "Python")

# Số lần phần tử Python xuất hiện trong tuple
p = languages.count("Python")
print("Số lần phần tử Python trong tuple = ", p)

# Tìm kiếm phần tử C# trong tuple
i_c_sharp = languages.index("C#")
print("Index of C# = ", i_c_sharp)

# Nếu phần tử không tồn tại trong tuple, sẽ phát sinh lỗi
print("Index of Swift = ", languages.index("Swift"))
```

[Thực hành] Rùa chạy bộ

Mục tiêu:

Luyện tập cách sử dụng và thao tác với list

Mô tả bài toán

Tạo một số con rùa (giả sử 6 con rùa với các màu khác nhau) xuất phát tại vị trí bên trái của màn hình. Trước khi các con rùa chạy từ bên trái sang bên phải màn hình, cho phép người dùng đoán trước con rùa màu nào sẽ cán đích (cách vị trí xuất phát 1 khoảng cách nào đó) đầu tiên. Khi chạy, chương trình thay đổi vận tốc các chú rùa một cách ngẫu nhiên để kết quả con rùa cán đích sẽ là ngẫu nhiên. Chương trình kết thúc khi click trên màn hình.

Hướng dẫn thực hiện

Bước 1: Import các thư viện cần thiết cho bài: turtle, random

```
import turtle as t
import random
```

Bước 2: Tạo đường đua, nhập dự đoán của người dùng, và khởi tạo các tham số của cuộc đua.

```
# Tạo và quy định dài rộng của đường đua
# và các tham số
screen = t.Screen()
screen.setup(height=500, width=600)
# Hiển thị cửa sổ cho phép người dùng đoán
# con rùa màu nào thắng cuộc
guess = screen.textinput(prompt="Dự đoán con rùa nào chiến thắng?",
title="Nhập vào màu rùa (đỏ, nâu, xanh dương, xanh lá cây, cam, hồng)")
# List lưu lại màu của các con rùa
colors = ["red", "brown", "blue", "green", "orange", "pink"]
# Vị trí ban đầu theo trục y của các con rùa
# Theo trục x = -250, cách vị trí 0,0 250 về bên trái
y_position = [0, -30, 30, -60, 60, 90]
# List lưu lại vận tốc của các con rùa
# các giá trị này sẽ được chọn một cách ngẫu nhiên
# cho các con rùa khi chạy
turtle_speed = [10, 15, 20, 25, 30, 5]

# Tạo một list để lưu các con rùa
all_turtles = []
run = True
```

Bước 3: Xây dựng hàm tạo các con rùa, thiết đặt vị trí ban đầu, màu cho rùa

```
# Xây dựng hàm để tạo và thiết đặt vị trí ban
# đầu, màu cho các con rùa và lưu vào list
for turtle in range(0, 6):
    turtles = t.Turtle(shape="turtle")
    turtles.penup()
    # Di chuyển rùa về vị trí ban đầu,
    # bên trái cùng của đường đua
    turtles.goto(x=-250, y=y_position[turtle])
    # Màu của rùa
    turtles.color(colors[turtle])
    # Lưu vào list
    all_turtles.append(turtles)
```

Bước 4: Xây dựng hàm di chuyển cho các con rùa

```
# Xây dựng hàm di chuyển về đích của
# mỗi con rùa, khoảng cách di chuyển được
# chọn ngẫu nhiên trong các giá trị
# được lưu trong list phía trên
def random_walk(turtles):
    global run
    for turtle in turtles:
        turtle.forward(random.choice(turtle_speed))
        # Kiểm tra điều kiện cán đích
        # Khi 1 con cán đích thì dừng lại
        if turtle.xcor() > 250:
            run = False
```

Bước 5: Chạy chương trình

```
while run:
    random_walk(all_turtles)

# Chương trình kết thúc khi click
# chuột lên màn hình
screen.exitonclick()
```

[Thực hành] Print the number in words

Mục tiêu:

Luyện tập với cách thức khai báo và sử dụng tuples

Mô tả bài toán

Nhập vào một dãy số. In dãy số ra dưới dạng chữ. Ví dụ: Nhập 4390 thì in ra “bốn ba chín không”

Hướng dẫn thực hiện

Bước 1: Nhập dãy chữ số từ người dùng

```
numbers = input("Nhập dãy chữ số: ")
```

Bước 2: Khai báo tuples để lưu chữ từ không đến chín

```
text = ("không", "một", "hai", "ba", "bốn", "năm", "sáu", "bảy", "tám",
"chín")
```

Để ý rằng, trong tuples trên các chữ có chỉ số tương ứng với ý nghĩa của các chữ lưu trong tuple. Do đó, các chữ số trong dãy số nhập trên thực chất là chỉ số của tuple trên.

Bước 3: In kết quả ra dùng vòng lặp

