CSc 120
Introduction to Computer Programming II

CODE EXAMPLES 03
Some code from assg 4
Data structure:

{ park_name : ( area , [ flora_count, fauna_count ] ),
  ... }
""
File: biodiversity.py
Author: Saumya Debray
Purpose: Compute and print out biodiversity information about US National Parks.
""

### Positions of flora and fauna counts in the lists that keep track of the counts for each park.

\[
\text{FLORA} = 0 \\
\text{FAUNA} = 1
\]

### Mapping from categories in the input to flora/fauna

\[
\text{CATEGORIES} = \{ \\
\text{ 'Algae' : FLORA,} \\
\text{ 'Fungi' : FLORA,} \\
\text{ 'Nonvascular Plant' : FLORA,} \\
\text{ 'Vascular Plant' : FLORA,} \\
\text{ 'Amphibian' : FAUNA,} \\
\text{ 'Bird' : FAUNA,} \\
\text{ 'Crab/Lobster/Shrimp' : FAUNA,} \\
\text{ 'Fish' : FAUNA,} \\
\text{ 'Insect' : FAUNA,} \\
\text{ 'Invertebrate' : FAUNA,} \\
\text{ 'Mammal' : FAUNA,} \\
\text{ 'Reptile' : FAUNA,} \\
\text{ 'Slug/Snail' : FAUNA,} \\
\text{ 'Spider/Scorpion' : FAUNA } \}
\]
def main():
    """Process and print biodiversity information for national parks.

    Parameters: none

    Returns: none

    Pre-condition: none

    Post-condition: bio-diversity information printed out """

    info = read_biodiversity_info()
    print_biodiversity_info(info)
def read_biodiversity_info():
    """Read bio-diversity information from files and organize the data into a
dictionary.

Parameters: none

Returns: A dictionary mapping national park names to their area and
flora and fauna counts.

Pre-condition: none

Post-condition: The dictionary returned contains park area and
flora and fauna information from the input files.
    """

    info_dict = {}

    # read park information
    pfile_name = input()
pfile = open(pfile_name)

    for line in pfile:
        if line[0] == '#':
            continue
        park_info = line.split(',,')
        assert len(park_info) == 5
        park_name = park_info[0]
park_area = int(park_info[2])
        info_dict[park_name] = (park_area, [0, 0])
    pfile.close()

    # read species information
    sfile_name = input()
sfile = open(sfile_name)

    for line in sfile:
        if line[0] == '#':
            continue
        species_info = line.split(',,')
        assert len(species_info) >= 3
        park_name = species_info[0]
species_info = species_info[1]
        species_category = CATEGORIES[species_info]
        if park_name in info_dict:
            park_biodiversity_info = info_dict[park_name][1]
park_biodiversity_info[species_category] += 1
    sfile.close()

    return info_dict
def read_biodiversity_info():
    """Read bio-diversity information from files and organize the data into a dictionary.

    Parameters: none

    Returns: A dictionary mapping national park names to their area and flora and fauna counts.

    Pre-condition: none

    Post-condition: The dictionary returned contains park area and flora and fauna information from the input files.
    """

    info_dict = {}
    # read park information
    for line in pfile:
        if line[0] == '#':
            continue
        species_info = line.split(',')
        assert len(species_info) >= 3
        park_name = species_info[0]
        species_category = CATEGORIES(species_info[1])
        if park_name in info_dict:
            park_biodiversity_info = info_dict[park_name][1]
            park_biodiversity_info[species_category] += 1
        else:
            info_dict[park_name] = [1, {species_category: 1}]

    # read spec info
    for line in sfile:
        if line[0] == '#':
            continue
        species_info = line.split(',')
        assert len(species_info) >= 3
        park_name = species_info[0]
        species_category = CATEGORIES(species_info[1])
        if park_name in info_dict:
            park_biodiversity_info = info_dict[park_name][1]
            park_biodiversity_info[species_category] += 1
        else:
            info_dict[park_name] = [1, {species_category: 1}]

    return info_dict
def read_biodiversity_info():
    """Read biodiversity information from files and organize the data into a
dictionary.
Parameters: none
Returns: A dictionary mapping national park names to their area and
    flora and fauna counts.
Pre-condition: none
Post-condition: The dictionary returned contains park area and
    flora and fauna information from the input files.""

    info_dict = {}
    # read park information
    pfile_name = input()
pfile = open(pfile_name)
    for line in pfile:
        if line[0] == '#':
            continue
        park_info = line.split(',')
        assert len(park_info) == 5
        park_name = park_info[0]
park_area = int(park_info[2])
        info_dict[park_name] = (park_area, [0, 0])
pfile.close()

    # read species information
    sfile_name = input()
sfile = open(sfile_name)
    for line in sfile:
        if line[0] == '#':
            continue
        species_info = line.split(',')
        assert len(species_info) >= 3
        park_name = species_info[0]
species_category = CATEGORIES[species_info[1]]
        if park_name in info_dict:
            park_biodiversity_info = info_dict[park_name][1]
park_biodiversity_info[species_category] += 1
        else:
            info_dict[park_name] = (park_area, [species_category, 1])
sfile.close()

    return info_dict
# read species information
sfile_name = input()
sfile = open(sfile_name)

for line in sfile:
    if line[0] == '#':
        continue

    species_info = line.split(',' ,')
    assert len(species_info) >= 3

    park_name = species_info[0]
    species_category = CATEGORIES[species_info[1]]
    if park_name in info_dict:
        park_biodiversity_info = info_dict[park_name][1]
        park_biodiversity_info[species_category] += 1
    else:
        info_dict[park_name] = {species_category: 1}
        park_biodiversity_info = [park_name, species_category, 1]

