Algorithmic Approaches for Biological Data, Lecture #3

Katherine St. John

City University of New York
American Museum of Natural History

27 January 2016

Outline

- More on Functions

Outline

- More on Functions
- Simple Input

Outline

- More on Functions
- Simple Input
- Program Design: top-down design, stepwise refinement

Recap

- Modules: `turtles`, `random`, `math`

Recap

- Modules: `turtles`, `random`, `math`
- Control structures: `for`

Recap

- Modules: `turtles`, `random`, `math`
- Control structures: `for`
- Built-in functions: `print`, `range()`

Recap

- Modules: `turtles`, `random`, `math`
- Control structures: `for`
- Built-in functions: `print`, `range()`
- Design patterns: accumulator, input-process-output.

Recap

- Modules: `turtles`, `random`, `math`
- Control structures: `for`
- Built-in functions: `print`, `range()`
- Design patterns: accumulator, input-process-output.
- Overview of functions

More on Functions

- *Think CS* function example (Alex the turtle drawing squares)

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

More on Functions

- *Think CS* function example (Alex the turtle drawing squares)
- Parts of a function:

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

More on Functions

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

- *Think CS* function example (Alex the turtle drawing squares)
- Parts of a function:
    - Header: refers to the line beginning with `def` that includes the name and parameters

More on Functions

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

- *Think CS* function example (Alex the turtle drawing squares)
- Parts of a function:
    - Header: refers to the line beginning with `def` that includes the name and parameters
    - Inputs: also called arguments or parameters

More on Functions

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

- *Think CS* function example (Alex the turtle drawing squares)
- Parts of a function:
  - Header: refers to the line beginning with `def` that includes the name and parameters
  - Inputs: also called arguments or parameters
  - Body: the commands inside the function

More on Functions

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

- *Think CS* function example (Alex the turtle drawing squares)
- Parts of a function:
  - Header: refers to the line beginning with `def` that includes the name and parameters
  - Inputs: also called arguments or parameters
  - Body: the commands inside the function
  - Outputs: also called return values

Local Variables

Standard form:

```
def myFunc(in1,in2,...):
    command1
    command2
    ...
    return(out1,out2,...)
```

- Variables created inside a function, exist only in that function.

Local Variables

Standard form:

`def myFunc(in1,in2,...):`
    command1
    command2
    ...
    return(out1,out2,...)

- Variables created inside a function, exist only in that function.
- Any changes to local variables don't affect variables outside the function.
- *Think CS* demo (variables & local parameters)

Local Variables

Standard form:

```
def myFunc(in1,in2,...):
    command1
    command2
    ...
    return(out1,out2,...)
```

- Variables created inside a function, exist only in that function.
- Any changes to local variables don't affect variables outside the function.
- *Think CS* demo (variables & local parameters)
- Good style: only use values passed to the function in calculations.

Local Variables

Standard form:

```
def myFunc(in1,in2,...):
    command1
    command2
    ...
    return(out1,out2,...)
```

- Variables created inside a function, exist only in that function.
- Any changes to local variables don't affect variables outside the function.
- *Think CS* demo (variables & local parameters)
- Good style: only use values passed to the function in calculations.
- Any changes to local variables don't affect variables outside the function.

Functions calling Functions

- Functions can call other functions.

Functions calling Functions

- Functions can call other functions.
- *Think CS* demo

Functions calling Functions

- Functions can call other functions.
- *Think CS* demo
- Group Work: *Think CS*, Flow of Execution Summary Questions

Recap

- Lab at 3pm.

Recap

- Lab at 3pm.
- Email lab reports to `kstjohn@amnh.org`

Recap

- Lab at 3pm.
- Email lab reports to `kstjohn@amnh.org`
- Challenges available at `rosalind.info` (use emailed link to access course page).