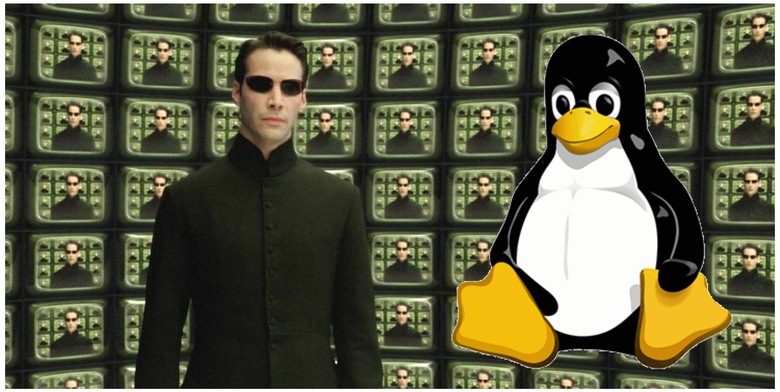
15.10.21 20:51

CSCI 120
Introduction to Computer Science and Programming I
Lecture 5: Dictionaries



Jetic Gū

`dict` type data

A very powerful tool

P1
dict

Data Structures

- Data structures are particular ways of storing data to make some operation easier or more efficient. That is, they are tuned for certain tasks

- Data structures are suited to solving certain problems, and they are often associated with algorithms.

Concept

P1
dict

Kinds of data structures

Roughly two kinds of data structures:

- built-in data structures, data structures that are so common as to be provided by default

- user-defined data structures (classes in object oriented programming) that are designed for a particular task

Concept

P1
dict

Python built in data structures

- Python comes with a general set of built in data structures:
  - lists √
  - tuples √
  - string √
  - dictionaries (This lecture)
  - others...

Concept

P1
dict

Python `dict`

- Indexed Access
  - In `list`/array, indexed access can be very convenient, in Array practice 1, you have seen how easy and fast it is to do so
  - But what if you want the "index" to be a string?

Concept

P1
dict

Company Telephone Directory

Name	Phone Number
Graham Chapman	555-606-1234
John Cleese	555-606-5464
Terry Gilliam	555-445-4654
Eric Idle	555-687-4534
Terry Jones	555-756-1395
Michael Palin	555-987-1213

- You want to provide fast retrieval for people's telephone number
- Queries are now done using names, instead of ID numbers
- Using a `list` is OK, but `dict` can do it better!

P1
dict

Company Telephone Directory

```
phone = {}
phone["Graham Chapman"] = "555-606-1234"
phone["John Cleese"]    = "555-606-5464"
phone["Terry Gilliam"]  = "555-445-4654"
phone["Eric Idle"]      = "555-687-4534"
phone["Terry Jones"]    = "555-756-1395"
phone["Michael Palin"]  = "555-987-1213"

# Query
print(phone["Terry Jones"])
```

Name	Phone Number
Graham Chapman	555-606-1234
John Cleese	555-606-5464
Terry Gilliam	555-445-4654
Eric Idle	555-687-4534
Terry Jones	555-756-1395
Michael Palin	555-987-1213

- To create a `dict`, use curly brackets
  `a_dict = {}`

- To create a new entry in the `dict`, use `(key, value)` pairs
  `a_dict[key] = value`

P1
dict

Company Telephone Directory

```
phone = {
    "Graham Chapman": "555-606-1234",
    "John Cleese":    "555-606-5464",
    "Terry Gilliam":  "555-445-4654",
    "Eric Idle":      "555-687-4534",
    "Terry Jones":    "555-756-1395",
    "Michael Palin":  "555-987-1213",
}

# Query
print(phone["Terry Jones"])
```

Name	Phone Number
Graham Chapman	555-606-1234
John Cleese	555-606-5464
Terry Gilliam	555-445-4654
Eric Idle	555-687-4534
Terry Jones	555-756-1395
Michael Palin	555-987-1213

- Alternative declaration, using `key:value` pairs separated by comma
- Inside the curly bracket, you may have multiple lines ended by comma, this is the same effect as a single line

P1
dict

Python `dict`

```
a = {}
a[1] ="red"
a[1.0] ="Honey"
a["funny stuff"] = list(range(10))
```

- The `key` of a python `dict`, must be an immutable value
  - `int`, `float`, `str`
  - `tuple`
- The `value` of a python `dict`, can be anything

Technical

Python `dict`

```
a_dict = ["a":1, "b":1.0, "c":"funny stuff"]
print("There are", len(a_dict), "entries in a_dict")
print("There are", len(a_dict.keys()), "keys in a_dict")
```

- `len()` function can be used to check the number of entries in a `dict`
- `dict` method `keys()` will return a list of all keys in the `dict` instance
- `len(a_dict)` and `len(a_dict.keys())` are equal

Python `dict`

```
# Continue the phone book example
for key in phone:
    print(key, ":", phone[key])

# The above is equivalent to
for key in phone.keys():
    print(key, ":", phone[key])
```

- If you use a `dict` instance as an iterator, it will iterate through all of the keys in that dict

Methods of `dict`

Find out if a key entry exists

```
if key in a_dict:
    print("Key", key, "exists in a_dict!")
```

- You can use the `in` operator to determine if a key entry exists or not
  - This is important, as if you try to access a key that doesn't exist, your programme will crash

`keys` and `values` method

```
# Following the phone book example
print(phone.values())
```

- `keys()` method
  Returns a `list` of all of the **keys** in the `dict` instance

- `values()` method
  Returns a `list` of all of the **values** in the `dict` instance

Remove entry from `dict`

```
# Create new entry in dict instance phone
phone["Jetic"] = "555-666-6666"

# Remove that entry from dict instance phone
phone.pop("Jetic")
# Alternatively, del also works
del phone("Jetic")
```

- `pop()` and `del`
  - `pop()` is a class method, `del` is more universal

Default values for entries

```
count = {}
for word in words:
    if word in count:
        count[word] += 1
    else:
        count[word] = 1
```

- Must always see if a key exists before increasing count
  - Otherwise upon seeing a new `word`, your programme will crash

clear()

```
phone.clear()
print(len(phone))
```

- `clear()` method removes all existing entries from a dict
  - Literally clears everything