CIS 210

Finishing / Starting

• List comprehensions, cont'd
• Intro to user-defined classes
• Programming environments
• Summing up

Final Exam Review – all labs (Q/A)

“When you express your understanding in code, you debug your brain.”

CIS 210
List Comprehensions

S = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

T is a list of x such that x is a member of S and x is even

{x | x ∈ S and x is even}  #mathematical

T = []  #procedural
for x in S:
    if even(x):
        T.append(x)

T = [x for x in S if even(x)]  #declarative (filter pattern)

CIS 210
List Comprehensions

def mode(alist):
    '''(list) -> list'''
    countd = genFreqTable(alist)
    countli = countd.values()
    maxct = max(countli)

    modeli = []
    # there may be more than one mode
    for k in countd:
        if countd[k] == maxct:
            modeli.append(k)

    return modeli

CIS 210
List Comprehensions

countd = {'CIS': 4, 'UNDL': 3}  # map pattern

doubleli = []
for k in countd:
    if countd[k] == maxct:
        modeli.append(k)

modeli = [k for k in countd if countd[k] == maxct]
List comprehensions

```python
countd = {'CIS': 4, 'UNDL': 3}

# map pattern
doubleli = [countd[k] * 2 for k in countd]

>>> doubleli
[8, 6]
```

List comprehensions are a concise way to create lists. The general syntax is:

```
[<expression> # mapping, etc. 
  for <item> in <sequence> ... # other collection(s) 
  if <condition>] # filtering (if needed)
```

Each item in the new list is the result of applying a given operation (<expression>) to a value (<item>) from a sequence (<sequence>).

```
modelli = [k for k in countd if countd[k] == maxct]
```

Replace the `??` with the result of executing the following code (indicate ‘error’ if the result would be a Python error message):

```
def addlist(m, n):
    '''(list, list) -> None
    Exam function.  
    '''
    for i in range(len(m)):
        if i < len(n):
            m[i] += n[i]
    return None

def q9(x, y):
    '''(list, list) -> None'''
    z = x
    addlist(x, y)
    print(x); print(y)
    w = []
    for item in y:
        w.append(item / 10)
    print(w)
    return None
```
The only way to learn is by practice (with advice). That's how you learn to ride a bike and that's how you learn to do anything.

A type, or class, is a template, for objects. Objects are instances of a class.

Every Python object is an instance of a class:

```python
>>> y = 'abc'
>>> y = str(4)
>>> type(y)  # return the type of object
<class 'str'>
>>> y  # return the value of the object
'abc'
>>> id(y)  # return the object's address in memory
4320804568
>>> y.count('a')  # return the number of times the character 'a' appears
4

>>> y  # return the value of the object
'4'
>>> id(y)  # return the object's address in memory
4298926768
```

A Python user-defined class

```python
class Turtle:
>>> t1 = Turtle()
>>> t2 = Turtle()
```

- Turtle is a constructor method for class Turtle
- Turtle() instantiates a Turtle object
- t1 and t2 are instances of the class Turtle (turtle objects)
```python
>>> t1
<turtle.Turtle object at 0x102692a90>

>>> type(t1)
<class 'turtle.Turtle'>

>>> id(t1)
4335413904
```

```python
>>> t1.heading()
>>> t1.pos()
>>> t2.heading()
>>> t2.pos()
>>> t1.color()
>>> t2.shape()

>>> t1.fd(100)
>>> t2.fd(200)
```

Recall: Python data types (objects):

- **type**: range of values → attributes
  - object methods (operations)
  - object descriptors
- **value(s)**
- **id**

```python
>>> dir(Turtle)
>>> dir(t1)

>>> t1.heading() >>> t1.seth(75)
>>> t1.pos() >>> t1.setpos(0, 0)
>>> t2.heading() >>> t1.shape('turtle')
>>> t2.pos() >>> t2.color('blue')
>>> t1.color() >>> t2.shape()
```

**CIS 210**

primitive objects – single value

collections objects – strings, lists, tuples, dicts

other Python objects – functions, turtle

**user-defined objects**

**CIS 210**

**Classes and Objects**

Creating our own classes –

Another way to extend the language

as with functions:
  - names
  - encapsulates

now data types: **Object-oriented programming**
Classes and Objects

Creating our own data types, or classes –

To create our own objects, or instances –

Example: Point

```python
class Point:
    '''Represents a point in 2-d space.'''
>>> p1 = Point()
>>> p1
<__main__.Point object at 0x15695d0>
```

special constructor method `__init__` initializes class names and values:

```python
class Point:
    '''Represents a point in 2-d space.'''
def __init__(self, x = 0, y = 0):
    self.x = x
    self.y = y
>>> p1 = Point()
>>> p2 = Point(3, 4)
```

