Tài liệu hướng dẫn thực hành Buổi 2
Môn : Lập trình Python

Nội dung chính : Kiểu dữ liệu trong python

1. Các kiểu dữ liệu

Python gồm các kiểu dữ liệu cơ bản như sau :

Kiểu số – Number : Kiểu dữ liệu numbers chứa giá trị là một con số, nó được tạo ra khi ta gán các giá trị là một con số cho biến đó.

Python hỗ trợ 4 kiểu dữ liệu numbers khác nhau:

- int
- long
- float
- complex

Ví dụ :

```
a = 5
b = -7
c = 1.234
```

Kiểu chuỗi – String : Kiểu dữ liệu strings được xác định khi giá trị được gán cho nó nằm giữa cặp dấu ' ' hoặc " "

Ví dụ :

```
str1 = "Hello"
str2 = "welcome"
str3 = "abcdef12345"
```

Kiểu danh sách – List : Một list trong Python là một nhóm có thứ tự các đối tượng. Điều quan trọng cần phải nhấn mạnh là những đối tượng đó không cần phải là cùng loại. Nó có thể là một hỗn hợp tùy ý của các đối tượng như số, chuỗi hoặc có thể là một danh sách khác.

Dưới đây là một số thuộc tính của list:

- List là một tập hợp có thứ tự.
- Dữ liệu trong một list có thể thuộc nhiều kiểu khác nhau.
- Các phần tử trong list có thể được truy xuất thông qua chỉ số của chúng.
- Kích thước của một list có thể thay đổi.
- Chúng ta có thể thay đổi một list, ví dụ như việc thêm mới, xóa hoặc cập nhật phần tử trong list.

Ví dụ : Khai báo một danh sách gồm 5 phần tử kiểu chuỗi :

```
cats = ['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']
```

Chúng ta có thể truy xuất các phần tử của list thông qua chỉ số của chúng như sau:

```
print cats[2] # in ra phần tử có chỉ số là 2
// Kitty
```

```
print cats[-1] # in ra phần tử cuối cùng của list
// Chester

print cats[1:3] # in ra các phần tử có chỉ số từ 1 đến 3
//['Snappy', 'Kitty']
```

Vì list trong Python có thể thay đổi nên chúng ta có thể thêm mới, sửa hoặc xóa bỏ phần tử trong list:

```
print cats
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']

cats.append('Jerry') # thêm mới phần tử vào list
print cats
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester', 'Jerry']

cats[-1] = 'Jerrt Cat' # gán lại giá trị cho phần tử cuối cùng của list
print cats
['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester', 'Jerrt Cat']

del cats[1] # xóa bỏ phần tử có số thứ tự 1 khỏi list
print cats
['Tom', 'Kitty', 'Jessie', 'Chester', 'Jerrt Cat']
```

Bảng dưới đây liệt kê các hàm dùng để làm việc với list:

- `cmp(list1, list2)`: So sánh phần tử của 2 list
- `len(list)`: Lấy số lượng phần tử trong list.
- `max(list)`: Trả về phần tử có giá trị lớn nhất trong list.
- `min(list)`: Trả về phần tử có giá trị nhỏ nhất trong list.
- `list.append(obj)`: Thêm một phần tử vào list.
- `list.count(obj)`: Đếm số lần xuất hiện của phần tử obj trong list
- `list1.extend(list2)`: Thêm tất cả các phần tử của list2 vào list1
- `list.index(obj)`: Trả về chỉ số bé nhất mà obj xuất hiện trong list
- `list.insert(index, obj)`: Thêm phần tử obj vào vị trí index trong list.
- `list.pop(index)`: Xoá và trả về phần tử có chỉ số index trong list.
- `list.remove(obj)`: Xóa phần tử trong list.
- `list.reverse()`: Đảo ngược các phần tử trong list.
- `list.sort()`: Sắp xếp các phần tử trong list.

Kiểu bộ – Tube : Một Tuple có thể được hiểu là một danh sách không thay đổi, điều này có nghĩa là bạn không thể thay đổi một tuple một khi nó đã được tạo ra. Một tuple được định nghĩa tương tự như list ngoại trừ việc tập hợp các phần tử được đặt trong dấu ngoặc đơn thay vì dấu ngoặc vuông như list, ngoài ra quy tắc về chỉ số phần tử của tuple cũng tương tự như list.

