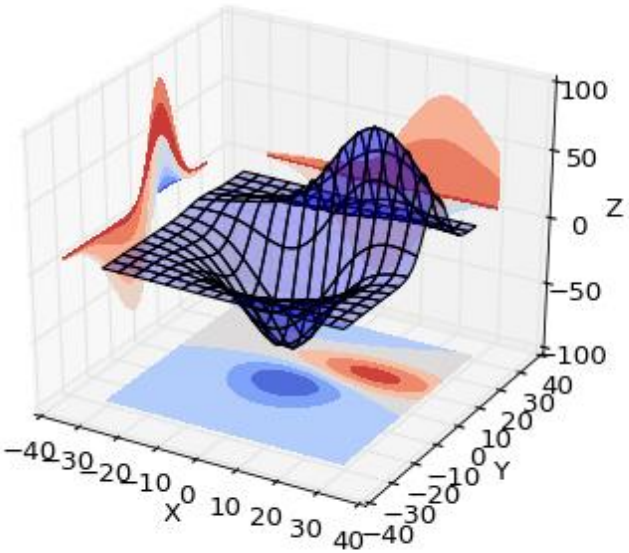
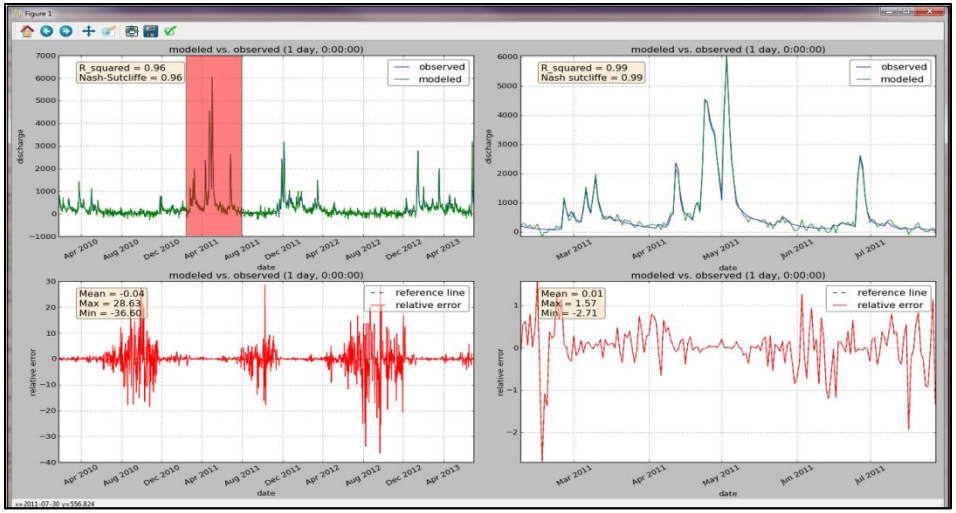
Introduction to Scientific Computing
Meeting 18
Programming with Python



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

Last Meeting

- **Lists** are **mutable**
  - Example:
- **Lists** can contain **arbitrary types**
  - Example?
- **Lists** have **length**
  - Name of function?
- **Lists** have **methods** that operate on a list
  - Examples ?
- **Function** that creates a list of numbers
  - ?
- Can **test** for **item membership** in **list**
  - Example?

Last Meeting

- 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

Last Meeting

- 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?

What we have learned so far?

- How to use Python interactively vs. writing a script
- Creating variables through assignment; **=**
  - **Variable is a name for a value(s)**
- Basic data types; **integers, floats, booleans, strings**
- Simple arithmetic operations; **+, -, %, *, ****
- Comparisons; **True or False**
- Printing to screen; **print()**
- Iteration/repeat operations; **while** and **for** loops
- How to make decisions/selections; **if, elif, else** statement
- Container/collection data types; **list**

