Outline

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

MCS 260 Lecture 44
Introduction to Computer Science
Jan Verschelde, 29 April 2016

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

general information

The exam will take place on Tuesday 3 May, from 8AM till 10AM, in BSB 389.

Bring your own paper to write the solutions.

If an emergency prevents you from participation, please contact me as soon as you are able to so we can schedule a makeup exam in finals week.

The final exam is comprehensive and covers the entire course.

Please review the posted answers to the midterms and the quizzes.

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

policies for the final exam

The final will be a a mix of computer literacy and Python scripting.

The exam will be closed book, no notes, and no computer.

Some of the types of literacy questions you could expect:

- Binary, decimal and hexadecimal number systems.
- Truth tables and electronic circuits
- Explain the difference, illustrate with an example.
- What is a good modular design?

This review contains some preliminary examples of questions which may help you prepare for the final exam.

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

run the script `literacy.py` ...

... and see if you can explain the terms *in your own words.*

Running the script provides inspiration for questions:

- `hash function`: what does hashing mean?
- `GPL`: difference with public domain?
- `linker`: its relation with a compiler?
- `segmentation`: is it the same as pagination?
- `complexity class`: can your problem be in a bad class?
- `fork a process`: what is the analogue for a thread?
- `beta testing`: is this whitebox or blackbox, or both?
- `portable code`: why is it important?
- `language generation`: how many generations do we know?
- `socket`: what is a socket good for?

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

strings and list comprehensions

Generate a random password of eight characters long.

A password consist of

- lower case letters of the alphabet,
  situated in the range from 97 to 122 of the ASCII code; and
- decimal digits in the range from 0 to 9.

Every password has 2 digits and 6 letters, placed at random.

Use list comprehensions to solve this problem.

generating a random password

```
from random import randint, shuffle

NUMBERS = [randint(0, 9) for k in range(2)]
LETTERS = [randint(ord('a'), ord('z')) \
    for k in range(6)]

STRLET = [chr(x) for x in LETTERS]
STRNUM = [str(x) for x in NUMBERS]

CHARACTERS = STRLET + STRNUM

shuffle(CHARACTERS)

PASSWORD = ''.join(CHARACTERS)
```

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

while and for loops

Consider the following piece of code:

```
L = [1,2,3,4]; x = 2
k = len(L)-1;  y = L[k]
while k > 0:
    k = k - 1
    y = y*x + L[k]
```

1. Make a table with the values for `k` and `y` at the start and for each time after the execution of the body in the loop.
2. Replace the `while` by a `for` loop which computes the same values for `y`.
   (*Hint:* `range(4,-1,-1)` returns `[4,3,2,1,0]`).

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

search the word that occurs most frequent in a file

Assume that all words in a text file are separated by spaces.

Describe an algorithm to find in a text file the word that occurs with the highest frequency.

The algorithm must return that word and the number of times it occurs.

What data structure(s) will you use?

review of for the final exam

1. final exam on Tuesday 3 May 2016, at 8AM, in BSB 389
   - general information
   - policies for the final exam
2. some example questions on computer literacy
   - run the script `literacy.py`
3. some example questions on Python scripting
   - strings and list comprehensions
   - while and for loops
   - working with files: find the most frequent word
   - GUI design, layout, and methods

GUI design, layout, and methods

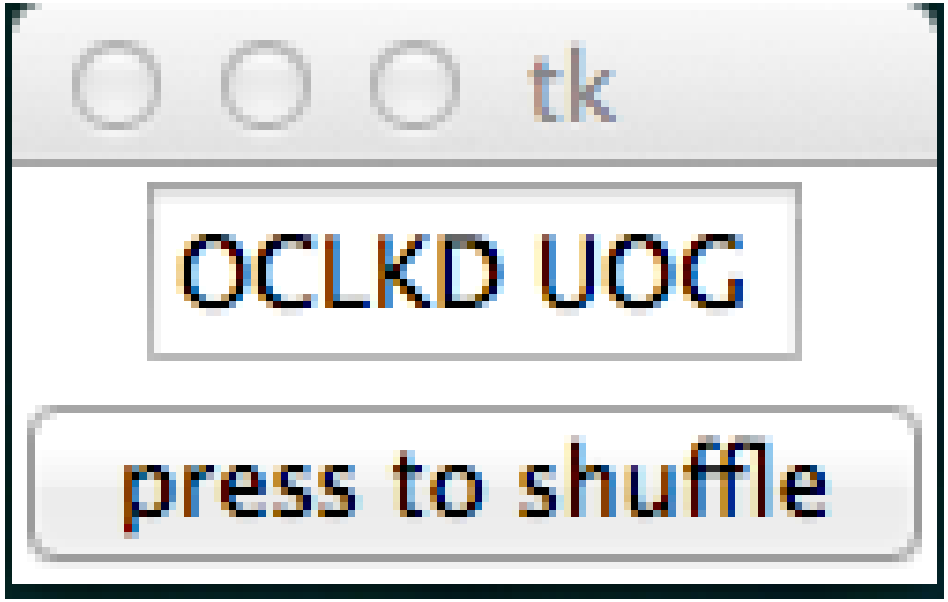
Give code for a GUI to shuffle the letters in a message.

A scrambled message is shown in an entry widget.

Each time the user presses the button,
the letters in the message are shuffled.

1. Draw the design of the GUI.
   Give the names of the widgets.
2. Write code for the constructor.
3. Define the method in the GUI.

design of the GUI



The grid in the GUI is defined as follows:

1. There is only one column in the GUI.
2. There are two rows:
   1. row 0: Entry widget to display the message;
   2. row 1: Button widget to shuffle the message.

the documentation strings of the class

```
from Tkinter import Tk, Entry, END, Button
from random import shuffle

class Message(object):
    """
    GUI to shuffle a message.
    """
    def __init__(self, wdw, message) :
        """
        One entry widget to display the message,
        above the button to shuffle the message.
        """
    def show(self):
        """
        Shuffles the riddle and places the
        shuffled letters in the entry widget.
        """
def main():
    """
    Defines the message and launches the GUI.
    """
```

code for the constructor

```
def __init__(self, wdw, message) :
    """
    One entry widget to display the message,
    above the button to shuffle the message.
    """
    self.riddle = message
    self.ent = Entry(wdw, width=len(message)+1)
    self.ent.grid(row=0)
    self.btt = Button(wdw, text='press to shuffle', \
        command = self.show)
    self.btt.grid(row=1)
    self.show()
```

code for the method to shuffle

```
def show(self):
    """
    Shuffles the riddle and places the
    shuffled letters in the entry widget.
    """
    shuffle(self.riddle)
    self.ent.delete(0, END)
    display = ''.join(self.riddle)
    self.ent.insert(0, display)
```