Class 6 - Complexity, Modules and Packages

[w200] MIDS Python Course 2018

Course Content | First 8 Weeks - Programming

Unit 1 | Introduction, the Command Line, Source Control

Unit 2 | Starting Out with Python

Unit 3 | Sequence Types and Dictionaries

Unit 4 | More About Control and Algorithms

Unit 5 | Functions

Unit 6 | Complexity, Modules and Packages

Unit 7 | Classes

Unit 8 | Object-Oriented Programming

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

Assignment Review | Week 5

Refresher:

1. Functions (nested)
2. Namespace
3. Functions as objects `int` `int()`
4. Exceptions

Week 6 | Polls

Discuss: What was the hardest part of HW5?

Poll: How long did you spend on this week's assignment?

Homework 4 Grading

- Overall: Much harder homework than previous ones!
  - Please look at the posted solutions
  - If you still have questions, email the google group or the three of us!
- A note on 'for' loops:
  - Embedding two 'For' loops like:

```
for x in range(1,100):
    for y in range(1,10):
```

  - Should be done only when no other solution can be used (this quickly becomes untenable with large # of iterations!)
  - Embedding three 'for' loops = find a better solution!

Jupyter Notebook Variable Space

- Variables in Jupyter:
  - For example: `x = 4; print(x)` in Jupyter
  - If you delete the `x = 4;` you can still `print(x)`
  - x is stored in the notebook memory even though it isn't defined anymore
- This is a problem:
  - When we re-run your code "x" is not in our notebook's memory
  - Code crashes with: "x is undefined" error
  - Please go to the Kernel menu - restart and clear output
  - Then re-run all of your code blocks before turning it in!

Variable Names in Python

- **Be careful about naming variables in Python**
  - In Jupyter notebook the python reserved words display as green text (for example: list)
- **Python will let you overwrite these functions!**
  - You can say list = [1, 2, 3, 4] even though list is a function in python (to make something a list)
  - As soon as you overwrite that function, list is now your variable and can't be used as the function to make something a list
  - If this happens by accident, do the Kernel - restart and clear output to reset it (and of course rename your variable something like my_list = [1, 2, 3, 4]

Scrabble Assignment (Homework 6)

- Notes for the Scrabble assignment:
  - Put comments in your code and docstrings in the functions
  - Error check and give good error messages to the user
  - Runtime - should run in a reasonable amount of time (single digit minutes or seconds)
  - You can use a pre-processed dictionary if this makes sense to you

Week 6 | Your Focus Next Week

Project 1 (20% of final grade)

- Email proposed topic
- Start design document (due in 2 weeks)

