Lists so far:

Creating

Iterating

Indexing and slicing

Lists today:

List-related functions

List comprehensions

The `in` keyword

List-related functions

These functions get information from iterables (like lists!):

- `all()`
- `any()`
- `iter()` (optional)
- `len()`
- `max()`
- `min()`
- `next()` (optional)
- `sorted()`

Try using the `help()` function to learn a bit about each of these functions...

Exercise 3

Subject	Course
CHEM	1050
ENGI	1010, 1020, 1030, 1040
ENGL	1090
MATH	1001, 2050
PHYS	1051

*Calculate the Engineering One **average** and **promotability** of a student, given their Engineering One grades.*

Let's do this exercise again, this time using _____ to make for a simpler, more re-usable implementation.

Addition operator:

We can _____ lists together using the ____+_____. This is known as _____ _____, and it's one way to make lists _____.

A very common pattern:

```
total = 0
for x in something:
    if x > something_else:
        total += x
```

```
passing_grades = []
for grade in courses:
    if grade >= 55:
        passing_grades += [grade]
```

```
aggregate = ...
for x in something:
    if f(x):
        aggregate += g(x)   # or *=, or ...
```

A common solution

List comprehension: *an elegant tool for a civilized age*

- syntax for constructing a list from something iterable
- Can *filter* and *transform* individual elements

e.g., The first ten squares of even numbers:

```
squares = []
for i in range(2, 21):
    if i % 2 == 0:
        squares += [i*i]
```

List comprehensions

Syntax:

- `[` — opening bracket
- *expression*
- `for` *loop parameter* `in` *iterable*
- `if` *condition*
- `]` — closing bracket

```
[i*i for i in range(2, 21) if i%2 == 0]
```

A familiar problem

Problem 1 from Project Euler:

Find the sum of all the multiples of 3 or 5 below 1000.

```
total = 0

for n in range(1000):
    if n % 3 == 0 or n % 5 == 0:
        total += n
```

```
total = sum([n for n in range(1000) if n % 3 == 0 or n % 5 == 0])
```

More list comprehensions

Possible to get more complex:

- Can have more than one `for` loop:

```
[i+j for i in range(10) for j in range(10,20)]
```

- Not required for this course!
- Further details available at python.org (5.1.3 in the Data Structures tutorial)

One more thing

Remember the `in` keyword?

```
for i in range(10):
```

Can also use `in` as an operator!

```
if 100 in grades:
    print('Wow, well done!')
```

```
is_cool_guy = 'Jon' in username
if letter in 'EAIONRTLSU': # ...
```


This new syntax may be helpful for assignment 2...

Summary

How cool are lists?

List-related functions

List comprehensions

`in` keyword