Continued…

Help

```
help(calculator)

______________________________________________________________________

Help on module calculator:


NAME
    calculator


DESCRIPTION
    Calculator Documentation
    This class holds the calculator functions and memory variables.


-- More  --
```

Lists & Dictionaries

Lists - Most general sequence

- Positionally ordered collections of arbitrarily typed objects.
- Mutable
  - Can be modified in place.
- Provide a very flexible tool for representing arbitrary collections.

Lists and lists of lists

Initialize with []s and “,” separator.

```
book1 = ["Learning Python", "Mark Lutz", 2013]
book2 = ["As I Lay Dying", "William Faulkner", 1930]
books = [book1, book2]
```

Can access parts by using []s.

REMEMBER – INDEXES START AT 0!!!

```
print(book[0])
print(books[1][2])
```

```
['Learning Python', 'Mark Lutz', 2013]
1930
```

Can do lots of fun stuff.

Get columns

```
authors = [row[1] for row in books]
print(authors)
```

```
['Mark Lutz', 'William Faulkner']
```

YOU'RE MY SON AND NO MATTER WHAT, YOUR MOTHER AND I WILL ALWAYS LOVE YOU.
ARRAYS START AT 1
FREE DISAPPOINTMENT

Dictionaries

Mappings

```
book1 = {"title": "Learning Python", "author": "Mark Lutz", "PubDate":
2013 }
book2 = {"title": "As I Lay Dying", "author": "William Faulkner",
"PubDate": 1930 }
books = [book1, book2]

for book in books:
      print(book["title"])
```

```
Learning Python
As I Lay Dying
```

Dictionaries

```
tempTitle = input("Enter Title: ")
tempAuthor = input("Enter author: ")
tempPubDate = int(input("Enter Publication Year: "))

book3 = dict(title = tempTitle, author = tempAuthor, PubDate = tempPubDate )
```

Dictionaries

But what will happen if you don’t have an attribute?

```
book1 = {"title": "Learning Python", "author": "Mark Lutz", "PubDate":
2013 }
book2 = "author": "William Faulkner", "PubDate": 1930 }
books = [book1, book2]

for book in books:
      print(book["title"])
```

```
Learning PythonTraceback (most recent call last):
  File “Example.py", line 7, in <module>
    print(book["title"])
KeyError: 'title'
```

Enum

```
from enum import Enum
class Color(Enum):
	RED = 0
	GREEN = 1
	BLUE = 2
```

File I/O

File I/O

Write to a file

```
myfile = open('myfile.txt', 'w')
myfile.write('Hello text file\n')
myfile.write('Goodbye text file\n')
myfile.close()
```

File I/O

Read a file

```
myfile = open('myfile.txt')
print(myfile.readline())
print(myfile.readline())
print(myfile.readline())
```

File I/O

Storing Python Objects in JSON Format

```
import json

json.dump(books, fp=open('myFile.txt','w'), indent=4)
```

fp=open(…) is required, don't rename it.

Also, JSON doesn't "play nicely" with Enums. There is a way to do it, but it's not standard; look it up if you're interested.

File I/O

Reading Python Objects from JSON Format

```
import json

newBook = json.load(open('myFile.txt'))
```

See Books.py for an example