Async / Homework

Review of Classes

Review of Course Concepts

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

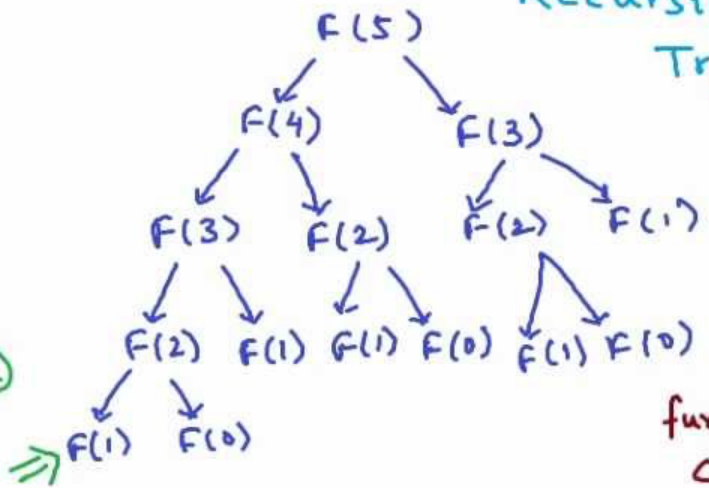
On Deck - Next Week

Fibonacci Sequence – Space Complexity analysis

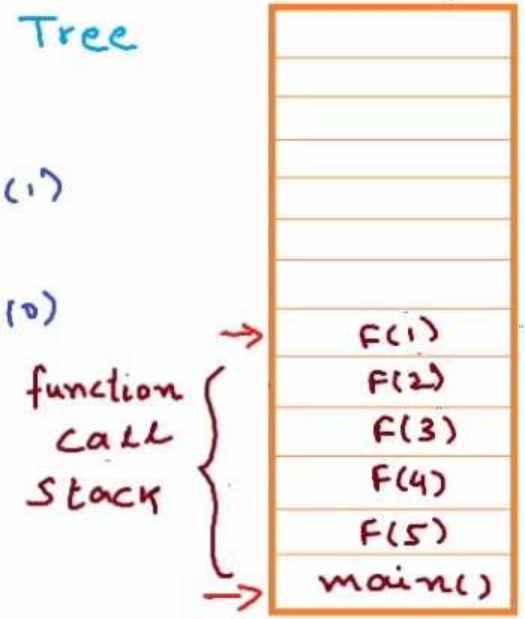
0 1 1 2 3 5 8 . . .

```
Fib(n)
{
   if n<=1
     return n
   else
     return Fib(n-1)+ Fib(n-2)
}
```

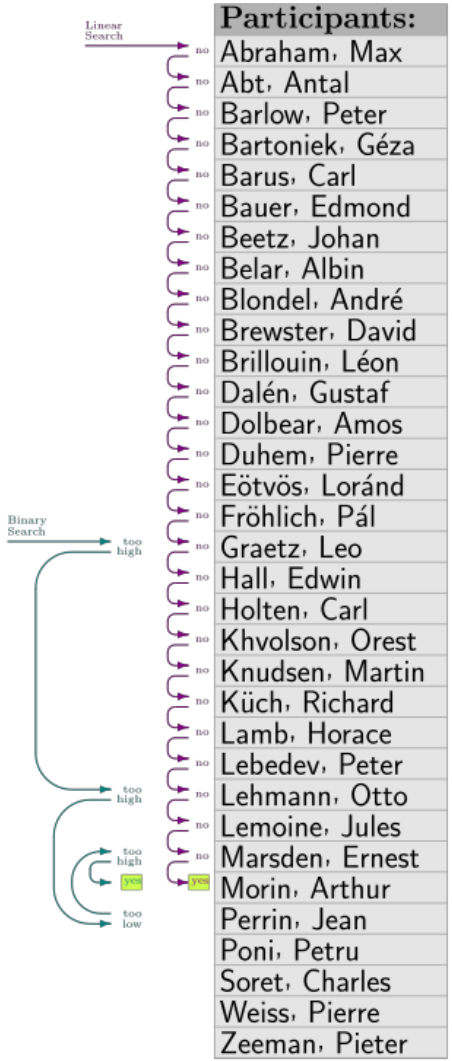
Recursion Tree



Memory



For looking up a given entry in a given ordered list, both the binary and the linear search algorithm (which ignores ordering) can be used. The analysis of the former and the latter algorithm shows that it takes at most $\log_2(n)$ and n check steps, respectively, for a list of length n. In the depicted example list of length 33, searching for *"Morin, Arthur"* takes 5 and 28 steps with binary (shown in cyan) and linear (magenta) search, respectively.



Complexity | What the...?

Question: What do we mean when we mention the "complexity" of code?

Complexity | What the...?

Question: What do we mean when we mention the "complexity" of code?

- Think of complexity as the number of steps and how those steps SCALE with the size of the data set

Complexity | What the...?

Question: Why is it important to think about code complexity?

Complexity | What the...?

Question: Why is it important to think about code complexity?

- Code complexity will have important ramifications when you run your code on different size data sets. Code that does not scale well will have limited usefulness when used at scale and may even make an approach practically unusable.