```
for num in numbers:
      print(text[int(num)], end=' ')
```

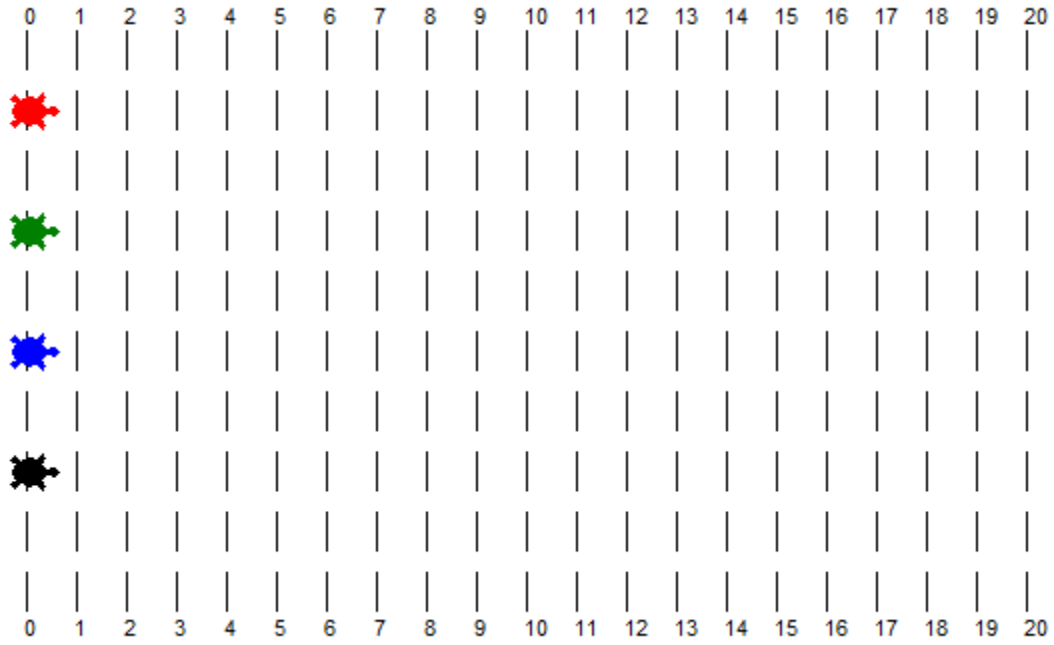
[Thực hành] Cuộc đua của những chú rùa phần 1

Mục tiêu:

Luyện tập với cách sử dụng list, các câu lệnh trong thư viên turtle

Mô tả bài toán

Bài toán rùa chạy đua như ở phần 1, chỉ khác ở chỗ ở bài toán này, có thêm các làn chạy cho rùa và các cột mốc trên đường chạy như hình dưới.



Hướng dẫn thực hiện

Bước 1: Import các thư viện cần thiết cho bài: turtle, random

```
import turtle as t
import random
```

Bước 2: Vẽ các cột mốc và làn chạy như yêu cầu bài toán

```
# Tạo khung cảnh và vẽ các cột mốc bên trái đường đua
screen = t.Screen()
screen.setup(height=500, width=600)
pen = t.Turtle(visible=False)
pen.penup()
pen.speed(0)
pen.goto(-250, 200)
for i in range(21):
    pen.write(i)
    pen.forward(25)

# Vẽ các đường đứt đoạn trên đường đua và
# đánh dấu các cột mốc bên phải đường đua
x = -250
pen.goto(-250, 200)
pen.right(90)
for i in range(21):
    for j in range(10):
        pen.pendown()
        pen.forward(20)
```

```
        pen.penup()
        pen.forward(10)
    pen.penup()
    pen.forward(5)
    pen.write(i)
    pen.goto(x + (i + 1) * 25, 200)
```

Bước 3: Tạo các con rùa và cho rùa về vị trí xuất phát của làn chạy bên trái cùng

```
all_turtles = []
y_position = [160, 100, 40, -20]
colors = ['red', 'green', 'blue', 'black']
for turtle in range(0, 4):
    turtles = t.Turtle(shape="turtle")
    turtles.penup()
    # Di chuyển rùa về vị trí ban đầu,
    # bên trái cùng của đường đua
    turtles.goto(x=-250, y=y_position[turtle])
    # Màu của rùa
    turtles.color(colors[turtle])
    for i in range(5):
        turtles.left(72)
    # Lưu vào list
    all_turtles.append(turtles)
```

Bước 4: Xây dựng hàm di chuyển về phía trước của các con rùa, các con rùa sẽ dừng chạy khi có 1 con cán đích đầu tiên.

```
def random_walk(turtles):
    global run
    for turtle in turtles:
        turtle.forward(random.randint(1, 10))
        # Kiểm tra điều kiện cán đích
        # Khi 1 con cán đích thì dừng lại
        if turtle.xcor() > 250:
            run = False
```

Bước 5: Gọi hàm ở bước 4

```
run = True
while run:
    random_walk(all_turtles)
screen.exitonclick()
```

Bài 2. Đảo tuple

Cho 1 tuple = (10, 20, 30, 50). Đảo ngược tuple đó thành reverse_tuple = (50, 30, 20, 10)

Bài 3. Modify tuple

Cho 1 tuple1 = (11, [22, 33], 44, 55) . Viết 1 chương trình sửa phần tử 22 của list của tuple thành phần tử 222

Output : tuple1: (11, [222, 33], 44, 55)

Bài 4. Sắp xếp tuple

Cho 1 tuple như sau : tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

Sắp xếp tuple để được kết quả ouput như sau : (('c', 11), ('a', 23), ('d', 29), ('b', 37))