Contact

Dr. James Shackleford
shack@drexel.edu
Bossone 211

Office Hours: 3 – 4 pm (Tuesday)
Course Website: http://learn.dcollege.net

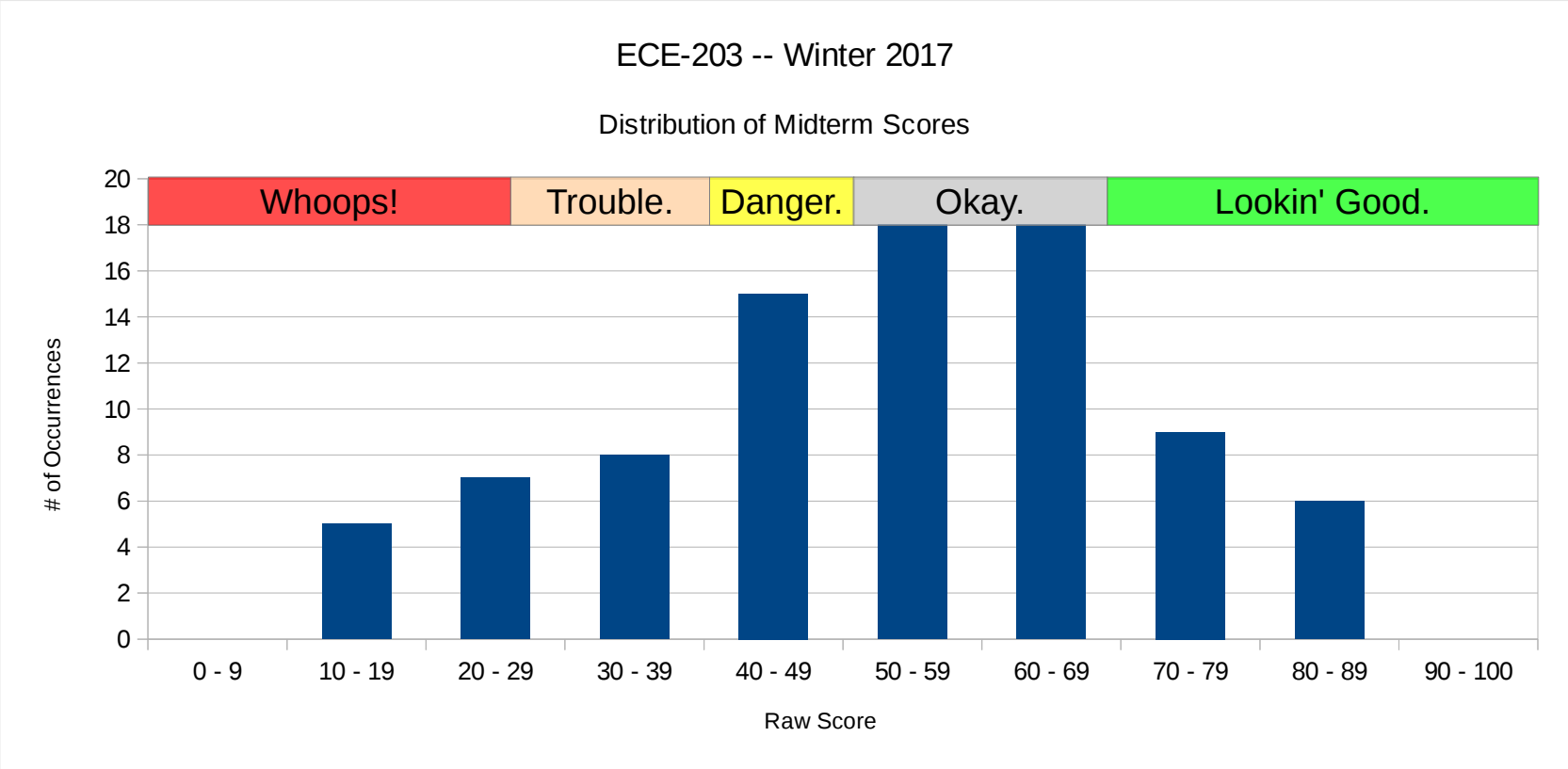
Textbook

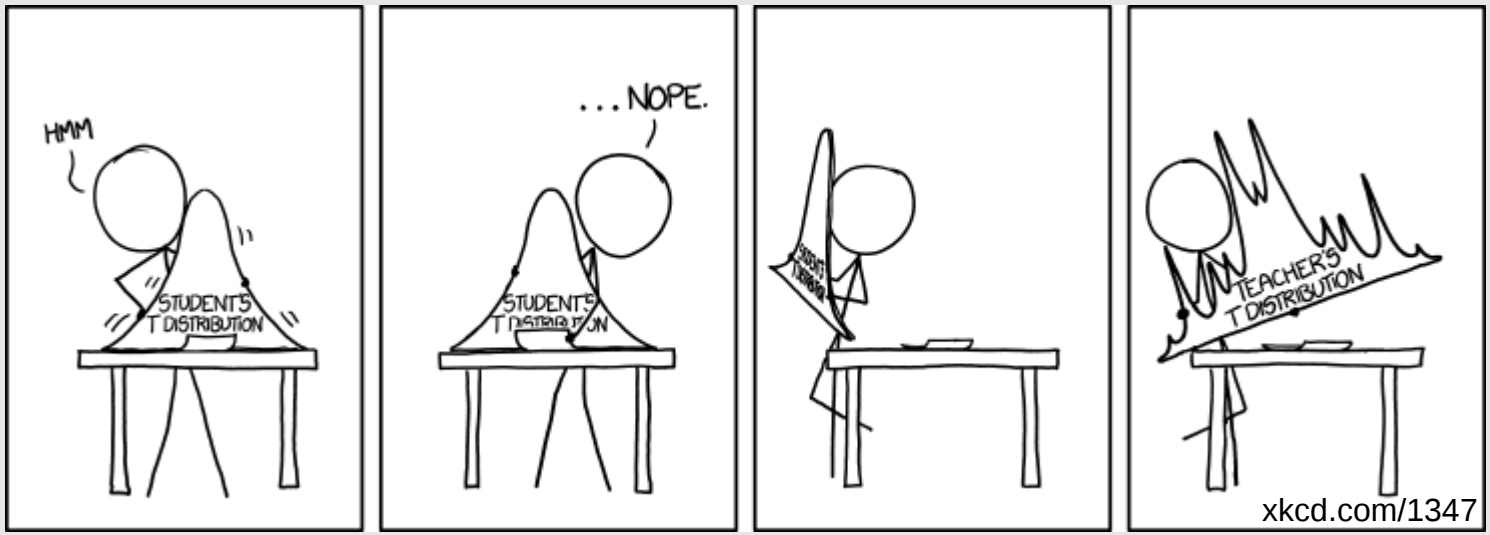
Think Python
by Allen Downey
O'Reilly Press, 2015
ISBN-13: 978-1449330729
(Freely available in PDF format, check course website)

Grading

- 10% In-lab Programming Assignments
- 10% Take-Home Programming Assignments
- 35% Mid-term Exam
- 45% Final Exam

Midterm Grades





New Topic

Dictionaries

(a.k.a. Associative Arrays)

Dictionaries

```
>>> employee = {'job': 'boss', 'age': 38, 'pay': 72000}
>>> print employee['job']
boss
>>> print employee['age']
38
>>> print employee['pay']
72000
```

Be Careful! Stored Order Can Be Different!

```
>>> employee
{'pay': 72000, 'job': 'boss', 'age': 38}
```

Default iterator only provides keys

```
>>> for i in employee:
...   print '%s: %s' % (str(i), str(employee[i]))
...
pay: 72000
job: boss
age: 38
```

The .items() method provides list of (key, values) tuples

```
>>> employee.items()
[('pay', 72000), ('job', 'boss'), ('age', 38)]
>>> for key, val in employee.items():
...     print '%s: %s' % (str(key), str(val))
...
pay: 72000
job: boss
age: 38
```

NOTE:
.iteritems() is the generator version of .items()

Creating an Empty Dictionary

```
# How to create an empty dictionary
my_dict = {}

# OR
my_dict = dict()

```

Adding to an Existing Dictionary

```
>>> my_dict = {}
>>> my_dict['color'] = 'blue'
>>> my_dict['process'] = 8923
>>> my_dict
{'color': 'blue', 'process': 8923}
```

Delete a key in a dictionary

```
>>> my_dict = {}
>>> my_dict['color'] = 'blue'
>>> my_dict['process'] = 8923
>>> my_dict
{'color': 'blue', 'process': 8923}
>>> del my_dict['color']
>>> my_dict
{'process': 8923}
```

Copy a dictionary

```
>>> my_dict