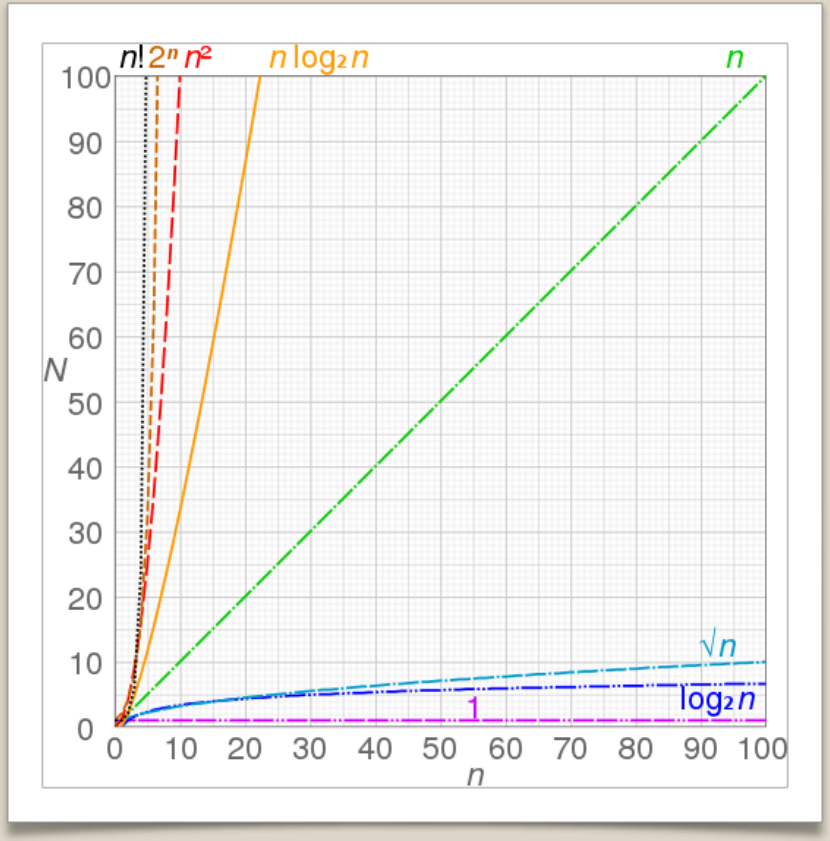
Complexity | Big O Notation

There are multiple standard complexity "measures"

- `O(1)`
- `O(N)`
- `O(log(N))`
- `O(Nlog(N))`
- `O(log(N^2))`

Which of these take the most time to run?

Which take the least?



Hard to figure out … don't stress over $O(n)$ …

- We care most about n^2 and $2n^2$

Check out wolframalpha.com for charting the difference in complexity models.
E.g., xlog(x) from 1 to 100

http://www.wolframalpha.com/input/?i=plot+xlog(x)+from+1+to+100

http://www.wolframalpha.com/input/?i=plot+log(x)+from+1+to+100

Complexity | Big O Notation

How do we classify algorithms by time

- Time it
- We care about orders of magnitude differences
- Abstracting away from hardware
- Abstract away from which elemental steps are used

Think in terms of how many steps

Complexity I Big O Notation

Rules to simplify - understand the growth rate

$$f(n) = 2n^2 + n + 100$$

- Think in terms of worst case scenario
- Remove lower order terms (n and 100)
  - Those that do not scale or scale at lower rates
- Remove constants (2)
  - Not reliable when considering hardware and elementary steps
- Can always refine the algorithm to reduce the constant

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

Packages and modules | Organize functions

Module

- A python (.py) file with related functions

Package

- A folder of related modules

Import

- packages and modules with “import”

Packages and modules | The import statement

Import

Import packages and modules with “import”

The method of import changes how you name the imported function when using it

- `import sys`
  - Call with `sys.argv`
- `from sys import argv as argv`
  - Call with `argv`
- `import sys as s`
  - Call with `s.argv`
- `import sys.argv as s`
  - Call with `s`

`import numpy as np`

can you see a potential problem?

Packages and modules | example

Packages: Example

```
import numpy.matlib
```

- In this example "numpy" is the package and "matlib" is the module within the package numpy.
- The package "numpy" has a empty file named "__init__.py" that tells Python that numpy is a package.

- We can also import the whole package
  - **`import numpy`**

Packages and modules | Search Path

- First in current directory
- Then down the sys.path list
- You can modify the search path with sys.path.append()

Note on Module Search Path

```
import sys
for place in sys.path:
    print(place)
```

```
/Users/Personal/Documents/venv3/lib/python35.zip
/Users/Personal/Documents/venv3/lib/python3.5
/Users/Personal/Documents/venv3/lib/python3.5/plat-darwin
/Users/Personal/Documents/venv3/lib/python3.5/lib-dynload
/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5
/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/plat-darwin
/Users/Personal/Documents/venv3/lib/python3.5/site-packages
/Users/Personal/Documents/venv3/lib/python3.5/site-packages/IPython/extensions
/Users/Personal/.ipython
```

```
import sys
for place in sys.path:
    print(place)
```

Packages and modules | Troubleshooting

```
In [12]: !python3 helloworld.py
         Hello World, revisited
```

- Run the help() command to see functions in a package
- ! to invoke the command line in ipynb
- You can run or “cat” codes to print them directly
- You can specify python3 on CLI

Packages | Quick Demo & Activity

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

Sys.argv | Arguments into .py file

- Using `sys.argv`:
  - item 1 is script name
  - other items are passed items

```
# showargmuments.py
#
# Prints out the program arguments
# that have been passed into this
# standalone application

import sys # We need to import the sys modu

print("Program arguments:", sys.argv)
```

```
In [15]: !python3 showarguments.py
         Program arguments: ['showarguments.py']

In [16]: !python3 showarguments.py firstArg secondArg thirdArg
         Program arguments: ['showarguments.py', 'firstArg', 'secondArg', 'thirdArg']
```

