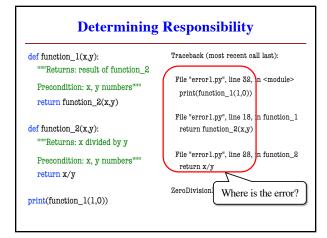


1 2



Assert Statements

Form 1: assert <boolean>
Does nothing if boolean is True
Creates an error is boolean is False
Form 2: assert <boolean>, <string>
Very much like form 2
But error message includes the string
Statement to verify a fact is true
Similar to assert_equals used in unit tests
But more versatile with complete stack trace

3

def anglicize(n):

"""Returns: the anglicization of int n.

Precondition: n an int, 0 < n < 1,000,000"""

assert type(n) == int, repr(n)+' is not an int'

assert [0 < n and n < 1000000] repr(n)+' is out of range'

Implement prodod here...

Check (part of)
the precondition

Error message
when violated

def lookup_netid(nid):

"""Returns: name of student with netid nid.

Precondition: nid is a string, which consists of 2 or 3 letters and a number"""

assert type(nid) == str, repr(nid) + ' is not a string' assert nid.isalnum(), repr(nid)+' is not letters/digits'

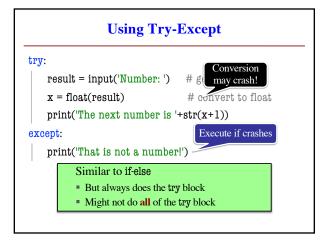
Returns True if s contains only letters, numbers.

Does this catch all violations?

5

1

def exchange(curr_from, curr_to, amt_from): """Returns: amount of curr_to received. Precondition: curr_from is a valid currency code Precondition: curr_to is a valid currency code Precondition: amt_from is a float""" assert ??????, repr(curr_from) + ' not valid' assert ??????, repr(curr_from) + ' not valid' assert type(amt_from)==float, repr(amt_from)+' not a float'



8

def isfloat(s):

"""Returns: True if string
s represents a float"""

try:

| x = float(s)
| return True | If attempt succeeds, string s is a float
except:

| return False | Otherwise, it is not

Try-Except and the Call Stack # recover.py • Error "pops" frames off stack Starts from the stack bottom def function_1(x,y): Continues until it sees that current line is in a try-block return function_2(x,y) Jumps to except, and then except: proceeds as if no error return float('inf') line in a try function_1 **def** function_2(x,y): return function_3(x,y) function_2 def function_3(x,y): function_3 return x/y # crash here

9 10

Tracing Control Flow def first(x): def third(x): print('Starting first.') print('Starting third.') assert x < 1print('Ending third.') second(x) except: print('Caught at first') What is the output of first(2)? print('Ending first') def second(x): print('Starting second.') third(x) print('Caught at second') print('Ending second')

Tracing Control Flow def first(x): def third(x): print('Starting third.') print('Starting first.') assert x < 1second(x) print('Ending third.') except: print('Caught at first') What is the output of first(0)? print('Ending first') def second(x): print('Starting second.') third(x) print('Caught at second') print('Ending second')

11 12

2