Sets and Dictionaries

Examples
How early was each kind of bird seen?
How early was each kind of bird seen?

2010-07-03 05:38 loon
2010-07-03 06:02 goose
2010-07-03 06:07 loon
2010-07-04 05:09 ostrich  # hallucinating?
2010-07-04 05:29 loon

Sets and Dictionaries

Examples
How early was each kind of bird seen?

2010-07-03  05:38    loon
2010-07-03  06:02    goose
2010-07-03  06:07    loon
2010-07-04  05:09    ostrich  # hallucinating?
2010-07-04  05:29    loon

Want the minimum of all times associated with a bird
How early was each kind of bird seen?

2010-07-03  05:38  loon
2010-07-03  06:02  goose
2010-07-03  06:07  loon
2010-07-04  05:09  ostrich  # hallucinating?
2010-07-04  05:29  loon

Want the minimum of all times associated with a bird

Use bird name as dictionary key
How early was each kind of bird seen?

2010-07-03  05:38  loon
2010-07-03  06:02  goose
2010-07-03  06:07  loon
2010-07-04  05:09  ostrich  # hallucinating?
2010-07-04  05:29  loon

Want the minimum of all times associated with a bird
Use bird name as dictionary key
And earliest observation time as value
def read_observations(filename):
    '''Read data, return [(date, time, bird)...].'''

    reader = open(filename, 'r')
    result = []

    for line in reader:
        fields = line.split('#')[0].strip().split()
        assert len(fields) == 3, 'Bad line "%s"' % line
        result.append(fields)

    return result
def read_observations(filename):
    '''Read data, return [(date, time, bird)...].'''

    reader = open(filename, 'r') ← Setup
    result = []

    for line in reader:
        fields = line.split('#')[0].strip().split()
        assert len(fields) == 3, 'Bad line "%s"' % line
        result.append(fields)

    return result
def read_observations(filename):
    '''Read data, return [(date, time, bird)...].'''

    reader = open(filename, 'r')
    result = []

    for line in reader:
        fields = line.split('#')[0].strip().split()
        assert len(fields) == 3, 'Bad line "%s"' % line
        result.append(fields)

    return result
def read_observations(filename):
    '''Read data, return [(date, time, bird)...].'''

    reader = open(filename, 'r')
    result = []

    for line in reader:
        fields = line.split('#')[0].strip().split()
        assert len(fields) == 3, 'Bad line "%s"' % line
        result.append(fields)

    return result

Check that the data might be right
def read_observations(filename):
    '''Read data, return [(date, time, bird)...].'''

    reader = open(filename, 'r')
    result = []

    for line in reader:
        fields = line.split('#')[0].strip().split()
        assert len(fields) == 3, 'Bad line "%s"' % line
        result.append(fields)

    return result
def earliest_observation(data):
    '''How early did we see each bird?'''

    result = {}
    for (date, time, bird) in data:
        if bird not in result:
            result[bird] = time
        else:
            result[bird] = min(result[bird], time)

    return result
def earliest_observation(data):
    '''How early did we see each bird?'''

    result = {}
    for (date, time, bird) in data:
        if bird not in result:
            result[bird] = time
        else:
            result[bird] = min(result[bird], time)

    return result
def earliest_observation(data):
    """How early did we see each bird?"""

    result = {}
    for (date, time, bird) in data:
        if bird not in result:
            result[bird] = time
        else:
            result[bird] = min(result[bird], time)

    return result
def earliest_observation(data):
    '''How early did we see each bird?'''

    result = {}
    for (date, time, bird) in data:
        if bird not in result:
            result[bird] = time
        else:
            result[bird] = min(result[bird], time)

    return result
def earliest_observation(data):
    '''How early did we see each bird?'''

    result = {}
    for (date, time, bird) in data:
        if bird not in result:
            result[bird] = time
        else:
            result[bird] = min(result[bird], time)

    return result
What birds were seen on each day?
What birds were seen on each day?

Very similar structure...
What birds were seen on each day?
Very similar structure...
...but use a set to record one or more birds, rather than taking the minimum time
def birds_by_date(data):
    '''Which birds were seen on each day?'''

    result = {}
    for (date, time, bird) in data:
        if date not in result:
            result[date] = set()
            result[date].add(bird)

    return result

Sets and Dictionaries

Examples
def birds_by_date(data):
    '''Which birds were seen on each day?'''

    result = {}
    for (date, time, bird) in data:
        if date not in result:
            result[date] = set()
            result[date].add(bird)

    return result
def birds_by_date(data):
    '''Which birds were seen on each day?'''

