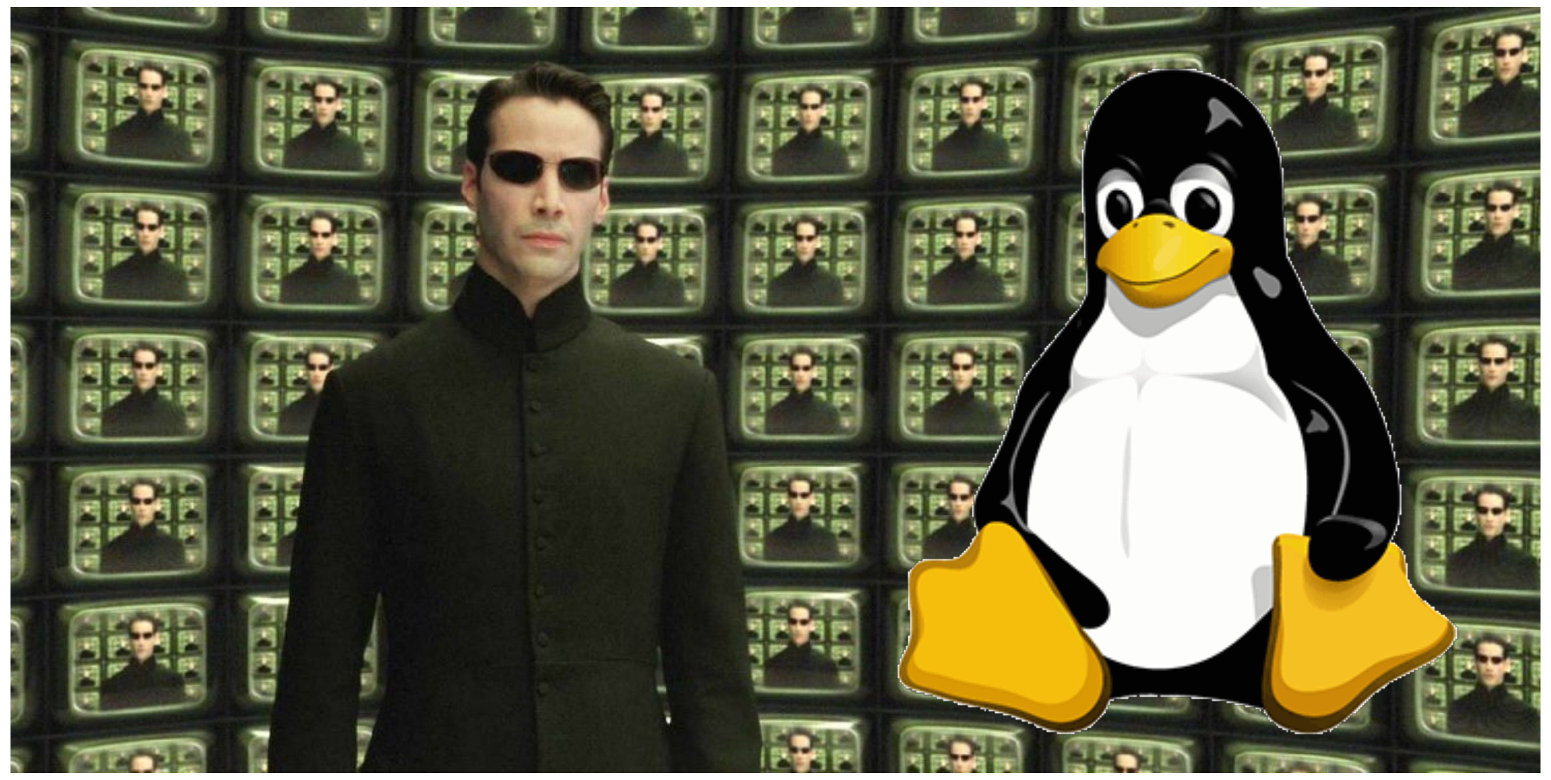
08.10.21 16:10

CSCI 120
Introduction to Computer Science and Programming I
Lecture 4: Lists



Jetic Gū

Overview

- Focus: Basic Python Syntax
- Core Ideas:
  1. `list`s revisited
  2. `list` as a class: methods of `list`

`list` **type data**

Literally a list of variables...

