Lecture 6: Parameterized Functions

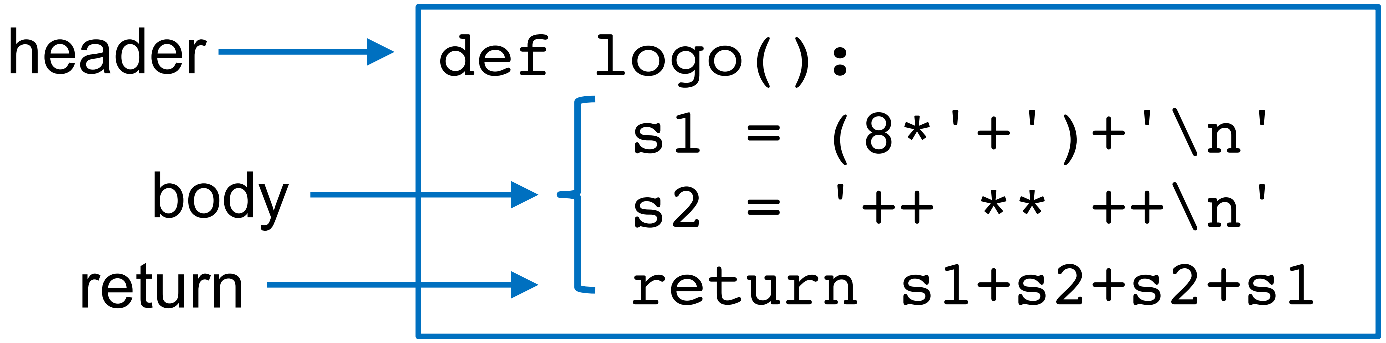
CS 51P

September 21, 2022

Review: 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?



```
def logo():
    s1 = (8*'+')+'\n'
    s2 = '++ ** ++\n'
    return s1+s2+s2+s1
```

Review: Calling Functions

```
def logo():
     s1 = (8*'+')+'\n'
     s2 = '++ ** ++\n'
     return s1+s2+s2+s1

design = logo()
print(design)

# or

print(logo())
```

Example

- Define a function called good_choice() that asks the user for a positive integer and evaluates to True if the user enters 13 and False if they enter anything else?
- We want to be able to use the function as follows:

```
def main():
  if good_choice():
    print("yay")
  else:
    print("boo")
```

What if you wanted your good_choice function to be able to check for numbers other than 13?

Parameterized Functions

- Functions can be defined with **parameters**, special variables that can be used inside the function and that are defined when the function is called

- Defining a parameterized function:

parameter

```
def good_choice(n):
    x = int(input("pos int?\n"))
    return x == n
```

- Calling a parameterized function:

argument

```
b = good_choice(13)
```

Example: Parameterized Functions

- Define a function called `square` that takes a number n (an `int` or `float`) as a parameter and returns that number squared

- Define a function called `sum_squares` that takes a number n (an `int`). If the number is a positive `int`, it returns the sum of the squares 1,…, n. Otherwise it returns 0.

Exercise

- Define a function `is_pos_int` that takes a string and returns True if the string represents an integer value and False otherwise

- Write a function `main` that uses the functions `is_pos_int` and `sum_squares` to get a positive integer from the user and then print the sum of the squares from 1 to that number

Example: Multi-parameter Functions

- Define a function called `area` that takes two numbers l and w (an `int` or `float`) as parameters and returns the area of a rectangle with length l and width w

Exercise

- Define a function called `exp` that takes a number n (an `int` or `float`) and a number p (an int or float) as parameters and returns the value n^p

- Define a function called `sum_powers` that takes a number n (an `int or float`) and a power p (an `int or float`). If n is a positive `int`, it returns the sum of the powers $1^p + 2^p$ ... $+ n^p$. Otherwise it returns 0.

Main functions

- By convention, the only code that goes in the body of a Python file is the two-line program

```
if __name__ == "__main__":
    main()
```

- The rest of the program is defined in a function called main()
- (or in other functions!)

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


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


if __name__ == "__main__":
    main()
```

Docstrings

- "A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition."
- every file should start at the top with a multiline comment that gives the author, date, description of what the code does
- every function header should be followed by a multiline comment that describes what the function does, specifies any input parameters, and specifies the return type/value

```
def square(n):
    """
    Computes the square of n
    :param n (int or float): a number
    :return (int or float): n*n
    """
    return n * n
```