CS 110 - Functions and Return

Benjamin Dicken

DID YOU FINISH WRITING THE SOFTWARE?
NO.
I SPENT THE LAST THREE DAYS SETTING UP MY PROGRAMMING ENVIRONMENT.
SO... YOU'VE DONE... NOTHING?
NOTHING YOU'D UNDER-STAND.
Dilbert.com @ScottAdamsSays
1-2-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel

Returning a value

- Using we can send a value to a function using **arguments** and **parameter variables**.
- We can also **return** values from a function using the **return** statement
- It is often useful to have a function yield a particular value

```
def function_name():
    statementA
    . . .
    statementN

statement . . .

function_name()

statements . . .
```

```
def function_name():
    statementA
    . . .
    return n

statement . . .

var = function_name()

statements . . .
```

```
def function_name():
    statementA
    . . .
    return

statement . . .

function_name()

statements . . .
```

```
def function_name():
    statementA
    if …:
        return
    statementY

statement . . .

function_name()

statements . . .
```

```
def categorize(height):
    if height > 70:
        return "tall"
    else:
        return "short"

statements . . .

category_1 = categorize(75)
category_2 = categorize(65)

statements . . .
```

Activity

What would this print?

```
def repeat(content, times):
    to_return = ''
    i = 0
    while i < times:
        to_return += content
        i += 1
    return to_return

result = repeat('110', 5)
print(result)
```

Activity

The pythagorean theorem

https://en.wikipedia.org/wiki/Pythagorean_theorem

$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}.$$

Activity

The pythagorean theorem

- Write a function that accepts two ints as parameters
- These represent the length of the two non-hypotenuse sides
- Returns the length of the hypotenuse

$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}.$$

```
# return 5.0
pythagorean(3, 4)
# return 14.142135623730951
pythagorean (10, 10)
```

Implement the pythagorean function

```python
def pythagorean(a, b):
    c_squared = (a**2 + b**2)
    c = (c_squared)**0.5
    return c
```

Implement the pythagorean function

```python
def pythagorean(a, b):
    return (a**2 + b**2)**0.5
```

```python
def pythagorean(a, b):
    '''
     Calculates the length of c (the hypotenuse) of a right triangle using
     the pythagorean theorem.
     a and b: The length of the sides of a right-triangle that are adjacent
              to the right-angle.
     Returns an integer that is the calculated length of side c.
     '''
    c_squared = (a**2 + b**2)
    c = (c_squared)**0.5
    return c

def main():
    a_value = float(input('Enter a value: '))
    b_value = float(input('Enter b value: '))
    result = pythagorean(a_value, b_value)
    print(result)

main()
```

Multiple return

- It is possible to return multiple values from a function
- As with arguments and parameters, use comma-separated list

```
def function_name():
    statementA
    . . .
    return a

statement . . .

r1 = function_name()

statements . . .
```

```
def function_name():
    statementA
    . . .
    return a, b

statement . . .

r1, r2 = function_name()

statements . . .
```

```
def function_name():
    statementA
    . . .
    return a, b, c

statement . . .

r1, r2, r3 = function_name()

statements . . .
```

Activity

What will this program print?

```
def compute_a_sum(number):
    i = 1
    a_sum = 0
    while i <= number:
        a_sum += i
        i += 2
    return number, i, a_sum

def main():
    hopefully_an_integer = int(input('Enter a value:\n')) # 4
    result_1, result_2, result_3 = compute_a_sum(hopefully_an_integer)
    print(result_1, result_2, result_3)

main()
```

Activity

Write the **min_max** function

The min_max function should have three parameters.

The function should return *both* the minimum and maximum value.