Date times

- Time zone
- Calendars
- Time deltas
- UTC offset
- Time zone
- Date time object (time stamp)
- Print in the desired format (strftime)
  - Formatting code reference ::: scroll to the bottom
  - https://docs.python.org/2/library/datetime.html

```
In [6]: # Adding this line to ensure that this code works outside its context:
import datetime
weatherData = {}
weatherData["dt"] = 1440270976

# First need to define the UTC offset for Berkeley, CA (UTC - 8:00)
current_utc_offset = -datetime.timedelta(hours=8)

# Next we create a timezone based on the utc offset for Pacific Standard Time (not daylight savings time)
current_timezone = datetime.timezone(current_utc_offset)

# Last we create a datetime object based on the timestamp provided by the response, and
# we localize the timezone to make it "aware"
current_datetime = datetime.datetime.fromtimestamp(weatherData["dt"], current_timezone)

# Printing of the forecast

# Note we use strftime to format how we would like to print out the datetime
forecastStr = "Forecast for Berkeley, CA on " + current_datetime.strftime("%A, %B %d, %Y %H:%M %p") + " local time\n" \

# printing result for demonstration purposes
print(forecastStr)

Forecast for Berkeley, CA on Saturday, August 22, 2015 11:16 AM local time
```

see the “forecast.py” file

Date times

https://docs.python.org/3/library/datetime.html

see the “forecast2.py” file

- Unix timestamp highlighted
- Daylight savings causes issues at the changeover - Use ptyz package
- Print in the desired format (strftime)
  - https://docs.python.org/2/library/datetime.html (bottom)

In [6]:
```
# Adding this line to ensure that this code works outside its context:
import datetime
weatherData = {}
weatherData["dt"] = 1440270976

# First need to define the UTC offset for Berkeley, CA (UTC - 8:00)
current_utc_offset = -datetime.timedelta(hours=8)

# Next we create a timezone based on the utc offset for Pacific Standard Time (not daylight savings time)
current_timezone = datetime.timezone(current_utc_offset)

# Last we create a datetime object based on the timestamp provided by the response, and
# we localize the timezone to make it "aware"
current_datetime = datetime.datetime.fromtimestamp(weatherData["dt"], current_timezone)

# Printing of the forecast

# Note we use strftime to format how we would like to print out the datetime
forecastStr = "Forecast for Berkeley, CA on " + current_datetime.strftime("%A, %B %d, %Y %H:%M %p") + " local time\n" \

# printing result for demonstration purposes
print(forecastStr)
```

```
Forecast for Berkeley, CA on Saturday, August 22, 2015 11:16 AM local time
```

see also: counter1.py

Standard library | collections.counter()

Quickly create a counting dictionary by simply passing in a list

1) Makes list
2) Passes into counter()

Operations on counters

- MyCounterObj.most_common()
- Add, subtract, | (or), &

```
stanford_list = list()
for item in book_data["items"]:
    stanford_list.append(item["volumeInfo"].setdefault("publisher", "None"))

stanford_counter = Counter(stanford_list)
print(stanford_counter)
```

```
Counter({'None': 10, 'Stanford University Press': 5, 'Springer Science & Business Media': 4, 'Genealogical Publishing Com': 2,
 'AuthorHouse': 2, 'CRC Press': 2, 'A&C Black': 2, 'Cambridge University Press': 2, 'Wentworth Press': 1, 'Routledge': 1, 'Xlib
ris Corporation': 1, 'Llumina Press': 1, 'Pickpocket Publishing': 1, 'Strategic Book Publishing': 1, 'Cuvillier Verlag': 1, 'Pa
lala Press': 1, 'Lexington Books': 1, 'Book Guild Publishing': 1, 'FrancoAngeli': 1})
```

Standard library | Default values

Avoid key error with defaults. Two flavors:

- Regular Dictionary: `dictionary_name.setdefault(KEY, VALUE)`
- DefaultDict: `collections.defaultdict(defaultvalue)` # default as a function, int, empty list ...