Vậy tại sao Python lại định nghĩa một kiểu dữ liệu tương tự như list? Dưới đây là các ưu điểm của tuple:

- Tốc độ truy xuất, xử lí dữ liệu của tuple nhanh hơn so với list.
- Trong lập trình, có những dữ liệu không được thay đổi, trong trường hợp đó bạn nên dùng tuple thay vì list, điều này sẽ đảm bảo được dữ liệu của bạn, chống lại những sự thay đổi ngẫu nhiên đối với dữ liệu đó.

Bảng dưới đây liệt kệ các hàm dùng để làm việc với list:

- `cmp(tuple1, tuple2)`: So sánh phần tử của 2 tuple
- `len(tuple)`: Lấy số lượng phần tử trong tuple.
- `max(tuple)`: Trả về phần tử có giá trị lớn nhất trong tuple.
- `min(tuple)`: Trả về phần tử có giá trị nhỏ nhất trong tuple.
- `tuple(list)`: Chuyển một list sang tuple.

Kiểu Tập hợp – Set : Đây là kiểu tập hợp các phần tử duy nhất và không có thứ tự. Các phần tử trong một Set nằm giữa cặp dấu ngoặc {} và phân cách với nhau với dấu phẩy. Ta có thể thêm sửa xóa các dữ liệu bên trong 1 set.

Một số hàm dùng cho Set trong Python:

- `add()` : Thêm một phần tử vào set.
- `clear()` : Xóa tất cả phần tử của set.
- `copy()` : Trả về bản sao chép của set.
- `difference()` :Trả về set mới chứa những phần tử khác nhau của 2 hay nhiều set.
- `difference_update()` : Xóa tất cả các phần tử của set khác từ set này.
- `discard()` : Xóa phần tử nếu nó có mặt trong set.
- `intersection()` : Trả về set mới chứa phần tử chung của 2 set.
- `intersection_update()` : Cập nhật set với phần tử chung của chính nó và set khác.
- `isdisjoint()` : Trả về True nếu 2 set không có phần tử chung.
- `issubset()` : Trả về True nếu set khác chứa set này.
- `issuperset()` : Trả về True nếu set này chứa set khác.
- `pop()` : Xóa và trả về phần tử ngẫu nhiên, báo lỗi KeyError nếu set rỗng.
- `remove()` : Xóa phần tử từ set. Nếu phần tử đó không có trong set sẽ báo lỗi KeyError.
- `symmetric_difference()` : Trả về set mới chứa những phần tử không phải là phần tử chung của 2 set.
- `symmetric_difference_update()` :Cập nhật set với những phần tử khác nhau của chính nó và set khác.
- `union()` : Trả về set mới là hợp của 2 set.
- `update()` : Cập nhật set với hợp của chính nó và set khác.

```python
A = {1, 2, 3, 4, 5, 6}
B = {4, 5, 6, 7, 8, 9}
print(A | B) # output : set([1, 2, 3, 4, 5, 6, 7, 8, 9])
# using union()
print(A.union(B)) # output : set([1, 2, 3, 4, 5, 6, 7, 8, 9])

print(A & B) # output : set([4, 5, 6])
# using intersection()
print(A.intersection(B)) # output : set([4, 5, 6])

print(A - B) # output :set([1, 2, 3])
# using different()
print(A.difference(B)) # output : set([1, 2, 3])
print(B.difference(A)) # output : set([8, 9, 7])

print(A ^ B) # output : set([1, 2, 3, 7, 8, 9])
# using symmetric_difference()
print(A.symmetric_difference(B)) # output : set([1, 2, 3, 7, 8, 9])
```

Kiểu từ điển – Dictionary : Một Dictionary trong Python có thể được hiểu là một tập các cặp key-value. Mỗi key được phân cách với giá trị của nó bằng dấu hai chấm (:), các phần tử trong ditionary được phân cách bằng dấu phẩy. Một dictionary rỗng, không chứa bất kì phần tử nào được định nghĩa bằng hai dấu ngoặc nhọn như sau: {}.