Add methods to get and set (mutate) attribute values:

```python
class Point:
    '''Represents a point in 2-d space.'''
def __init__(self, x = 0, y = 0):
    self.x = x
    self.y = y
def getx(self):
    return self.x
def setx(self, newx):
    self.x = newx
    return None
def gety(self):
    return self.y
def sety(self, newy):
    self.y = newy
    return None
```

```python
>>> p2 = Point(3, 4)
>>> p2.getx()
4
>>> p2.setx(5)
>>> p2.getx()
5
>>> p2.gety()
4
```
Classes and Objects

```python
>>> p1 = Point()  
>>> type(p1)
<class '__main__.Point'>
```

```python
>>> p2 = Point(3, 4)  
>>> id(p1)
4389818152
```

```python
>>> p1.getx()  
>>> Point.getx(p2)
0
```

```python
>>> p1.sety(100)  
>>> p1.gety()
100
```

Other special methods: `__str__` and `__add__`, for example

```python
>>> ctr = 1  
>>> ctr.__add__(1)  
>>> ctr + 1
```

```python
>>> s = 'abc'  
>>> s.__add__('def')  
>>> s + 'def'
```

`__add__` is a special method used by the `+` operator

its behavior varies per object type

```
CIS 210

Classes and Objects

Other special methods: `__str__` and `__add__`, for example

```python
>>> ctr = 1  
>>> ctr = ctr.__add__(1)  
>>> ctr = ctr + 1
```

```python
>>> s = 'abc'  
>>> s.__add__('def')  
>>> s + 'def'
```

`__add__` is a special method used by the `+` operator

its behavior varies per object type

```
CIS 210

Classes and Objects

__str__ is a special method used to print the object using print

```python
__str__ should return a string
```

```python
class Point(object):
    '''Represents a point in 2-d space. '''
    def __str__(self):
        #return '***{}, {}***'.format(self.x, self.y)
        return f'***{self.x}, {self.y}***'
```

```python
>>> p4 = Point(0, 100)  
>>> print(p4)
*** 0, 100 ***
```

```
CIS 210

Classes and Objects

__str__ is a special method used to print the object using print

```python
class Point(object):
    '''Represents a point in 2-d space. '''
    def __str__(self):
        return f'***{self.x}, {self.y}***'
```

```python
>>> p4 = Point(0, 100)  
>>> print(p4)
*** 0, 100 ***
```

```
CIS 210

Finishing / Starting

✓ List comprehensions
✓ Intro to user-defined classes
• Programming environments
• Summing up

Final Exam Review – all labs (Q/A)

_The only way to learn is by practice (with advice). That’s how you learn to ride a bike and that’s how you learn to do anything._ –Roger Shank
**Algorithms → Programs**

- **Editor** for writing programs in a particular language and according to style guidelines
- **Translator** (interpreter/compiler, linker) for executing program (and reporting results)
- **Other tools**, e.g., for testing and debugging

---

**An integrated development environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development, e.g.,
- source code editor that supports writing programs
- Shell (compiler, interpreter)
- debugger
- version control system
- class browser, an object browser, and a class hierarchy diagram, for use in object-oriented program development
- tools to manage GUI interface development

---

**Once upon a time...**

- switches
- cards
- teletype machines (command-line)
- video display terminals (command-line)
- Graphical User Interfaces (GUIs – point and click)
- **command line interface programs** (Terminal/cmd.exe)
Integrated Development Environment (IDE) v. Command-line programming

command line - shell* for operating system
text-based (pre-GUI)

*program that accepts commands as text input and translates them to appropriate functions in an underlying program, e.g., interpreter, OS

Integrated Development Environment (IDE) v. Command-line

Recall:
... 
def main():
    '''Square root comparison program driver.'"
sqrt_compare(25, 5)
sqrt_compare(25, 10)
...
return None