Python `list`

- An **ordered** sequence of variables
  - `type([1, 2, 3, 4, 5])   # This would return list`
    list instances are declared using square brackets, with values inside separated by comma
  - Indexed access: just like str, values inside the `list` can be accessed through indexing

P1
List

Python `list`

```
a = [1, 24, 76]
b = ["red", "green", "blue"]
c = [1, 1.0, "funny stuff"]
```

- A `list` instance can have different types of values
  - In the last example above, `list` instance c contains an `int`, a `float`, and a `str`

Technical

P1
List

Python `list`

```
c = [1, 1.0, "funny stuff"]
print("There are", len(c), "objects in list c")
d = [1, 2, c]
print("There are", len(d), "objects in list d")
```

- `len()` function can be used to check the number of characters in a string
- `len()` function can also be used to check the number of objects in a list
- A `list` instance itself is also an object

Technical

P1
List

Python `list`

```
a = [1, 2, 3, 4, 5]
for item in a:
    item = item + 1  # This will not work
for i in range(len(a)):
    a[i] = a[i] + 1  # This will work
```

- If you want to change the value of an object in a list, you must use indexing

Technical

Python `list`

```
a = [1, 2, 3, 4, 5]
b = [5, 6, 7, 8, 9]
c = a + b  # c = [1, 2, 3, 4, 5, 5, 6, 7, 8, 9]
d = b + a  # d = [5, 6, 7, 8, 9, 1, 2, 3, 4, 5]
```

- Two lists can be combined as one using the plus sign
  - This is called concatenation, similar to what you have in linear algebra

Technical

P1
List

Python `list`

```
a = [1, 2, 3, 4, 5]
b = [5, 6, 7, 8, 9]
c = a[1:3]
d = (b + a)[4:8]
```

- Slicing
  - Using indexing, you can take a slice of the list as a "sub-list"
  - Similar to substrings, but careful: indexing in `str` always returns a `str`, for list it could be an element (e.g. `d[3]`) or a sub-list (e.g. `d[3:4]`)

Technical

Methods of `list`

What is a method?

- In python, you have a few things that are **callable**
  - A callable thing can be called using parenthesis (e.g. `print(...)`)
  - Callable names: **functions**, and **methods**
  - Functions can be called on its own, methods must be called by instances
    - e.g. you have a `str` variable, `split` is a method for `str` type data
      `print("I like cheese".split())`
  - Methods can only be called on values/variables with that type

Concept

append method

```
a = []
a.append(1)
a.append("second element")
print(a)
```

- You can add values/variables/objects to the end of a list using append
- `append` is the name of a method for `list` objects

Concept

insert method

```
a = []
a.insert(0, 1)
a.insert(0, "new first element")
a.insert(1, "new second element")
print(a)
```

- `insert` takes 2 arguments: `position`, and `value`
  - `position` specifies which index the new `value` is inserted in
  - after insertion, every element originally from `[position:]` is pushed back by to `[position+1:]`

Concept

Check of a value is in a list

```
a = ["Jetic", "Jeremy", "Jenny", "Johanna"]

if "Jetic" in a:
    print("Jetic is in the list!")
else:
    print("Jetic is not in the list!")
```

- `in` operator
  - Similar to strings, the `in` operator lets you check if a particular value is inside the list

Concept

Some extras

```
a = ["Jetic", "Jeremy", "Jenny", "Johanna"]
a.sort()
```

- Sort alphabetically (works with `list` of `str` only)
  - The `sort` method can help you sort a `list` with all `str` values alphabetically

Technical

Some extras

```
a = [5, 7, 1, 9]
a.sort()
```

- Sort in increase order (works with `list` of `int` or `float` only)
  - The `sort` method can help you sort a `list` with all numerical values in increasing order

Some extras

```
a = [5, 7, 1, 9]
print("The greatest value in a is:", max(a))
print("The smallest value in a is:", min(a))
print("The sum of all values in a is:", sum(a))
print("The average of all values in a is:", sum(a)/len(a))
```

- min, max, sum
  - works with `list` of `int` or `float` only

Technical