sfile.close()
return info_dict
```python
info_dict = {}

# read park information
pfile_name = input()
pfile = open(pfile_name)

for line in pfile:
    if line[0] == '#':
        continue

    park_info = line.split(',
assert len(park_info) == 5

    park_name = park_info[0]
park_area = int(park_info[2])

    info_dict[park_name] = (park_area,
pfile.close()

# read species information
sfile_name = input()
sfile = open(sfile_name)

for line in sfile:
    if line[0] == '#':
        continue

    species_info = line.split(',
assert len(species_info) >= 3

    park_name = species_info[0]
species_category = CATEGORIES[species_info[1]]

    if park_name in info_dict:
        park_biodiversity_info = info_dict[park_name][1]
park_biodiversity_info[species_category] += 1

sfile.close()

return info_dict
```
Can this repetition be eliminated?

```python
info_dict = {}

# read park information
pfile_name = input()
pfile = open(pfile_name)
for line in pfile:
    if line[0] == '#':
        continue
    park_info = line.split(',',)
    assert len(park_info) == 5
    park_name = park_info[0]
park_area = int(park_info[2])
    info_dict[park_name] = (park_area)
pfile.close()

# read species information
sfile_name = input()
sfile = open(sfile_name)
for line in sfile:
    if line[0] == '#':
        continue
    species_info = line.split(',',)
    assert len(species_info) >= 3
    park_name = species_info[0]
    species_category = CATEGORIES[species_info[1]]
    if park_name in info_dict:
        park_biodiversity_info = info_dict[park_name][1]
        park_biodiversity_info[species_category] += 1
sfile.close()
return info_dict
```
def print_biodiversity_info(info):
    """Print out the bio-diversity density (per acre) of each national park

    Parameters: info is a dictionary mapping park names to area and
                flora/fauna counts

    Returns: none

    Pre-condition: info is a dictionary

    Post-condition: bio-diversity information printed out
    """

    assert type(info) == dict

    for park in info:
        area = info[park][0]
        flora = info[park][1][FLORA]
        fauna = info[park][1][FAUNA]

        assert area > 0 and flora >= 0 and fauna >= 0

        flora_density = flora / area
        fauna_density = fauna / area

        if flora == 0 and fauna == 0:
            print("{0} -- no data available".format(park))
        else:
            print("{0} -- flora: {1:f} per acre; fauna: {2:f} per acre".
                   format(park, flora_density, fauna_density))
Some style comments
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo))
    sinfo[i] = sinfo[i].split(',', ')
if sinfo[i][0][0] == '#':
    continue
else:
    if sinfo[i][1] in fauna and sinfo[i][0] in parks:
        parks[sinfo[i][0]][1][1] += 1
    elif ...
```python
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo))
    sinfo[i] = sinfo[i].split(',,')
if sinfo[i][0][0] == '#':
    continue
else:
    if sinfo[i][1] in fauna and sinfo[i][0] in parks:
        parks[sinfo[i][0]][1][1] += 1
    elif ...
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo)):
    sinfo[i] = sinfo[i].split(',,')
    if sinfo[i][0][0] == '#':
        continue
    else:
        if sinfo[i][1] in fauna and sinfo[i][0] in parks:
            parks[sinfo[i][0]][1][1] += 1
        elif ...
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo))
    sinfo[i] = sinfo[i].split(',,')
    if sinfo[i][0][0] == '#':
        continue
else:  # not necessary
    if sinfo[i][1] in fauna and sinfo[i][0] in parks:
        parks[sinfo[i][0]][1][1] += 1
    elif ...
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo)):
sinfo[i] = sinfo[i].split(',
if sinfo[i][0][0] == '#':
    continue

if sinfo[i][1] in fauna and sinfo[i][0] in parks:
parks[sinfo[i][0]][1][1] += 1
elif ...
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo))
    sinfo[i] = sinfo[i].split(',', ')
if sinfo[i][0][0] == '#':
    continue
if sinfo[i][1] in fauna and sinfo[i][0] in parks:
    parks[sinfo[i][0]][1][1] += 1
elif ...

Hard to understand
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo)):
sinfo[i] = sinfo[i].split(',',)
if sinfo[i][0][0] == '#':
    continue
if sinfo[i][1] in fauna and sinfo[i][0] in parks:
parks[sinfo[i][0]][1][1] += 1
elif ...

Hard to understand
repetitious
sinfo = input()
sinfo = open(sinfo).readlines()
for i in range(len(sinfo)):
    sinfo[i] = sinfo[i].split(',,')

park_name = sinfo[i][0]
if park_name[0] == '#':
    continue

if sinfo[i][1] in fauna and park_name in parks:
    parks[park_name][1][1] += 1
elif ...

Use meaningful variable names to make operations on complex structures easier to understand
sinfo = input()
sinfo = open(sinfo).readlines()  # No need for.readlines()

for i in range(len(sinfo)):
    sinfo[i] = sinfo[i].split(',', ')
park_name = sinfo[i][0]
    if park_name[0] == '#':
        continue

    if sinfo[i][1] in fauna and park_name in parks:
        parks[park_name][1][1] += 1

    elif ...
sinfo = input()
sfile = open(sinfo)
for line in sfile:
    sinfo = line.split(',
    park_name = sinfo[0]
if park_name[0] == '#':
    continue

if sinfo[1] in fauna and
   park_name in parks:
    parks[park_name][1][1] += 1
elif ...
Summary

• Use constants to make code more readable
  – lists, dictionaries, etc. can also be constants

• Clean up repetition in code after it's written
  – avoids embarrassment when showing code to class

• When operating on complex data structures:
  – use meaningful names for intermediate results to make
    the code easier to understand