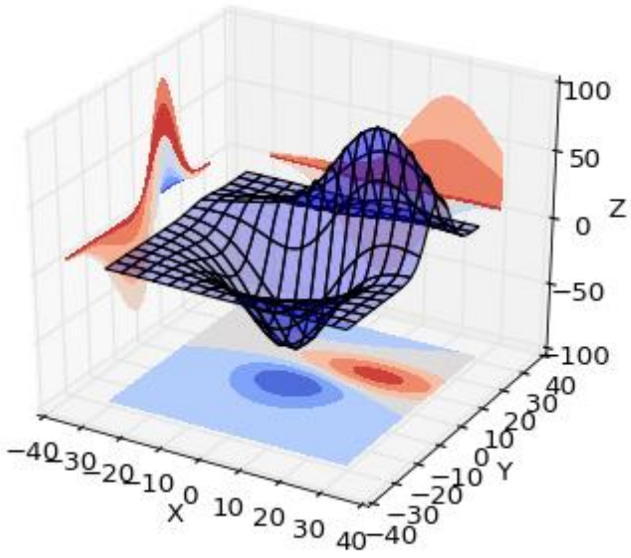
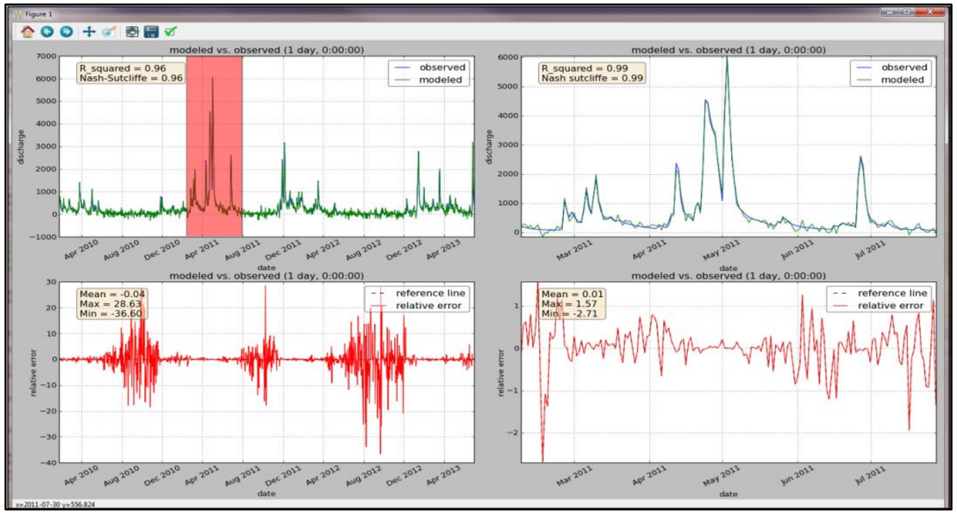
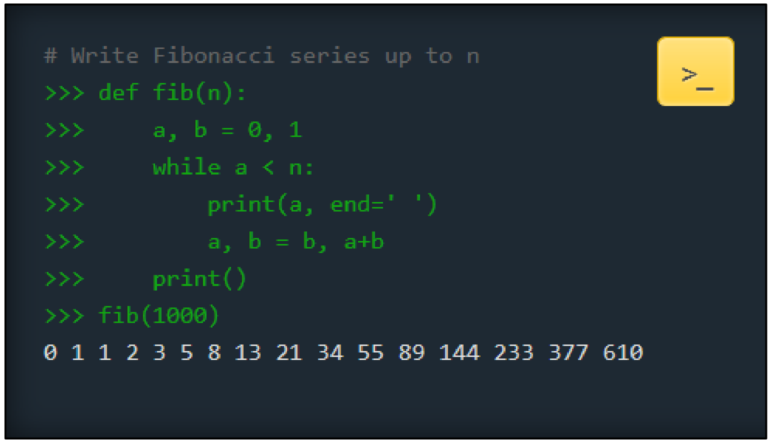
Introduction to Scientific Computing
Meeting 16
Programming with Python







```
# Write Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

Jeremiah Lant, Hydrologist
USGS Kentucky Water Science Center
jlant@usgs.gov

Last Meeting

- Learned multiple ways to print; **print**

