# ECS 10

10/27

## Assignment

- Due Wds. Night
- Understand before you start:
  - Lists
  - **for** loops
  - Indexing lists and strings
  - The split method
  - The in operator
  - The range function

## All a computer does…

- The things we learned in the first month:
  - Floating point, integer, string and Boolean (**and, or, not**) expressions
  - Variables
  - **if** statements
  - **while** loops
- …are really all that a computer does
- The rest is `syntactic sugar'
- Big exception: internet.

## The find method

```
string = '$23.95'
decimal = string.find('.')
```

- decimal gets the value 3.
- **find** returns the first index of the string argument in the parenthesis.

```
string = 'romania'
string.find('oman')
```

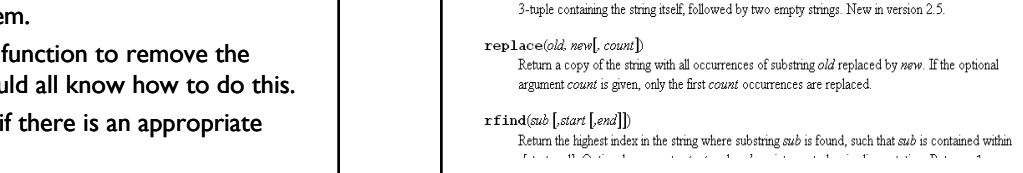
- If string is not found, returns -1

## Replacing parts of strings

- Common problem with number strings (you might encounter in assignment) - there's commas inside them.
- We could write a function to remove the commas. You should all know how to do this.
- Or, we could see if there is an appropriate string method….

## The replace method

- Google "Python string methods"



separator, the separator itself, and the part after the separator. If the separator is not found, return a 3-tuple containing the string itself, followed by two empty strings. New in version 2.5.

`replace`(*old*, *new*[, *count*])
Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

`rfind`(*sub* [,*start* [,*end*]])
Return the highest index in the string where substring *sub* is found, such that *sub* is contained within

## The replace method

```
string = '407,018'
popString = string.replace(',', '')
population = int(popString)
```

- Replaces all copies of the first argument with the second.
- Here, replaces all commas with the empty string; that is, eliminates commas.

```
s = 'Flinch'
s = s.replace('Fl', 'Gr')
```

## Get data from file

- Get access to the file:

```
inFile = open('article.txt', 'r')
```

variable to **refer to file**

**name** of file

**read** – it's an **input** file; similarly **w** for **write,** i.e., an output file

## File type data

- File is a new kind of data.
- Mostly use it with file methods.
- Variables are in your computer's memory. Where are the files?

## After we're done...

- **close** method releases file after we're done.

```
inFile.close()
```

variable we have used to **refer to file.**

## Reading one Line

- **readline** method returns lines of file as strings.

```
line = inFile.readline()
```

variable to hold **string** return value

variable **referring to file**

## Typical file reading loop

```
inFile = open('article.txt','r')
line = inFile.readline()
while line != '':
    process a line
    line = inFile.readline()
inFile.close()
```

- Use **while** because we don't know how many lines there are.

## Why not read whole file?

- Might be very big.
- Take the parts we need, while we read, throw out excess immediately.
- Not a problem in this course, but when you are dealing with, eg. all students at UC Davis, data can get very big.
- That's why programs can run out of memory.

## Organize into sentences

```
sentence = ''
for word in words:
        sentence = sentence + ' ' +word
        if '.' in word and '$' not in word:
                print sentence,'\n'
                sentence = ''
```