    result = {}
    for (date, time, bird) in data:
        if date not in result:
            result[date] = set()
        result[date].add(bird)

    return result

Process each tuple in turn
def birds_by_date(data):
    '''Which birds were seen on each day?'''

    result = {}
    for (date, time, bird) in data:
        if date not in result:
            result[date] = set()
            result[date].add(bird)

    return result

Make sure there is space to record this bird

Sets and Dictionaries
Examples
def birds_by_date(data):
    '''Which birds were seen on each day?'''

    result = {}
    for (date, time, bird) in data:
        if date not in result:
            result[date] = set()
            result[date].add(bird)  # Record this bird

    return result
'2010-07-03'

'loon'
Sets and Dictionaries

Examples

'2010-07-03'
'loon'
'goose'

'2010-07-03' 05:38  loon
2010-07-03  06:02  goose
2010-07-03  05:38    loon
2010-07-03  06:02    goose
2010-07-03  06:07    loon
Sets and Dictionaries

Examples

2010-07-03  05:38  loon
2010-07-03  06:02  goose
2010-07-03  06:07  loon
2010-07-04  05:09  ostrich  # hallucinating?
Sets and Dictionaries

Examples

2010-07-03  05:38    loon
2010-07-03  06:02    goose
2010-07-03  06:07    loon
2010-07-04  05:09    ostrich  # hallucinating?
2010-07-04  05:29    loon
Which bird did we see least frequently?
Which bird did we see least frequently?

Actually, which *birds*, since two or more could be tied for the low score
Which bird did we see least frequently?
Actually, which *birds*, since two or more could be tied for the low score

Two-pass algorithm
Which bird did we see least frequently?
Actually, which *birds*, since two or more could be tied for the low score

Two-pass algorithm

- Find the minimum value in the dictionary
Which bird did we see least frequently?
Actually, which *birds*, since two or more could be tied for the low score

Two-pass algorithm
- Find the minimum value in the dictionary
- Find all keys with that value
Which bird did we see least frequently?
Actually, which *birds*, since two or more could be tied for the low score

Two-pass algorithm
- Find the minimum value in the dictionary
- Find all keys with that value

Combine these calculations in a one-pass algorithm
Which bird did we see least frequently? Actually, which *birds*, since two or more could be tied for the low score.

Two-pass algorithm
- Find the minimum value in the dictionary
- Find all keys with that value

Combine these calculations in a one-pass algorithm

Assume we already have a dictionary `counts` recording how often each kind of bird was seen.
def least_frequently_seen(counts):
    """Which bird(s) were least frequently seen?""

    result = set()
    number = 0
    for bird in counts:
        ...handle this bird...

    return result
def least_frequently_seen(counts):
    '''Which bird(s) were least frequently seen?'''

    result = set()
    number = 0
    for bird in counts:
        ...handle this bird...

    return result

    if len(result) == 0:
        result = {bird}
        number = counts[bird]
    elif counts[bird] < number:
        result = {bird}
        number = counts[bird]
    elif counts[bird] == number:
        result.add(bird)
def least_frequently_seen(counts):
    '''Which bird(s) were least frequently seen?'''

    result = set()
    number = 0
    for bird in counts:
        ...handle this bird...

    return result

    if len(result) == 0:
        result = {bird}
        number = counts[bird]
    elif counts[bird] < number:
        result = {bird}
        number = counts[bird]
    elif counts[bird] == number:
        result.add(bird)

Case 1: first bird (initializing data structures)
```python
def least_frequently_seen(counts):
    '''Which bird(s) were least frequently seen?'''
    result = set()
    number = 0
    for bird in counts:
        ...handle this bird...
    if len(result) == 0:
        result = {bird}
        number = counts[bird]
    elif counts[bird] < number:
        result = {bird}
        number = counts[bird]
    elif counts[bird] == number:
        result.add(bird)
    return result
```

Case 2: new minimum, so replace everything
def least_frequently_seen(counts):
    '''Which bird(s) were least frequently seen?'''

    result = set()
    number = 0
    for bird in counts:
        ...handle this bird...
        result.add(bird)

    return result

    if len(result) == 0:
        result = {bird}
        number = counts[bird]
    elif counts[bird] < number:
        result = {bird}
        number = counts[bird]
    elif counts[bird] == number:
        result.add(bird)

    Case 3: tied equal for minimum
{ 'loon' : 3, 'goose' : 1, 'ostrich' : 1 }

Before the loop
Case 1: first bird (initializing data structures)
```
{ 'loon' : 3, 'goose' : 1, 'ostrich' : 1 }
```

Case 2: new minimum, so replace everything
Case 3: tied equal for minimum
created by

Greg Wilson

July 2010