1) Initialize defaultdict to produce an int if there is no value
2) For each volume, use setdefault to get the publisher or "none" as the key
3) Increment the publisher count as the value

```
# Here we create a dictionary whose default is int(), or zero
publisher_counter = defaultdict(int)

# We will go through all of the books that are related to berkeley and count
# how many books were published from a particular publisher
for item in book_data["items"]:

    # Note how we use the function setdefault to set publisher to None if there's no
    # publisher in the response
    publisher = item["volumeInfo"].setdefault("publisher", "None")

    # The default nature of publisher_counter enables us to do this without any raised
    # exceptions
    publisher_counter[publisher] += 1
```

Standard library | Kitchen sink

https://docs.python.org/3/library/collections.html

```
collections.deque()
collections.OrderedDict()   # for ordered streams etc.
    pop()
    popleft()
pprint()                #pretty print:
pip install()
Virtualenv()            #e.g.,: conda env create
```

Web scraping | Requests

Google books

Open a URL
Read and decode
Load it as json

```
from collections import defaultdict
from urllib.request import urlopen
import json

response = urlopen('https://www.googleapis.com/books/v1/volumes?q=berkeley&maxResults=40')
rawData = response.read().decoding("utf-8")
book_data = json.loads(rawData)
```

Weather data

```
from urllib.request import urlopen
import json

def get_report():
    """
    Returns the current forecast of Berkeley right now
    """
    response = urlopen(
        'http://api.openweathermap.org/data/2.5/weather?q=Berkeley,ca&appid=7dc34849d7e8b6fbd
cb3f12454c92e88')
    rawWeatherData = response.read().decode("utf-8")
    weatherData = json.loads(rawWeatherData)

    forecast = "Berkeley, CA Forecast: " + weatherData["weather"][0]["main"]
    return forecast
```

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

The Project | Your Mission

Create a small, object-oriented program of your choosing:

Examples:

- An ATM
- A flower shop
- An adventure game
- Something relating to your everyday work

The Project | Code

1. Python 3 code, 300-500 lines (750 max)
2. All code should be well commented!
3. Must use Object Oriented design and classes
4. Demonstrate various flow controls and data types
5. Robust to common user errors and exceptions

The Project | Your Mission

The user will interact with your program via Terminal/Shell

Three documents due before your class on 3/13 or 3/15:

1. Proposal (10%)
2. Code(s) (80%)
3. Reflective Summary (10%)

You will demo your progress in a breakout room the week before the project is due

You may only use Python libraries that come installed with Anaconda

The Project | Proposal

Describe your project concept

Pseudocode your major classes and functions

1. Briefly describe the purpose of each class
2. List expected functions belong to each class
3. List inputs and outputs for each function

Instructors will “approve” your draft proposal

Coding is *iterative*. Your final code may not match the proposal exactly.

The Project | Reflection

Submit a 1-page reflection with your code

Instructors will read your reflection before grading your project

Tell us how to use your project!

Discuss challenges you faced and how you overcame them

The Project | Demo

As time allows, show 1-2 examples of strong projects from last semester.

The Project | Questions

Week 6 | Agenda

Homework Review and Admin

Jupyter Notebooks are Just a Dictionary!

Complexity

Packages and Modules - Activity

The Kitchen Sink

Project 1 Discussion

On Deck - Next Week

Up next

Scrabble Assignment

Project 1 (build your own object oriented project)

- Plan due in two weeks

Unit 7 (Classes)

Unit 8 (Object-Oriented Programming)

Week 6 | Your Focus Next Week

Project 1 (20% of final grade)

- Email proposed topic
- Start design document (due in 2 weeks)

Async / Homework

Review of Classes

Review of Course Concepts

Extra slides - calculate bigO run time

Q 1

Compute the (tightest) Big O running time bound of the following scripts:

```
for i in range(N):
    for j in range(N^2):
        print(i + j)
```

For each turn how operations do we do?

Does the input change each turn?

see also bigO.py

Extra slides - calculate bigO run time

```
Q 2
i = N
j = 1
while j<i:
    i *= 100
    j *= 101
```

For each turn how many function calls are there ?

How does the input change each turn?

What is the stopping condition?

When will you reach the stopping condition

Extra slides - calculate bigO run time

```
Q 3
i = 0
j = 1
while i<N:
    for k in range(j):
        i += 1
    j *= 2
```

For each turn how many loops are there ?

How do the number of operations change as N is increased?

What is the stopping condition?

Extra slides - calculate bigO run time

Q 4

```
def func(i):
    if i<1:
        return 1
    else:
        return func(i-1) + func(i-1)
print(func(N))
```

For each turn how many function calls are there ?

How does the input change each turn?

What is the stopping condition?