Key trong dictionary phải là duy nhất nhưng điều này không cần thiết với các value, nghĩa là trong một dictionary có thể có nhiều key có cùng một value. Các value trong dictionary có thể thuộc bất kì một kiểu nào còn các key thì chỉ được cố định trong một số kiểu như: string, number hoặc tuple.

Dictionary không dùng chỉ số của các phần tử để truy xuất dữ liệu, thay vào đó, chúng ta có thể truy xuất dữ liệu thông qua key như ví dụ sau:

```
dict1 = {'Name' : 'Zyra', 'Age' : 7, 'Class' : 'A5'} # định nghĩa 1 dictionary
print dict['Name'] # lấy giá trị có key = 'Name'
'Zyra'
print dict['Age'] # lấy giá trị có key = 'Age'
7
```

Chúng ta có thể chỉnh sửa một Dictionary như việc thêm mới, xóa, hoặc chỉnh sửa phần tử như ví dụ dưới đây:

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8 # cập nhật một phần tử trong dict
dict['School'] = "DPS School"  # thêm mới một phần tử vào dict
print dict['Age']
// 8
print  dict['School']
// DPS School
```

Dưới đây là danh sách các hàm để làm việc với Dictionary trong Python:

- `cmp(dict1, dict2)`: So sánh phần tử của 2 dict
- `len(dict)`: Lấy số lượng phần tử trong dict.
- `str(dict)`: Tạo ra một chuỗi có thể in được của dict

- `type(variable)`: Trả về kiểu của biến được truyền vào, nếu biến được truyền vào là kiểu dictionary hàm sẽ trả về kiểu dictionary.
- `dict.clear()`: Xóa tất cả các phần tử của dict
- `dict.copy()`: Trả về một bản sao của dict
- `dict.get(key, default=None)`: Trả về giá trị tương ứng với key hoặc trả về giá trị default nếu key không tồn tại trong dict
- `dict.has_key(key)`: Trả về true nếu key tồn tại trong dict, ngược lại trả về false.
- `dict.items()`: Trả về một list các cặp key/value của dict.
- `dict.keys()`: Trả về một list các key của dict
- dict.setdefault(key, default=None): Hàm này gần giống với hàm get() nhưng sẽ gán dict[key]=default nếu key chưa tồn tại trong dict.
- `dict1.update(dict2)`: Thêm các phần tử của dict2 và dict1.

2. Bài tập

Sinh viên tạo file <<Hoten>>_THBuoi2.py (vd : tranvanan_THBuoi2.py), Nhớ chú thích họ tên và MSSV lên đầu file.

Câu 1 : Viết một chương trình tìm tất cả các số trong đoạn 100 và 300 (tính cả 2 số này) sao cho tất cả các chữ số trong số đó là số chẵn. In các số tìm được thành chuỗi cách nhau bởi dấu phẩy, trên một dòng.

Câu 2 : Viết chương trình cho người dùng nhập vào một dãy số, cách nhau bởi dấu phẩy, lọc ra các số lẻ từ dãy số đó và in nó ra màn hình. Ví dụ: người dùng nhập : 1, 5, 4, 10, 6, 3, 7 thì kết quả sẽ là 1, 5, 3, 7

Câu 3 : Khai báo một từ điển mà key là các số từ 1 – 20, và value là các giá trị bình phương của key tương ứng. In từ điển trên ra màn hình theo mẫu : Key-value

Câu 4 : Viết chương trình in list sau khi đã xóa số ở vị trí thứ 0, thứ 4, thứ 5 trong [12,24,35,70,88,120,155]

Câu 5 : Với 2 list cho trước: [12,3,6,78,35,55,120] và [12,24,35,78,88,120,155], viết chương trình để tạo list có phần tử là giao của 2 list đã cho

Câu 6 : Viết chương trình tạo một list chứa giá trị bình phương của các số từ 1 – 20. In list đó ra màn hình trừ 5 giá trị đầu tiên.

