• Questions/Announcements

• Testing and Debugging
  • doctest module

• Midterm 2 Review (Q/A)

More than the act of testing, the act of designing tests is one of the best bug preventers ...
Testing

Aspects of Testing

- program runs
- program does not have bugs
- program handles the unexpected gracefully
- program runs under extreme conditions
- program is usable
Testing

Integrative testing
-- looks at behavior of the whole program/system

Unit testing
-- look at one isolated component
Testing

Testing starts at program design time

Function design recipe:
Inputs/outputs and types
Short description of function
Examples of how the function should work
Testing

Testing starts at design time
Examples of how the function works

• simple cases - can be done by hand
• general cases – all paths
• boundary cases – 0, 1, large numbers, negative numbers, empty string, string len 1, etc.
def to_celsius(f):
    """ (number) -> float
    convert input fahrenheit temperature to celsius and return celsius value
    ""

    pass

    return #cel_temp
Testing

```python
>>> to_celsius(32):
    0.0
>>> to_celsius(212)
    100.0
>>> to_celsius(0)
    -17.78
>>> to_celsius(1)
    -17.72
>>> to_celsius(-10)
    -23.33
>>> to_celsius(75)
    23.89
```
Testing

def to_celsius(f):
    
    convert input fahrenheit temperature to celsius and return celsius value

    
cel_temp = f - 32 * 5/9

return cel_temp
Testing

def watch_out(n):
    """(int) -> float"

    return 1 / n

>>> watch_out(10)
 .1
 """

print('1/' + n)
return 1 / n
Testing

def watch_out(n):
    """(int) -> float"

    return 1 / n

>>> watch_out(10)
0.1
""

    if n == 0:  # add a guard
        return 'Error'  # maybe better return value? precondition?
    else:
        print('1/' + str(n))
        # print('1/ { }'.format(n))
    return 1 / n
Errors

Syntax

Runtime

Semantic/logical
Summary of New Topics for Midterm 2

- Boolean data type/relational operators
- Conditional statements (if)
- Looping (while, for)
- Turtle graphics
- String methods (that return string, Bool)
Summary of Topics, 1.

- Programming = Computational Thinking + Coding
- High level languages: Python, etc.

- Python primitive elements
- Objects (have types)
- Values
  - Operators/Functions
  - Id
- Expressions – primitive elements and combinations of primitive elements - evaluate to a value
- Assignment statements (are not expressions) - associate names/values
- Variables are expressions
Summary of Topics, 2.

– Python built in data types – values and operations:
  – numbers – integers and floating point
  – strings

– Python built in functions and operators:
  • + - * // % **
  • round pow abs min max len int float str print input type help

– User-defined functions:
  • defining and calling
  • parameters and arguments
  • local variables
  • returning a value or not returning a value
  • function design recipe
  • memory model
CIS 122 Introduction to Programming Spring 2015

Summary of Topics, 3.

- Python input and output (print)
- Python modules – math, random, turtle, etc.
Functions Concepts Checklist

• Python built-in functions
• combining functions
• user-defined functions
• keywords (def, return)
• parameters, arguments
• local variables
• output/returned value
• defining v. calling
• memory model/namespace
• function design recipe
Tricky Concepts Checklist

• assignment statement v. expression stmt
• defining v. calling a function
• parameters v. arguments
• return v. print output
• functions always return a value (it may be None)
• variables v. strings
• parameters v. local variables v. input