```
>>> print “hello”     # Python 2.* print is a statement
>>> print(“hello”)   # Python 3.* print is a function
```

- Ways to print out a single variable or value with string

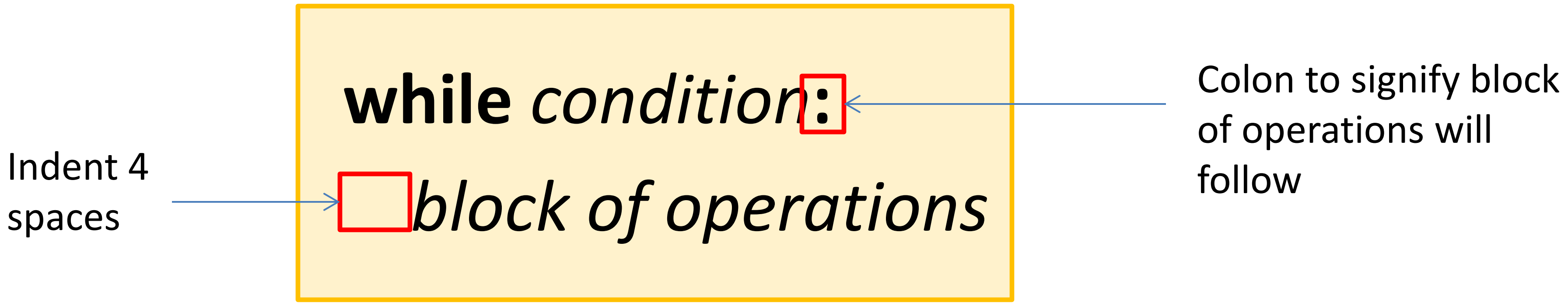
```
>>> x = 10
>>> print(“The value is”), x
>>> print(“The value is %s”) % x
>>> print(“The value is {}”.format(x))
```

- Ways to print out multiple variables with string

```
>>> x = 10
>>> y = 20
>>> print(“The value of x is   and y is   ”), x, y
>>> print(“The value of x is %s and y is %s”) % (x, y)
>>> print(“The value of x is {} and y is {}”.format(x, y))
>>> print(“The value of y is {1} and x is {0}”.format(x, y))
```

Last Meeting – While Loop

- While loops are used to repeat an operation or set of operations until a certain condition is met.



while *condition***:**
    block of operations

```
>>> number = 0
>>> while number <= 5:
...     print(number)
...     number += 1
```

Last Meeting – if, elif, else

- if, elif, else are used to make decisions (choices)

Indent 4 spaces

```
if condition:
    block of operations
elif condition:
    block of operations
else condition:
    block of operations
```

Colon to signify block of operations will follow

```
>>> number = 10
>>> if number == 0:
. . .      print(“equals zero”)
. . .  elif number < 0:
. . .      print(“negative”)
. . .  else number > 0:
. . .      print(“positive”)
positive
```

Today’s Objectives

- Learn about main built-in container called a **list**
- Learn how to loop through lists using a **for loop**

Demo – List

- **List** is a **container** or a **collection of items**.

list = [*item0, item1, item2, …***]**

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> names = [“Jeremiah”, “Justin”, “Dave”, “Loren”]
```

Demo – List

- **List** is a **container** or a **collection of items.**

list = [*item0, item1, item2, …***]**

Enclosing brackets

Use commas to separate items

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> names = [“Jeremiah”, “Justin”, “Dave”, “Loren”]
```

Demo – List

- Items in a **list** are in an **ordered sequence**.

  list = [*item0, item1, item2, item3***]**

```
>>> names = [“Jeremiah”, “Justin”, “Dave”, “Loren”]
```

- Items in a **list** can be accessed using an **index** (where item is located in the sequence)

```
>>> names[0]
“Jeremiah”
>>> names[2]
“Dave”
```

- **Lists** support **slicing**

```
>>> names[1:3]                  # slice from 1 up to BUT NOT INCLUDING 3
[“Justin”, “Dave”]
>>>  names[2:]                  # slice from 2 to end of list
[“Dave”, “Loren”]
>>> names[3]
“Loren”
>>> names[-1]                   # access last element of list
“Loren”
>>> names[:2]                   # slice from beginning up to BUT NOT INCLUDING 2
[“Jeremiah”, “Justin”]
```

Demo – List

- **Lists** are **mutable** – items can be changed after list is created.

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> numbers[0] = 10
>>> print(numbers)
[10, 1, 2, 3, 4, 5]
```

- **In contrast, a tuple is immutable** – items can NOT be changed after tuple is created

```
>>> numbers = (0, 1, 2, 3, 4, 5)
>>> numbers[0] = 10
TypeError: ‘tuple’ object does not support item assignment
```

