LECTURE 30

REGULAR EXPRESSIONS

MCS 260 Fall 2021
Emily Dumas

REMINDERS

- Homework 10 due tomorrow at 10am
- Worksheet 11 coming this afternoon
- Project 3 autograder open
- Project 3 due Fri at 6pm

LOOSE END: RECURSION PROS AND CONS

Often can solve a problem with recursion or with loops (an **iterative** solution). Why use recursion?

Pros:

- Short code
- Clear code

Unclear:

- Speed

Cons:

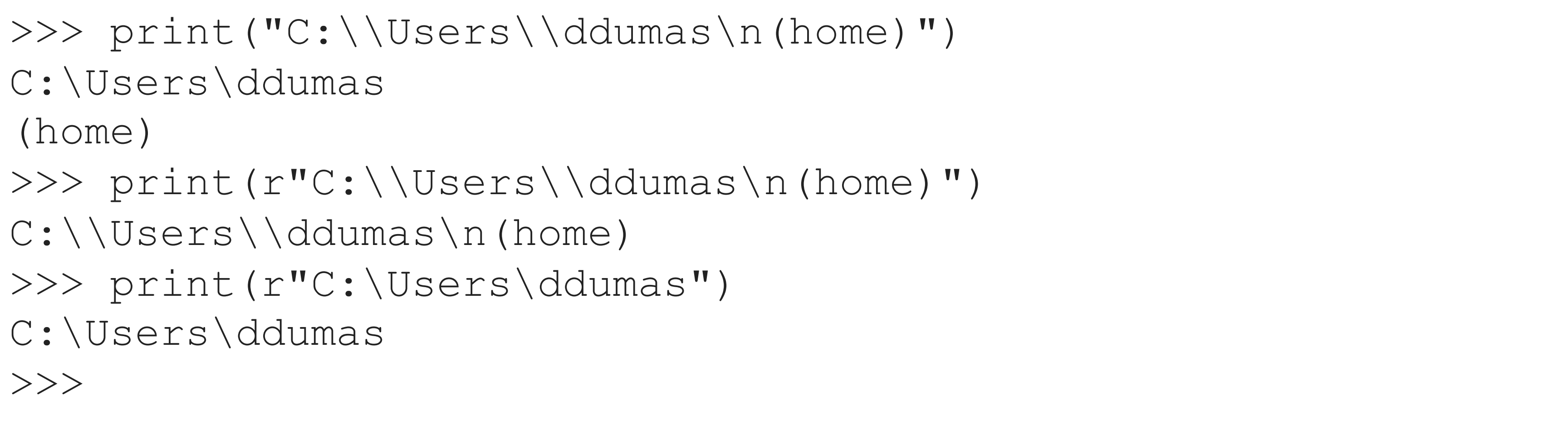
- Uses more memory

RAW STRINGS

Recall that backslash `\` in a string starts an escape sequence in Python.

You can disable escape sequences by putting the letter `r` immediately before the quotation mark(s). This is known as a **raw string**. In a raw string, a single \ represents the \ character.

However, raw strings cannot end with a single \



```
>>> print("C:\\Users\\ddumas\n(home)")
C:\Users\ddumas
(home)
>>> print(r"C:\\Users\\ddumas\n(home)")
C:\\Users\\ddumas\n(home)
>>> print(r"C:\Users\ddumas")
C:\Users\ddumas
>>>
```

REGULAR EXPRESSIONS

Today we'll learn about the module `re` in Python, which supports a text searching language known as **regular expressions** or **regexes**.

Some of its key functions include:

- Searching for text matching a pattern
- Replacing text matching a pattern

LANGUAGE SUPPORT

Regexes are a mini programming language for specifying patterns of text.

Dialects of regex are supported in many programming languages. We'll cover the Python dialect.

MINIMAL EXAMPLE

Simplest usage: Find and replace a substring.

```
import re
s = "Avocado is usually considered a vegetable."
print(re.sub("vegetable","fruit",s))
```

`re.sub(pattern, replacement, string)`

The first argument of `re.sub` is a **pattern**.

Unless it contains characters with special meaning in a regex pattern, the pattern just matches substrings equal to the pattern.

- `"vegetable"` matches the string `"vegetable"`
- `"foo"` matches the string `"foo"`

SPECIAL CHARACTERS IN PATTERNS

- `.` — matches any character except newline
- `\s` — matches any whitespace character
- `\d` — matches a decimal digit
- `\w` — matches a "word character" (a-z, A-Z, 0-9, _)

SPECIAL CHARACTERS FOR REPETITION

- `+` — previous item must repeat 1 or more times
- `*` — previous item must repeat 0 or more times
- `?` — previous item must repeat 0 or 1 times
- `{n}` — previous item must appear n times

EXAMPLE PROBLEM

Replace any price in whole dollars (written like `$2` or `$1999`) with the string `-PRICE-`.

Note: `$` is a special character. To match a dollar sign, put `\$` in the pattern.

SEARCHING WITHOUT REPLACING

MATCH OBJECTS

Most regex functions return *match objects* that contain info about a part of the string matching the expression.

A match object has a method `.group()` that returns the full text of the match.

`.start()` and `.end()` return the indices where the match begins and ends in the string.

PARENTHESES

A part of a pattern in parentheses is a **group**. A group is treated as a unit for operators like `+`, `*`, `?`.

e.g. pattern `(ha)+` means one or more repetitions of `ha`.

It matches `ha` or `haha` or `hahaha` but does not match `Haha` or `h` or `hah`.

In contrast, `ha+` means the letter `h` followed by one or more repetitions of `a`, e.g. `haaaaaaa`

RETRIEVING GROUPS

Matched groups are available as `.group(1)`, `.group(2)`, etc., with the 1-based number referring to the order of left parentheses in the pattern.

Group 0 always refers to the entire pattern.

e.g. pattern `My name is (\w+).` will capture the name (not containing spaces!) in group 1.

EXAMPLE PROBLEM

Find all of the phone numbers in a string that are written in the format `319-555-1012`, and split each one into area code (e.g. `319`), exchange (e.g. `555`), and line number (e.g. `1012`).

REFERENCES

- pythex.org is a nice web tool to check regex matches (and debug problems)
- In *Downey*:
  - Regular expressions are not discussed.
- The documentation of the `re` module is good as a reference.
- Google's free online Python course has a unit on regular expressions.
  - This course was developed for Python 2, so calls to `print` are lacking parentheses. Otherwise, the code should work.

REVISION HISTORY

- 2021-11-01 Initial publication