Intro to Python (Class 5)

Ben Bettisworth

1. Recursion

2. Files

3. Imports

4. General Advice

5. Project

Section 1

Recursion

Recursion

When working with certain structures (particularly trees) it is often easiest to express the logic using *recursion*.

Definition

*A **recursive function** is a function which calls itself.*

Example

Code

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n-1)

print(factorial(5))
```

Output

```
120
```

Recursive Functions

In order to not run forever, every recursive function *must have*:

- A base case, where recursion *does not happen*.
- A recursive case, where recursion does happen.

In order to prevent infinite recursion, you must ensure that the base case is hit eventually.

Example

Code

```
def factorial(n):
  if n == 0: # Base Case
    return 1
  else: # Recursive Case
    return n * factorial(n-1)
```

Graded Exercise

The Fibonacci has a pair of rabbits. Every month, they have 2 baby rabbits. In a month, these baby rabbits will become adults. Every adult pair of rabbits produces a new pair baby rabbits every month.

Example

- At month 1, Fibonacci has 1 pair.
- At month 2, Fibonacci still has 1 pair.
- At month 3, Fibonacci has 2 pairs.
- At month 4, Fibonacci has 3 pairs.
- At month 5, Fibonacci has 5 pairs.

Graded Exercise

The formula for Fibonacci's rabbits is given by

$$F(1) = 1$$
$$F(2) = 1$$
$$F(n) = F(n-1) + F(n-2)$$

Write a function that outputs the pairs of rabbits that Fibonacci has at month n.

Exercise

Code

```
def fibo(n):
  if n == 0 or n == 1: # Base Case
    return 1
  else: # Recursive Case
    return fibo(n-1) + fibo(n-2)
```

Recursion as a Tree

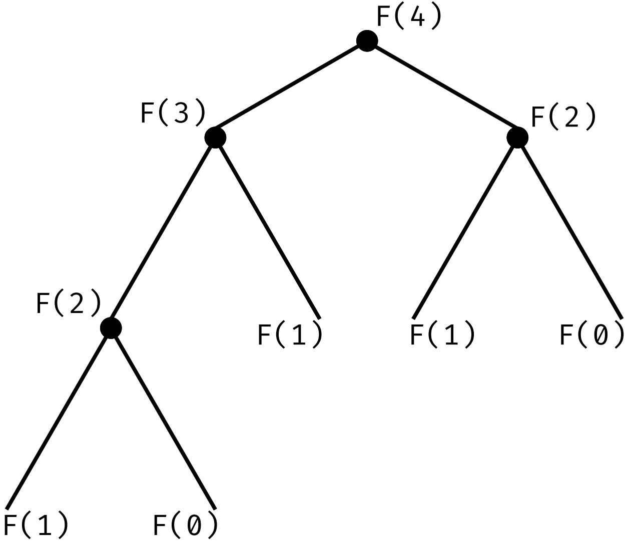

Figure 1: The call tree for `fibo`

Recursion as a Tree

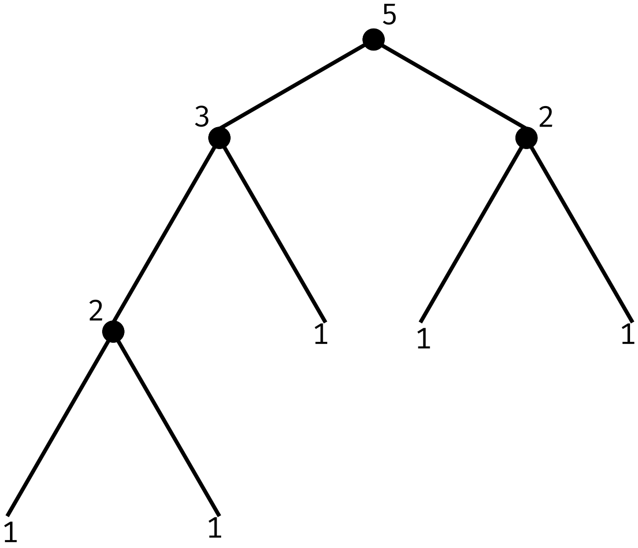

Figure 2: Return tree for `fibo`

Recursion on Phylogenetic Trees

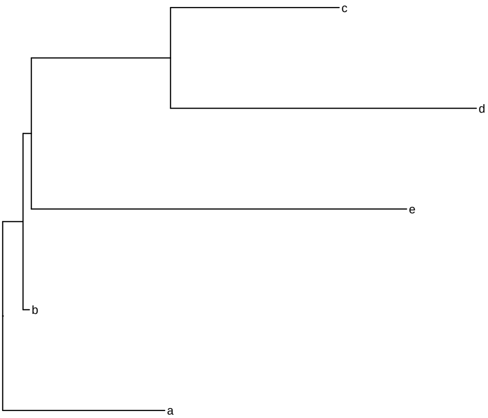

Figure 3: Return tree for `fibo`

Recursion on Phylogenetic Trees

Code