Today’s Objectives

- Learn about main built-in container called a **dictionary**

Demo - dictionary

- **Dictionary** is a **container** or a **collection of items**.
  - Unordered collection of key/value pairs
  - Mapping/association between keys and values

dictionary = {*"key"*: *value, "key"*: *value***}**

```
>>> info = {"name": "Bob", "age": 30, "height": 6}
>>> info["name"]
Bob
>>> info["age"]
30
```

Demo - dictionary

- **Dictionary** is a **container** or a **collection of items**.
  - Unordered collection of key/value pairs
  - Mapping/association between keys and values

dictionary = **{** *"key"*: *value, "key"*: *value* **}**

```
>>> info = {"name": "Bob", "age": 30, "height": 6}
>>> info["name"]
Bob
>>> info["age"]
30
```

Demo - dictionary

- Unlike a **list** where you must **use an integer index** to **access items**, in a **dictionary**, you **use a key** to **access items**.

```
>>> info_list = [“Bob”, 30,  6,]
>>> info_list[0]
Bob
>>> info_list[1]
30

>>> info_dict = {“name”: “Bob”, “age”: 30, “height”:  6}
>>> info_dict[“name”]
Bob
>>> info_dict[“age”]
30
```

Demo - dictionary

- **Adding items** to a dictionary after it is created

```
>>> info_dict[“weight”] = 160
>>> info_dict[“hobbies”] = [“golf”, “tennis”]
```

- Test whether key is present using **in**

```
>>> “weight” in info_dict
True
>>> “birthday” in info_dict
False
```

Demo - dictionary

- Dictionaries have **methods**
  - https://docs.python.org/2/library/stdtypes.html

```
>>> temperature = {“measurements”: [50, 60, 70],
                   “units”: “Fahrenheit”}
>>> temperature.keys()
[“units”, “measurements”]
>>> temperature.values()
[“Fahrenheit”}, [50, 60, 70]]
>>> temperature.get(“measurements”)
[50, 60, 70]
>>> temperature.items()
[(“units”, “Fahrenheit”), (“measurements”, [50, 60, 70])]
```

Demo - dictionary

- **Print all keys and key/value pairs**

```
>>> for key in info_dict.keys():
. . .     print(key)

>>> for values in info_dict.values():
. . .     print(values)

>>> for key, value in info_dict.items():
. . .     print("Key {} maps to value {}".format(key, value))

>>> for key, value in info_dict.iteritems():
. . .     print("{} maps to {}".format(key, value))
```

Video – Python Basics

software carpentry

- Software Carpentry, Greg Wilson
  - Python: Dictionaries

  http://software-carpentry.org/v4/setdict/dict.html

Practice Objectives: dictionary

- Create a dictionary called *states* that has a mapping of state name to state abbreviation.

  states = {“Kentucky”: “KY”, “Indiana”: “IN”}

- Create a dictionary called *cities* that has a mapping of state abbreviation to city name.

  cities = {“KY”: “Louisville”, “IN”: “Indianapolis”}

- Add another key/value pair to states and another key/value pair to cities
- Using **states** print abbreviation for Kentucky
- Using **cities** print city for Indiana

Practice Objectives: dictionary

- Print using the state and then the cities dictionary

```
print("Kentucky has: {}".format(cities[states["Kentucky"]]))
```

- Print all the state abbreviations

```
for state, abbrev in states.items():
    print("{} is abbreviated {}".format(state, abbrev))
```

- Print all the cities

```
for abbrev, city in cities.items():
    print("{} has the city {}".format(abbrev, city))
```

Practice Objectives: dictionary

- Print state, abbrev, and city

```
for state, abbrev in states.items():
    print(state, abbrev, cities[abbrev))
```

- Format the print a bit

```
for state, abbrev in states.items():
    print("{} state is abbreviated {} and has city
".format(state, abbrev, cities[abbrev))
```

Review – page 1

- Explain how lists and dictionaries are the same

- Explain how lists and dictionaries are different

- When to use a list and when to use a dictionary?

Next meeting

- Python – Learn about
  - **Strings**
  - **Input and output**