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

IDLE: <Run Module>

COMMAND LINE:
sysadmins-air:projects-F18 kfreeman$ python3.6 p22_sqrt_key.py
For 25 using 5 iterations:
mysqrt value is: 5.000023178253949
math lib sqrt value is: 5.0
This is a 0.0 percent error.
For 25 using 10 iterations:
mysqrt value is: 5.0
math lib sqrt value is: 5.0
This is a 0.0 percent error.
... 
sysadmins-air:projects-F18 kfreeman$

Integrated Development Environment (IDE) v. Command-line

def main(n, r):
    sqrt_compare(n, r)

def main():
    with open('test.txt') as t:
        tests = int(t.readline())
        for i in range(tests):
            line = t.readline()
            n = line[0]
            r = line[1]
            sqrt_compare(n, r)
            return None

def main():
    '''Square root comparison'''
sqrt_compare(25, 5)
sqrt_compare(25, 10)
... 
return None

def main():
    with open('test.txt') as t:
        tests = int(t.readline())
        for i in range(tests):
            line = t.readline()
            n = line[0]
            r = line[1]
            sqrt_compare(n, r)
            return None

def main(n, r):
    sqrt_compare(n, r)

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects

Integrated Development Environment (IDE) v. Command-line

sysadmins-air:projects-W18 kfreeman$ python3.6
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 03:03:55) 
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> >>> print('hello, world')
hello, world
>>> quit()
sysadmins-air:projects-F18 kfreeman$

pwd
/Users/kfreeman/Documents/cis210F18/F18-projects
Integrated Development Environment (IDE) v. Command-line

kfreeman$ python3.6 p22_sqrt_key.py 81 10

import sys
def main():
    '''Use arguments supplied at command line'''
    print(sys.argv)  # what is sys.argv?
    n = int(sys.argv[1])
    r = int(sys.argv[2])
    sqrt_compare(n, r)
    return None

kfreeman$ python3.6 p22_sqrt_key.py 81 10
['p22_sqrt_key.py', '81', '10']
For 81 using 10 iterations:
mysqrt value is: 9.0
math lib sqrt value is: 9.0
This is a 0.0 percent error.

# p22_wk10demo.txt
3
25,5
25,10
100,10

def main():
    '''Use arguments from a file: p22_wk10demo
$p3.6 p22_sqrt_key.py < p22_wk10demo.txt

    num_tests = int(input())  # first line of file is number of tests
    for i in range(num_tests):
        testli = input().split(',')  # form the list
        n = int(testli[0])
        r = int(testli[1])
        sqrt_compare(n, r)
    return None

Writing – executing – testing and debugging code:

IDE: GUI, integrated, handy tools in one place, overhead to learn (and then learn again). Great for development and testing, especially for large projects.

command line: Shell (text only), not integrated, portable built-in editors: e.g., emacs, vim, TextEdit, IDLE

Finishing / Starting

✓ List comprehensions
✓ Intro to user-defined classes
✓ Programming environments

• Summing up

Final Exam Review – all labs (Q/A)

The only way to learn is by practice (with advice). That’s how you learn to ride a bike and that’s how you learn to do anything.
CIS 210 / Welcome

Computational Problem Solving

Computational problem solving is an approach to problem solving that is inspired and constrained by the possibilities and limitations of computers and computing.

Python/programming toolkit so far

- Assignment statement
- Expressions
- Python conditionals – if
- Numeric data types (int, float) and operations (e.g., +, **, round, abs)
- Boolean data types and operations (e.g., <, and)
- String data types and operations (e.g., str, len, count, find, format), formatted strings
- Python collections: data types and operations – tuples, lists, dictionaries; comprehensions
- Python files and file processing
- Data type "coercion" functions (e.g., int, str, int, list, float)
- NoneType (None)
- Print/input
- Python Standard Library – math, turtle, random modules; import (if __name__ == '__main__')
- User-defined functions; function design; docstrings
- Intro to user-defined classes (types)
- IDLE interactive development environment; intro to programming environments
- Python introspection: help, dir, auto-complete
- Run-time checking of data and code: assert, try/except
- Python introspection – help, dir, auto-complete, type, id, instance, os.getcwd, os.chdir

CIS 210 Learning Outcomes

- Understand, develop, implement algorithms for computational problem solving;
- Use structured design and testing methods to develop and implement programs;
- Read, write, revise, document, test, and debug code;
- Demonstrate robust mental models of data representation and code execution;
- Demonstrate good understanding of a high level programming language;
- Introduce and/or implement a sampling of classic computer science problem domains and algorithms.