```
def phylo_recurse(PhyloNode):
  for c in PhlyoNode.children:
    phylo_recurse(PhyloNode)
  do_something(PhyloNode)
```

Section 2

Files

Files

At the beginning of the course, we defined a file as:

Definition

Files are chunks of memory stored in a file system

Practical Files

But practically, files are almost always *serialized*. This means that files have a *file format*, which governs their layout.

example.json

```json
{
  "foo": 3.14,
  "bar": "hello world"
}
```

File formats are a compromise between how the computer sees data, and how humans read data. This means that often the file format naturally fits into a `list` or `dict`.

File systems

There are (seemingly) a billion different file systems out there. But, they all identify files with a *path*.

Definition

A path is a ***ordered*** *series of directories and a final filename which identifies a file*

Example

Windows: `C:\Users\Docs\Final.docx`
Unix: `/home/user/Final.docx`

Opening a File

Files need to be *opened* before they can be read. This tells the Operating System (OS) to read the file from disk, and give it to the program.

Code

```
f = open("my_super_cool_file.txt")
print(f)
```

Output

```
<_io.TextIOWrapper name='my_super_cool_file.txt'
  mode='r' encoding='UTF-8'>
```

Reading a File

Normally when reading files in Python, we use a `with` guard.

Code

```
with open("my_super_cool_file.txt") as my_file:
  print(my_file.readline())
```

Output

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor
```

Reading Lines Iteratively

Lines of a file can be read one-by-one until the file is over with a `for` loop.

Code

```
with open("my_super_cool_file.txt") as my_file:
  for line in my_file:
    print(line)
```

Exercise

Read the lines of the file `my_super_cool_file.txt` and place them into a `list`.

Extension: Find the lines which contain a comma (`,`) and reverse those lines.

Exercise

(A Possible) Solution

```
lines = []
with open("my_super_cool_file.txt") as my_file:
  for line in my_file:
    lines.append(line)
```

File Formats

As mentioned before, files generally have a *format*. Examples of file formats include:

- `.docx` (Word Files)
- `.pdf` (Portable Document Format)
- `.fasta` (FASTA)
- `.nwk` (Newick)
- `.bam` (Binary Alignment Map)
- `.zip` (Zip Archive)

Text vs Binary files

There are two ways that a file can be stored on disk:

- Binary, where the files require a special program to read them, and
- Text, which can be read in a text editor (Vscode, emacs, vim, etc.)

In Python, these files are treated differently.

Opening Binary Files

Code

```
with open("my_secret_binary_file.bin", 'rb')\
                                             as my_binary:
  print(my_binary)
```

Output

```
<_io.BufferedReader name='my_secret_binary_file.bin'>
```

Exercise

Parse a `fasta` file into a dictionary, such that the key is the taxa name, and the value is the sequence. Use the file `data/tree1.fa` to test your code.

Extension: Convert the `fasta` file into `phylip`.

Exercise

```python
sequences = {}
with open("data/tree1.fa") as fa_file:
  taxa_line = None
  sequence_line = None
  for line in fa_file:
    line = line.strip()
    if line[0] == ">":
      taxa_line = line[1:]
    else:
      sequence_line = line
    if taxa_line is not None and\
      sequence_line is not None:
      sequences[taxa_line] = sequence_line
      taxa_line, sequence_line = None, None
```

Writing Files

Files are opened in read mode by default. To write a file, you must open the file in write mode. To do this, we pass `"w"` as the second argument of `open`.

Example

```
with open("my_file.txt", "w") as outfile:
  outfile.write("apples have a good flavor")
```

This will create a file with the name `my_file.txt`, and write the text “`apples have a good flavor`” to the file.

Creating Files

Opening a file in in write mode *creates* the file. Even if there is already a file there.

Example

```
with open("my_file.txt", "w") as outfile:
  outfile.write("pears have a sweet flavor\n")
with open("my_file.txt", "w") as outfile:
  outfile.write("but I prefer a banana\n")
```

Result

```
> cat my_file.txt
but I prefer a banana
```

Writing to an Existing File

To write to an existing file, the "append" mode must be used. A file can be opened in append mode by passing "a" as the second argument to `open`.

Example

```
with open("my_file.txt", "a") as outfile:
  outfile.write("only if the banana is ripe though.\n")
```

Result

```
> cat my_file.txt
but I prefer a banana
only if the banana is ripe though
```

Exercise

Write a n by m box to a file named `my_box.txt`.

