Corrections

Page 5: Second full paragraph should read

The static semantics defines which syntactically valid strings have a meaning. Consider, for example, the strings “He run quickly” and “I runs quickly.” Each is of the form <pronoun> <regular verb> <adverb>, which is a syntactically acceptable sequence. Nevertheless, neither is valid English, because of the rather peculiar rule that for a regular verb when the subject of the phrase is first or second person the verb does not have an “s” at the end, but when the subject is third person plural it does. These are examples of static semantic errors.

Page 11, Figure 2.1: \(i//j\) Should be described as “floor division,” not “integer division”

Page 17: Add to the end of the first full paragraph “A line of code that is too long to read easily on a screen can be broken into multiple lines on the screen by ending each line other than the last one with a backslash.”

Page 51: factR in Figure 4.5 should be factR

Page 67: The docstring for findExtremeDivisors should read

"""Assumes that n1 and n2 are positive ints
Returns a tuple containing the smallest common divisor > 1 and the largest common divisor of n1 and n2. If no common divisor other than 1, returns (None, None)"

Page 70, Figure 5.2: The lower “Univs” should be “Univs1.”

Page 114: Second sentence in 2nd full paragraph should read “We could also print the value of s by writing print(s.__str__()) or even print(IntSet.__str__(s)), but using those forms is less convenient.”

Page 205: The code for fastFib in Figure 13.2 should be

```python
def fastFib(n, memo = None):
    """Assumes n is an int >= 0, memo used only by recursive calls
    Returns Fibonacci of n""
    if memo == None:
        memo = {}
    if n == 0 or n == 1:
        return 1
    try:
        return memo[n]
    except KeyError:
        result = fastFib(n-1, memo) + fastFib(n-2, memo)
        memo[n] = result
    return result
```
The start of fastMaxVal in Figure 13.7 should be

```python
def fastMaxVal(toConsider, avail, memo = None):
    """Assumes toConsider a list of items, avail a weight
    memo supplied by recursive calls
    Returns a tuple of the total value of a solution to the
    0/1 knapsack problem and the items of that solution""
    if memo == None:
        memo = {}
```

ydist in distFrom should be yDist

paragraph before figure should say that some “ranges of values” are more common ...

print scipy.integrate.quad(abs, 0, 5)[0] should be
print scipy.integrate.quad(gaussian, -2, 2, (0, 1))[0]

print scipy.integrate.quad(gaussian, -2, 2, (0, 1))[0] should be
print scipy.integrate.quad(gaussian, -2, 2, (0, 1))[0]

In the first sentence of the paragraph before first equation, “to to” should be
replaced by “to”
The following sentence should be added after the first sentence of that paragraph, “Consider a sample containing the three examples: 100, 200, and 300.”

Page 333, Figure 19.5 should start with the line
from scipy import stats

Page 416: Second full paragraph, last sentence: model.coef[1][0] should be
model.coef_[1][0].

Page 420, Figure 24.14, 6th line currently reads labels.append(15). It should read
labels.append('D').

Page 421, first paragraph: “Since the weights are negative” should read “Since the weights are positive.”
def getTitanicData(fname):
    data = {}
    data['class'], data['survived'], data['age'] = [], [], []
    data['gender'], data['name'] = [], []
    f = open(fname)
    line = f.readline()
    while line != '':
        split = line.split(',')
        data['class'].append(int(split[0]))
        data['age'].append(float(split[1]))
        if split[2] == 'M':
            data['gender'].append(1)
        else:
            data['gender'].append(0)
        data['survived'].append(int(split[3]))  # 1 = survived
        data['name'].append(split[4:])
        line = f.readline()
    return data

def buildTitanicExamples(fileName):
    data = getTitanicData(fileName)
    examples = []
    for i in range(len(data['class'])):
        p = Passenger(data['class'][i], data['age'][i],
                      data['gender'][i], data['survived'][i],
                      data['name'][i])
        examples.append(p)
    return examples

examples = buildTitanicExamples('TitanicPassengers.txt')