What does this code do?

There’s probably some unfamiliar syntax; take your best guess!

```python
lines = open('data.txt').readlines()
data = [line[:-1] for line in lines]
values = [map(int, row.split(',')) for row in data]
results = map(sum, values)
```

(Your response)
Blocks in Python
(conditionals, loops, functions)
Getting input from user

```
value = int(input('Enter a number: '))
```

- **prompt**
- **returns a string**
- **converts to an int**
- **binds result to a variable**
if / else statements

indentation is required!

```python
value = int(input('Enter a number: '))
if (value % 2 == 0):
    print('even')
else:
    print('odd')
```
Use spaces (not tabs) to indent.

That way, your code is more portable.
Most editors allow you to use the tab key to insert spaces.
for loops

```python
value = int(input('Enter a number: '))
for i in range(value):
    print(i)
```
while True:
    value = int(input('Enter an even number: '))
    if (value % 2 == 0):
        break
def getEvenNumber():
    '''Gets an even number from the user.''

    while True:
        value = int(input('Enter an even number: '))
        if (value % 2 == 0):
            break

    return value
Good programming practice

Write docstrings for all your functions.
That way, your code is more readable.
Reading from files

```python
open('data.txt').readlines()
['0,1,2,3,4,5,6,7,8,9
 0,2,4,6,8
100,200,300,400,500,600,700,800,900
10,17,24,31,38,45,52,59,66,73,80,87,94
',
'0,1,2,3,4,5,6,7,8,9
0,2,4,6,8
100,200,300,400,500,600,700,800,900
10,17,24,31,38,45,52,59,66,73,80,87,94
'
]
```
Slicing a sequence

```
seq[start : end : step]
```

think: \([\text{start, end})\]

```python
>>> values = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> values[3:]
[3, 4, 5, 6, 7, 8, 9]
>>> values[3:5]
[3, 4]
>>> values[-1]
9
>>> values[::2]
[0, 2, 4, 6, 8]
```
Functional programming in Python
List comprehensions
are syntactic sugar for functional programming concepts (e.g., map)

```python
lines = open('data.txt').readlines()

data = []
for line in lines:
    data.append(line[:-1])

```

```python
data = [line[:-1] for line in lines]
```

```python
data = list(map(lambda line: line[:-1], lines))
```

loop
( imperative programming )
map
( functional programming )
list comprehension
( functional programming )
List comprehensions are syntactic sugar for functional programming concepts (e.g., `filter`).

```python
positiveValues = []
for value in values:
    if value > 0:
        positiveValues.append(value)

data = list(filter(lambda value: value > 0, values))

data = [value for value in values if value > 0]
```
Good programming practice

Use list comprehensions.

List comprehensions are usually clearer (to Python programmers) than map / filter or single loops that build up lists.
Python sounds good!

http://tinyurl.com/hmc-python-sounds

Help with the terminal: http://tinyurl.com/hmc-python-terminal

Try to get as far as: static

We’ll stop at 10:45.