1. Carl Rustad, hours on Thursdays, 7-10pm in TCL 206. Homework 4 in today. Tonight: Midterm study session, here, 7:30-9:30pm. Homework 5 due on Monday. Exam on Tuesday evening, here, 6:00-7:30pm or 7:30-9:00pm.

2. Questions?

3. The try/except statement.
   (a) Has the form
   
   ```python
   try:
   <possibly faulty suite>
   except <error>:
   <cleanup suite>
   ```
   
   (b) The <possibly faulty suite> is a collection of statements that has the potential to fail, with error. If <error> occurs, the <cleanup suite> of statements is executed.
   
   (c) You can have more than one except, handling different types of errors.
   
   (d) You can ignore the error using pass as the <cleanup suite>.
   
   (e) Example: if you don't have a file named hello, then
   
   ```python
   try:
   open('hello')
   except FileNotFoundError:
   print('You have no file "hello"')
   ```
   
   prints
   
   You have no file "hello".

4. Recall: Generators.
   (a) Are identified with by the use of yield. Produce computations on demand with next.
   (b) Are also the result of comprehensions in parens:
   
   ```python
   >>> g = (i*i for i in range(1,10))
   >>> next(g)
   1
   >>> next(g)
   4
   >>> next(g)
   9
   ```
   
   Notice that, if the source of the for loop is infinite, g will be unending as well. Wowza.
5. Iterators.

(a) One can *iterate across* an object \( o \) using an *iterator*. An iterator is a generator that produces successive items from \( o \).

(b) You can ask \( o \) for its iterator with \( \text{it} = \text{iter}(o) \); it generates values, as you might expect, with \( \text{next(it)} \).

(c) Each iterator is created fresh and works independently of other iterators over \( o \).

(d) If \( g \) is a generator, then \( \text{iter}(g) \) is \( g \).

(e) Be careful: if \( o \) is mutable, you should think carefully about modifying it while you are iterating across its values.

6. Recall: Details of a for loop.

(a) A *for* loop iterates across some object, \( o \). For example:

```python
for item in l:  # l is a list of integers
    if item < 0:
        print(item)
```

prints the negative integers of \( l \).

(b) The *for* loop is simply a *while* loop, driving an iteration within a *try*-*except* statement. The above is really:

```python
try:
    it = iter(l)
    while True:
        item = next(it)
        print(item)
    except StopIteration:
        pass
```

Wowza.

7. Some new iterators.

(a) Digits of a number, base \( d \).

(b) Rearrangements of a list. Idea: take the first element of the list and insert it at every position in every rearrangement of the remaining elements.

(c) Subsets of a list. Idea: there are \( 2^n \) subsets of a list with \( n \) elements. They correspond to the subsets of digits that are one in numbers encountered when counting from 0 to \( 2^n - 1 \).

(d) Random subsets of a list. Idea: pick a random number.

(e) Application (due to Bob Floyd): How do you stack a collection of blocks with faces with areas 1 through \( n \) into two equal height towers?

\( \star \)