Talking to Python

```
csev$ python3
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwinType
"help", "copyright", "credits" or "license" for more information.
>>>
```

What next?

```
csev$ python3
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwinType
"help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print (x)
1
>>> x = x + 1
>>> print (x)
2
>>> exit()
```

This is a good test to make sure that you have Python correctly installed. Note that quit() also works to end the interactive session.

What Do We Say?

Elements of Python

- Vocabulary / Words - Variables and Reserved words (Chapter 2)
- Sentence structure - valid syntax patterns (Chapters 3-5)
- Story structure - constructing a program for a purpose

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

A short “story” about how to count words in a file in Python

```
python words.py
Enter file: words.txt
to 16
```

Reserved Words

- You cannot use reserved words as variable names / identifiers

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

Sentences or Lines

```
x = 2
```
Assignment statement

```
x = x + 2
```
Assignment with expression

```
print(x)
```
Print function

Variable (e.g., x)

Operator (e.g., = +)

Constant (e.g., 2)

Function (e.g., print ())

Programming Paragraphs

Python Scripts

- Interactive Python is good for experiments and programs of 3-4 lines long.
- Most programs are much longer, so we type them into a file and tell Python to run the commands in the file.
- In a sense, we are “giving Python a script”.
- As a convention, we add “.py” as the suffix on the end of these files to indicate they contain Python.

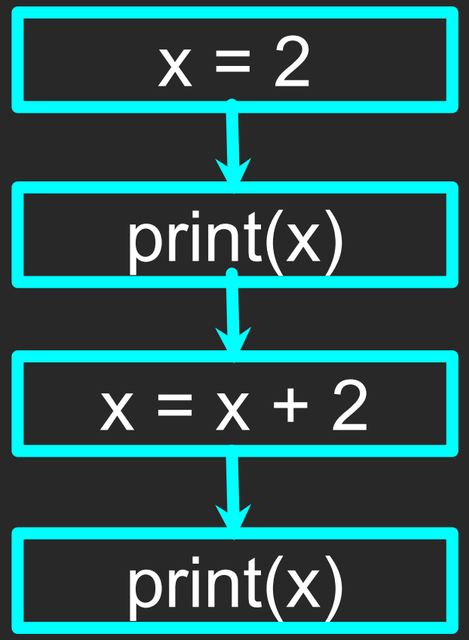
Interactive versus Script

- Interactive
  - You type directly to Python one line at a time and it responds
- Script
  - You enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the statements in the file

Program Steps or Program Flow

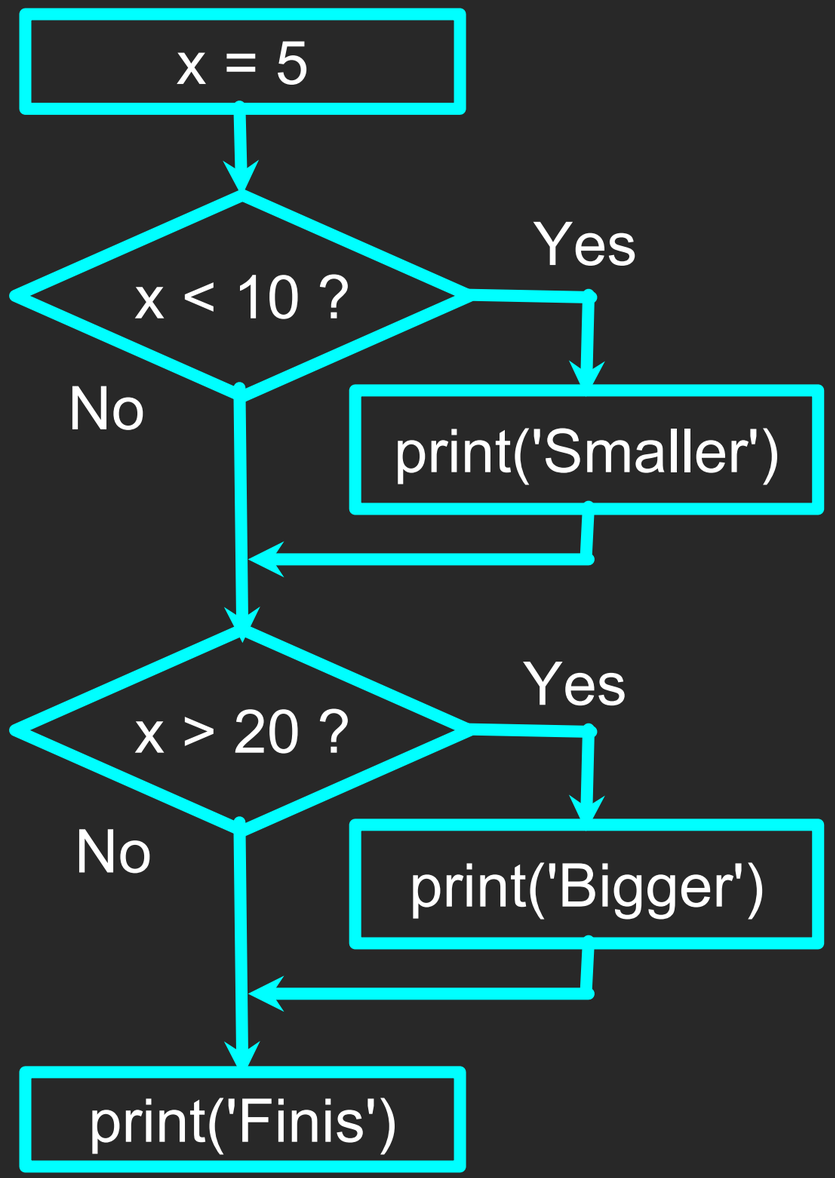
- Like a recipe or installation instructions, a program is a sequence of steps to be done in order.
- Some steps are conditional - they may be skipped.
- Sometimes a step or group of steps is to be repeated.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program (Chapter 4).