Section 3

Imports

Imports

Importing in python is the way that programmers include other people's code in their projects. It uses the `import`.

Example

```
import math
```

Imports

Once a *module* is imported, its contents can be accessed with the `.` operator.

Example

```
import math
print(math.exp(3))
```

Exercise

Write a function that computes the probability of getting a flush in a poker hand. As a reminder, a flush is a hand all of the same suit. Use the `math.comb` function.

Exercise

(A Possible) Solution

```
import math

def p_flush():
  denom = math.comb(52,5)

  numer = math.comb(4,1) * math.comb(13, 5)
  return numer / denom
```

Example

Imports can be done to do much of the heavy lifting for you.

Example

```
import csv
with open("cleaned-example.csv") as csv_file:
  csv_reader = csv.DictReader(csv_file)
  for row in csv_reader:
    print(row)
```

Output

```
{'software': 'baseline', 'error': '0.0', ... }
{'software': 'baseline', 'error': '0.011103817', ...}
...
```

Exercise

Write a program that finds the software with largest error in `cleaned-example.csv`.

Extension: Write a program that computes the average error for each different software in the CSV file.

Exercise

(A Possible) Solution

```
import csv
max_err = float("-inf")
max_software = None
with open("test.csv") as csv_file:
  csv_reader = csv.DictReader(csv_file)
  for row in csv_reader:
    err = float(row['error'])
    if err > max_err:
      max_err = err
      max_software = row['software']
```

Third Party Libraries

In addition to the standard libraries, there are *thousands* of third party libraries. These need to be downloaded, but once downloaded[1], then they can be imported just like any other library.

Example

```
import seaborn
import matplotlib.pyplot as plt
tips = seaborn.load_dataset("tips")
seaborn.boxplot(data = tips, x = "day",
                y = "total_bill")
```

[1] It's actually super complicated how you download the libraries. Commonly you might use: `pip`, `conda`, or `uv`.

Exercise

Plot something. Anything. Here is some example code if you need something (taken from the `seaborn` examples).

Example

```
import seaborn as sns
tips = sns.load_dataset("tips")

sns.violinplot(data=tips, x="day", y="total_bill",
               hue="smoker", split=True,
               inner="quart", fill=False,
               palette={"Yes": "g", "No": ".35"})
```

Useful Libraries

- File format parsers: `csv`, `json`, `yaml`.
- Specialized scientific libraries: `numpy`, `scipy`.
- Even more specialized libraries: `ete3`, `BioPython`.
- Plotting libraries: `matplotlib`, `seaborn`, `plotly`.
- Utility libraries: `argparse`.

Section 4

General Advice

General Advice

The remainder of the slides just contain general advice, from somebody who has been doing this a while.

Version Control Systems

You should learn a version control system. This includes any of

- `git`
- `mercurial`
- `fossil`

Version Control Systems

A version control system will:

- Save your work for you.
- Allow you revert changes easily.
- Help you find bugs.
- Make it easy to share code.

Version Control System Example

Example

```
git init .
git add .
git commit -m "initial commit"
```

Section 5

Project

Project

Melissa wants to grow *E. coli* for her experiments. Each i-th day she counts the bacteria in grams. Her boss told her that *E. coli* grows according to the formula

$$n_{i+1} = tn_i$$

where $0 < n_0 \leq 10$. Here t represents the constant temperature of the environment and is bounded by, $0 \leq t \leq 3$.

Melissa runs the experiments with various initial masses (n_0) and temperatures (t). After 5 days of growth, she measures the final mass to 4 decimal places of precision.

Project

Question

In the file `experiments_1.csv` are Melissa's experiments. Each row contains the initial mass (`n0`), the temperature (`t`), and the mass after 5 days (`n5`). Do the results she obtained conform to her boss's proposed formula.

Formula

$$n_{i+1} = tn_i$$

Project (Part 2)

Melissa now understands that pressure also plays an important role and so modified the growth formula to be

$$n_{i+1} = tn_i - pn_i^2$$

Where t is again the temperature, and p is pressure. Additionally, Melissa can ensure that $0 \leq (t, p) \leq 3$. Melissa reruns the experiments, measuring the mass of samples after 5 days.

Project (Part 2)

Question

In the file `experiments_2.csv` are Melissa's experiments. Each row contains the initial mass (`n0`), the temperature (`t`), the pressure (`p`), and the mass at day 5 (`n5`). At the end of 5 days of growth, do the results follow the formula that she proposed?

Formula

$$n_{i+1} = tn_i - pn_i^2$$

Project (Part 3)

Also in the file `experiments_2.csv`, there are the result masses after an infinite amount of time[2] as `n-inf`. However, Melissa is not sure about this data. Can you verify that it is correct, according to her formula?

[2] It was measured by a time traveler.