Câu 7 : Viết chương trình cho phép người dùng nhập vào một tuple gồm 7 giá trị số. Nếu giá trị người dùng nhập vào không phải số thì yêu cầu họ nhập lại. Sau đó tạo một tuple khác chứa các giá trị là số chẵn trong tuple của người dùng nhập vào.

Câu 8 : Viết chương trình cho phép người dùng nhập vào một câu, sau đó đếm và in số lượng của từng ký tự trong câu đó. Ví dụ : Người dùng nhập “Hello” thì sẽ in ra màn hình :

H : 1
e : 1
l : 2
o : 1

Gợi ý : dùng kiểu dữ liệu từ điển

Câu 9 : Viết chương trình cho phép người dùng nhập vào một câu, và in nó ra màn hình theo thứ tự ngược lại. Ví dụ : Người dùng nhập “Hello” thì in ra sẽ là : “olleH”

Câu 10 : Viết chương trình cho phép người dùng nhập vào một chuỗi, và in ra màn hình một chuỗi mới được tạo thành bởi 2 ký tự đầu và 2 ký tự cuối của chuỗi vừa nhập vào. Nếu chiều dài chuỗi nhập vào nhỏ hơn 2 ký tự thì yêu cầu người dùng nhập lại. Ví dụ: Người dùng nhập “abcd1234” thì in ra sẽ là : “ab34”, Người dùng nhập “ab” thì in ra sẽ là : “abab”.

Câu 11 : Viết chương trình cho phép người dùng nhập vào 2 chuỗi. Tạo một chuỗi mới là ghép của 2 chuỗi củ, cách nhau bởi dấu phẩy, và hoán đổi 2 ký tự đầu của 2 chuỗi đó.

Ví dụ :

Chuỗi 1 : abc
Chuỗi 2 : xyz
Chuỗi kết quả : xyc abz

Câu 13 :Viết chương trình cho người dùng nhập vào một chuỗi.Thay thế ký tự trùng lặp của ký tự đầu tiên trong chuỗi đó bằng ký tự @. In chuỗi mới ra màn hình. Ví dụ : Người dùng nhập "abcabc" thì in ra sẽ là : "abc@bc"

Câu 14 : Viết hàm getsuffix(a,b) với a và b là 2 list chứa các giá trị chuỗi ký tự. Kiểm tra xem có ký tự nào là tiền tố của ký tự còn lại giữa 2 list a và b hay không. Nếu có thì tách phần hậu tố và thêm vào một biến có kiểu Set. Hàm trả về biến có kiểu set chứa các phần tử là hậu tố đã tìm được. (Gợi ý : Dùng hàm startswith() để kiểm tra tiền tố và hàm rsplit() để tách phần hậu tố)

Câu 15 : Dựa vào hàm getsuffix() ở câu 14, hãy viết một hàm Kiemtrabangma(a) nhận vào một list a là một bảng mã gồm nhiều từ mã được lưu trữ dưới dạng chuỗi. Sau đó tiến hành kiểm tra tính tách được của bảng mã a bằng thuật toán đã học.

Gợi ý : Dựa vào đoạn mã giả bên dưới, sinh viên hãy lập trình giải thuật kiểm tra tính tách được của bảng mã :

```
def kiemtrabangma( a ) :
        Khởi tạo 2 list S0 và S1
        S1 = getsuffix (S0,S0)

        While(S1 không rỗng ):
                Tạo biến S2 kiểu Set
                S2 = getsuffix (S0,S1)
                Stemp = S2.intersection(S0)

                if S2 = rỗng :
                        print ("Đây là bảng mã tách được")
                elif Stemp != rỗng :
                        print ("Đây là bảng mã không tách được")
                elif S2 = Sn trước đó
                        print ("Đây là bảng mã không tách được")
```

Chú ý : Sinh viên tự điều chỉnh thêm vào các biến số cần thiết.

Câu 16 : Sử dụng hàm kiemtrabangma() ở câu 15 để kiểm tra tính tách được của các bảng mã sau :

```
a = ["abc","a","ab","b","bcad"]
b = ["a", "ad","bbcde", "c", "deb","ebd"]
c = ["a", "ad","bbcde", "c", "deb","ebad"]
```