Lecture 5: Functions

CS 51P

September 19, 2022

Review: Expressions

- Values
  - 47
  - "hello, world!\n"
  - True
- Variables
  - x
  - i
  - char
- Operations on values or variables
  - 1 * 2 * 3
  - "hello" + "world
  - x % 2
- Function calls
  - int("32")
  - print("hello, world")
  - str.isdigit("12345678")

Functions

- A function is a named sequence of instructions that performs some useful operation

- When you call a function, the sequence of instructions executes.

- A function call is an expression (it evaluates to a value)

- When should you define a function?
- How can you define your own functions?
- How do you use (call) your own functions?

Defining Functions

- Why?
  - There's some useful operation that you want to do over and over and over
  - Easier to read/understand
  - Easier to modify/change/debug

```
++++++++
++ ** ++
++ ** ++
++++++++
```

- How?

header

body

```
def print_logo():
    s1 = (8*'+')+'\n'
    s2 = '++ ** ++\n'
    print(s1+s2+s2+s1)
```

Calling Functions

```
def print_logo():
     s1 = (8*'+')+'\n'
     s2 = '++ ** ++\n'
     print(s1+s2+s2+s1)

print("Here's my company logo:")
print_logo()
print("I can easily print it as many times"
      + "as I need to")
print_logo()
```

Exercise: Defining a Function

- Define a function `print_flag()` that prints the following image:

```
9::::=======
|::::=======
|===========
|===========
|
|
|
```

- Write a program that asks the user for a positive integer and then prints that number of flags

Function Evaluation

- Functions calls are expressions, i.e. they evaluate to a value
  - int("47") evaluates to 47
  - str.isdigit("hello") evaluates to False
  - input() evaluates to the string the user enters

- We can store the value that an expression evaluates to in a variable
  - num = int("47")
  - is_pos_int = str.isdigit("hello")
  - input_str = input()

- What value does the expression print_flag() evaluate to?

Return Values

- keyword **`return`** defines a value for the function to evaluate to

```
def one():
    return 1



print(one())
three = 2*one()+one()
```

- function immediately terminates ("returns") when a return statement is executed
- if a function terminates without executing a return statement, it evaluates to the default value None (type is NoneType)

Example

- Define a function get_string_with_upper() that repeatedly asks the user for a string until the user enters a string with at least one upper case letter and then returns that string.

- Define a function get_string_with_2_upper() that gets two strings from the user, each of which must contain at least one upper case letter, and then returns the concatenation of those two strings.

- Write a program that calls get_string_with_2_upper and prints the value that function evaluates to.

Exercise

- Define a function get_pos_int() that repeatedly asks the user for an input until the user enters a positive integer and then returns that number as an int.

- Write a program that gets a positive integer from the user (using get_pos_int()) and then prints that number of flags (using print_flag())