{'color': 'blue', 'process': 8923}
>>> my_copy = dict(my_dict)
>>> my_copy
{'color': 'blue', 'process': 8923}
```

New Topic

Basic Exception Handling

Basic Exception Handling

Code:

```
1  my_list = range(5)
2
3  print 'for-loop:'
4  for i in my_list:
5      print i
6
7  print 'raw iter() w/ try-except:'
8  iterator = iter(my_list)
9  while True:
10     try:
11         i = iterator.next()
12     except:
13         break
14     else:
15         print i
16
17 del iterator
```

Output:

```
for-loop:
0
1
2
3
4
raw iter() w/ try-except:
0
1
2
3
4
```

Basic Exception Handling

Code:

```
1  my_list = range(5)
2
3  print 'for-loop:'
4  for i in my_list:
5      print i
6
7  print 'raw iter() w/ try-except:'
8  iterator = iter(my_list)
9  while True:
10     try:
11         i = iterator.next()
12     except:
13         break
14     else:
15         print i
16
17 del iterator
```

try this

Output:

```
for-loop:
0
1
2
3
4
raw iter() w/ try-except:
0
1
2
3
4
```

Basic Exception Handling

Code:

```
1  my_list = range(5)
2
3  print 'for-loop:'
4  for i in my_list:
5      print i
6
7  print 'raw iter() w/ try-except:'
8  iterator = iter(my_list)
9  while True:
10     try:
11         i = iterator.next()
12     except:
13         break
14     else:
15         print i
16
17 del iterator
```

if i = iterator.next( ) raises an exception, do this

Output:

```
for-loop:
0
1
2
3
4
raw iter() w/ try-except:
0
1
2
3
4
```

Basic Exception Handling

Code:

```
my_list = range(5)

print 'for-loop:'
for i in my_list:
    print i

print 'raw iter() w/ try-except:'
iterator = iter(my_list)
while True:
    try:
        i = iterator.next()
    except:
        break
    else:
        print i

del iterator
```

if i = iterator.next( ) DOES NOT raise an exception, do this

Output:

```
for-loop:
0
1
2
3
4
raw iter() w/ try-except:
0
1
2
3
4
```