Lecture 6: Functions

CS 51P

September 20, 2023

Review: Expressions

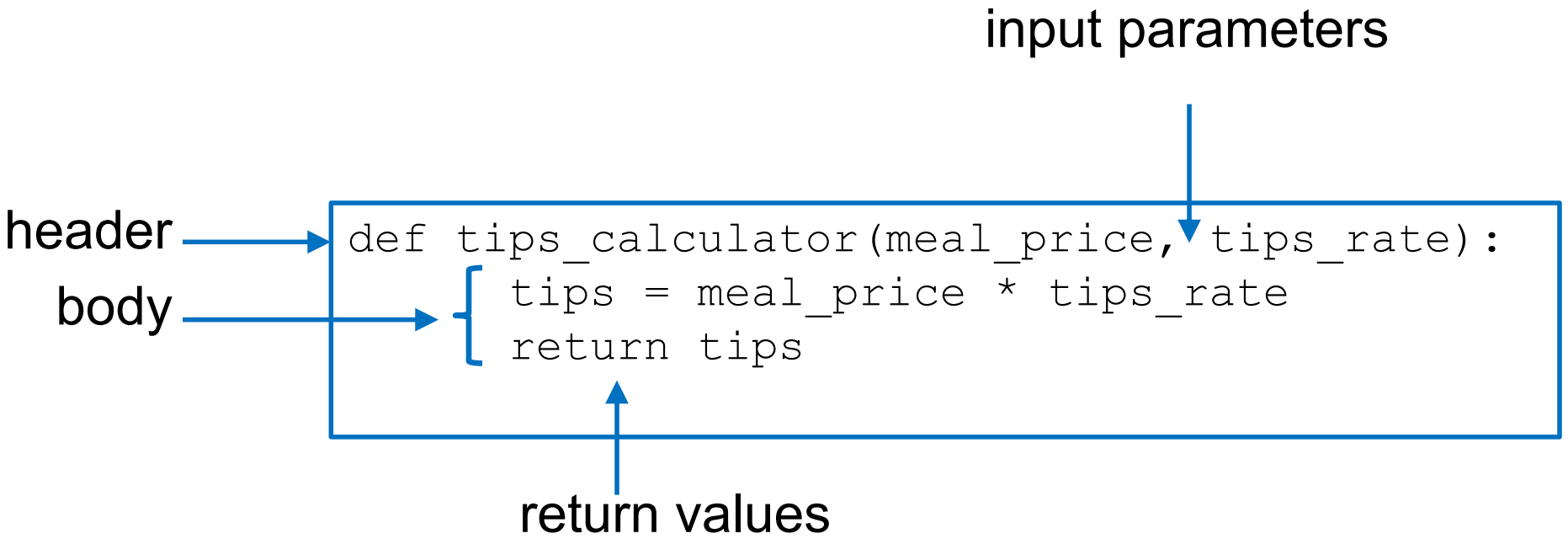
- Values
  - 47
  - "hello, world!\n"
  - True
- Variables
  - x
  - i
  - char
- Operations on values or variables
  - 1 * 2 * 3
  - "hello" + "world
  - x % 2
- Function calls
  - int("32")
  - print("hello, world")
  - str.isdigit("12345678")

Functions

- A function is like a helper or assistant
- When you call a function, it will do a certain job for you automatically.
  - tips_calculator(meal_price, tips_rate)
  - roofing_price(square_feet, unit_price)
- Benefits:
  - Automate the operations
  - Reusable

Defining Functions

- How to define a function?



Don’t forget the indentation!

Calling Functions

- How to use or call a function?

```
def roofing_price(square_feet, unit_price):
     total_price = square_feet * unit_price
     return total_price

price = roofing_price(2000, 10)
print(”The estimated roofing price is ” +
str(price)”)
```

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()

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

Exercise

- Define a function that takes in two numbers as input, e.g., num1 and num2, and then return the average value of these two numbers. Practice to call/use this function.

- Define a function that takes in two numbers as input, e.g., num1 and num2, and then return the absolute value of deducting num1 by num2 (e.g., |num1 – num2|). Practice to call/use this function.

Functions Summary

- 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)
- How can you define your own functions?
- How do you use (call) your own functions?
- When should you define a function?
  - There's some useful operation that you want to do over and over and over

Revisit return

- function immediately terminates ("returns") when a return statement is executed

```
def add(num1, num2):
    result = 0
    return result
    result = num1 + num2
```

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

```
def tips_calculator(meal_price, tips_rate):
    tips = meal_price * tips_rate

print(tips_calculator(20, 0.18))
```

More about return

- The return value can be int, float, Boolean, str, and etc.
- Sometimes, we don't need to return anything
- For example, print out a pyramid of "*" based on the given parameter, e.g., pyramid(4) will display a pyramid as below

```
*
**
***
****
```

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

Exercise

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