Demo – List

- **Lists** can contain **arbitrary types** – items can be of different types.

```
                    strings     integers  floats     lists
>>> some_list = [“hello”, 100, 2.5, [3, 4, 5]]
>>> some_list[-1]
[3, 4, 5]
```

- **Lists** have **length** – use function **len()**

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> len(numbers)
6
```

Demo – List

- **Lists** have **methods** that operate on a list
  - https://docs.python.org/2/tutorial/datastructures.html

- **Append method** – add item to end of list

```
>>> some_list = [0, 1, 2]
>>> some_list.append(3)   # add 3 to end of list
>>> print(some_list)
[0, 1, 2, 3]
```

- **Sort method** – sort items of the list in place

```
>>> numbers = [5, 4, 3, 2, 1, 0]
>>> numbers.sort()        # sort is ascending order
>>> print(numbers)
[0, 1, 2, 3, 4, 5]
```

Demo – List

- **Lists** with **+, * operators**

```
>>> numbers = [0, 1, 2]
>>> numbers + 3
TypeError: can only concatenate list (not “int”) to list


>>> numbers + [3]        # looks like append, BUT numbers list is not changed
[0, 1, 2, 3]
>>> print(numbers)
[0, 1, 2]
>>> numbers * 2          # doubles list, BUT numbers list is not changed
[0, 1, 2, 3, 0, 1, 2, 3 ]
>>> print(numbers)
[0, 1, 2]
```

Demo – list

- **range()** – built-in function that constructs a list of numbers; increments by 1 by default

range(*start, stop but not including, increment***)**

```
>>> range(3)
[0, 1, 2]
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 2)
[0, 2, 4, 6, 8]
```

Demo – List

- Can **test** for **item membership** in **list**

```
>>> names = [“Jeremiah”, “Justin”, “Dave”, “Loren”]
>>> group_member = “Moon”
>>> group_member in names
False
```

- Example program

```
names = [“Jeremiah”, “Justin”, “Dave”, “Loren”]
group_member = “Moon”

if group_member in names:
    print(“{} is already in list of names”.format(group_member))
else:
    names.append(group_member)
    print(“Added {} to list of names”.format(group_member))

print(names)
```

Demo – For Loop

- **For loops** are used to **repeat an operation** or set of operations **a certain number of times**.

for *target* **in** sequence:

block of operations

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> for num in numbers:
. . .     print(num)
```

Last Meeting – For Loop

- For loops are used to repeat an operation or set of operations a certain number of times.

for *target* **in** sequence:

    block of operations

Indent 4 spaces

Colon to signify block of operations will follow

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> for num in numbers:
. . .     print(num)
```

Demo – For Loop

- **For loop using indexing to access items/elements in a list**

for *i* **in** range(len(sequence))**:**
    block of operations

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> for i in range(len(numbers)):
. . .     print(numbers[i])
```

Demo – For Loop vs While Loop

- **For loop**

```
>>> numbers = [0, 1, 2, 3, 4, 5]
>>> for num in numbers:
. . .     print(num)
```

- **While loop**

```
>>> number = 0
>>> while number <= 5:
. . .     print(number)
. . .     number += 1
```

Video – Python Basics

software carpentry

- Software Carpentry, Greg Wilson
  - Python: Lists and for loop

  http://software-carpentry.org/v4/python/lists.html

Practice Objectives: lists

- Create a list called *cities* that contains 5 strings of the following city names in order:
  - Louisville, London, Paris, New York, Barcelona
- Select the city “Paris” out of the list
- Select the last city, “Barcelona”, out of the list
- Slice out the cities “Louisville”, “London”, “Paris”
- Append a new city to the end of the list
- Sort the list in alphabetical order
- What is the length of the list

Practice Objectives: lists and for loop

- Write a for loop to print out each city name in the list called *cities*.

- Write a program that tests if the cities London, Detroit, Miami, Cincinnati, Paris are in the list called *cities*. If item is in *cities*, print the following:

  City <*name of city*> is already in list cities

  else add city to list *cities* and print the following:

  Added the city <*name of city*> to the list cities

Review – page 1

1. What are some features of Python's lists?

    a) Ordered sequence, arbitrary type, mutability
    b) Arbitrary sequence, fixed type, mutability
    c) Arbitrary sequence, arbitrary type, mutability
    d) Ordered sequence, arbitrary type, immutability
    e) Arbitrary sequence, fixed type, immutability

2. What function could you use to construct a list of odd numbers from 0 to 10?

Next meeting

- Python – Learn about built-in containers (collections)
  - **more on lists and for loop**
  - **dictionaries**