Sequential Steps



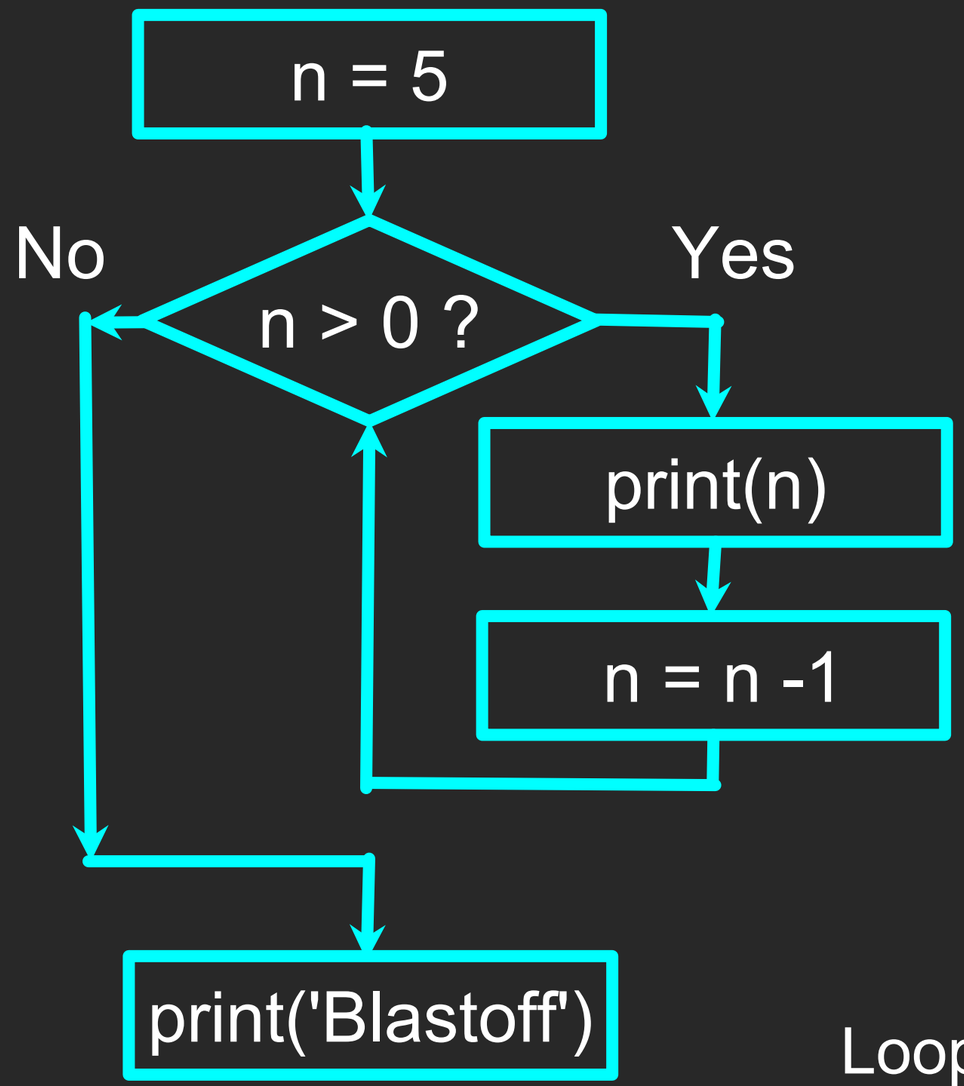
Program	Output
`x = 2`	
`print(x)`	2
`x = x + 2`	
`print(x)`	4

When a program is running, it flows from one step to the next.
As programmers, we set up “paths” for the program to follow.



Conditional Steps

Program	Output
`x = 5`	
`if x < 10:` `  print('Smaller')`	Smaller
`if x > 20:` `  print('Bigger')`	
`print('Finis')`	Finis



Repeated Steps

Program	Output
`n = 5`	

`while n > 0:`
`  print(n)`
`  n = n - 1`
	5
	4
	3
	2
	1

`print('Blastoff!')`	Blastoff!

Loops (repeated steps) have iteration variables that change each time through a loop.

Sequential

```
name = input('Enter file:')
handle = open(name)

counts = dict()
```

Repeated

```
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1
```

Sequential

```
bigcount = None
bigword = None
```

Repeated

```
for word,count in counts.items():
```

Conditional

```
if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count
```

Sequential

```
print(bigword, bigcount)
```

```
name = input('Enter file:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

A short Python "Story" about how to count words in a file

A word used to read data from a user

A sentence about updating one of the many counts

A paragraph about how to find the largest item in a list

Summary

- This is a quick overview of Chapter 1
- We will revisit these concepts throughout the course
- Focus on the big picture

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors and Translators here

Continue…