Fruitful Functions

CSC 1200 - Principles of Computing

Overview

- Return Values
- Multiple Return Statements
- Boolean Return Values
- Checking Types
- Incremental Development

Return Values

- Recall that fruitful functions have return values.
  - The purpose of the function is to produce a result (rather than just perform a task).
  - Calling the function generates a value.
- Typical ways to use fruitful functions:
  - Fruitful functions are often used in place of a value in an expression:
    - cyl_vol = area(base_radius) * height
    - tri_area = (1 / 2) * s1 * s2 * math.sin(incl_angle)
  - Fruitful functions are also used to assign a value to a variable that is used in the program.
    - x_bar = average(x1, x2)
    - choice = get_user_input()
    - NOTE: if you just *call* a fruitful function but do not *assign* it to a variable, the value produced by the function will be lost!

Multiple Return Statements ☹

- The author states, "Sometimes it is useful to have multiple return statements" and gives the following example:

```
def absolute_value( x ):
        if x < 0:
                return -x
        else:
                return x
```

- While it is true that you *can* have multiple return statements, software engineers frown on such things. It is best practice to **have a single return**.

```
def absolute_value( x ):
        if x < 0:
                answer = -x
        else:
                answer = x
        return answer
```

Boolean Return Values

- To make code more readable, it is often convenient to encapsulate complicated test conditions inside a function that returns a Boolean value.

- Example: Write a function is_between(x, y, z) that returns True if x is between y and z, inclusive, and False otherwise. Then use this function to determine if a user input value is valid.

Checking Types

- We can use the built-in function isinstance to verify the type of a variable.

```
def print_type( x ):
    if isinstance( x, int ):
        print(x,'is an integer')
    elif isinstance( x, str ):
        print(x,'is a string')
    elif isinstance( x, float ):
        print(x,'is a float')

print_type( 2 )
print_type( '2' )
print_type( 2.0 )
```

```
= RESTART: C:/Users/bgannod
ogram.py
2 is an integer
2 is a string
2.0 is a float
>>>
```

Incremental Development

- When developing more complex programs, it is a good idea to write small, logical portions of code and test them before moving on.
  - Start with a working program and make small incremental changes. At any point, if there is an error, you should have a good idea where it is.
  - You can use temporary variables to hold intermediate values so you can display and check them.
- Example: Write a program that asks the user for the coordinates of three points that represent the vertices of a triangle, then determine the lengths of the sides and the angles of the triangle.