For example:

```
minimum, maximum = min_max(40, 70, 10)
print(minimum, maximum)
```

Should print:

```
10, 70
```

```python
def min_max(a, b, c):
    '''
    This function accepts three numbers and returns two values: The min and max
    a, b, c: Can be any integer or float values
    returns: Two numbers. First the minimum, and then the maximum.
    '''
    minimum = a
    maximum = a
    if b >= c >= a or b >= a >= c:
        maximum = b
    elif c >= b >= a or c >= a >= b:
        maximum = c
    if b <= c <= a or b <= a <= c:
        minimum = b
    elif c <= b <= a or c <= a <= b:
        minimum = c
    return minimum, maximum
```

String slicing

- In class, we already discussed **string indexing**
  - With **string indexing**, you can grab an individual character from a string using square brackets
- You can also grab a sub-sequence of characters in a string with **string slicing**

String slicing

- In class, we already discussed **string indexing**
    - With **string indexing**, you can grab an individual character from a string using square brackets
- You can also grab a sub-sequence of characters in a string with **string slicing**

```
name = 'Jeremiah'
print(name[1:5])
print(name[0:3])
print(name[3:])
print(name[:3])
```

Print `'where are eagles'` with three slices

```python
movie = 'where eagles dare'
```

Print `'where are eagles'` with three slices

```python
movie = 'where eagles dare'
word_1 = movie[0:5]
word_2 = movie[14:]
word_3 = movie[6:12]
print(word_1, word_2, word_3)
```

Activity

Implement the function

- Write a function named **same_halves** that has a single string parameter
- Returns **True** if the first half of the string is the same as the second half
- Otherwise, return **False**

```python
print(same_halves('abcdabcd'))          # True
print(same_halves('another'))           # False
print(same_halves('123__321'))          # False
print(same_halves('123__123'))          # False
print(same_halves('123_4567123_4567'))  # True
```

```python
def same_halves(string):
    half_len = int(len(string)/2)
    first_half = string[:half_len]
    second_half = string[half_len:]
    if first_half == second_half:
        return True
    else:
        return False
```

```python
def same_halves(string):
    half_len = int(len(string)/2)
    first_half = string[:half_len]
    second_half = string[half_len:]
    return first_half == second_half
```

```python
def same_halves(string):
    half_len = int(len(string)/2)
    return string[:half_len] == string[half_len:]
```

```python
def same_halves(string):
    '''
    This function determines if the first half of a string is
    Identical to the second half of the string.
    string: any string of character.
    '''
    return string[:int(len(string)/2)] == string[int(len(string)/2):]
```

Scope

- Every variable that is created has a particular **scope**
- The **scope** of a variable is the range(s) of code over which that variable can be used or modified

Local and Global

- **Local Variable:** Is a variable with local scope
    - For example: A variable assigned inside of a function can only be used or modified within that function after the initial assignment
- **Global Variable:** Is a variable with global scope
    - For example: a variable declared outside of a function can be accessed or modified across multiple functions

Activity

How many global variables?

```
def calculate():
    total_pay = 0
    total_hours = 0
    index = 1
    while index <= weeks:
        pay = int(input('Week ' + str(index) + ' pay: '))
        hours = int(input('Week ' + str(index) + ' hours worked: '))
        total_pay += pay
        total_hours += hours
        index += 1
    return total_pay, total_hours

weeks = int(input('How many weeks of work? '))
total_pay, total_hours = calculate()
average_weekly_pay = total_pay / weeks
average_hourly_wage = total_pay / total_hours
print('Your AWP was $' + format(average_weekly_pay, ',.2f'))
print('Your AHW was $' + format(average_hourly_wage, ',.2f') + ' per hour')
```

How many global variables?

```
def calculate():
    total_pay = 0
    total_hours = 0
    index = 1
    while index <= weeks:
        pay = int(input('Week ' + str(index) + ' pay: '))
        hours = int(input('Week ' + str(index) + ' hours worked: '))
        total_pay += pay
        total_hours += hours
        index += 1
    return total_pay, total_hours

weeks = int(input('How many weeks of work? '))  # 1 Here
total_pay, total_hours = calculate()             # 2 Here
average_weekly_pay = total_pay / weeks           # 1 Here
average_hourly_wage = total_pay / total_hours    # 1 Here
print('Your AWP was $' + format(average_weekly_pay, ',.2f'))
print('Your AHW was $' + format(average_hourly_wage, ',.2f') + ' per hour')
```

Activity

How many local variables?

```
def calculate():
    total_pay = 0
    total_hours = 0
    index = 1
    while index <= weeks:
        pay = int(input('Week ' + str(index) + ' pay: '))
        hours = int(input('Week ' + str(index) + ' hours worked: '))
        total_pay += pay
        total_hours += hours
        index += 1
    return total_pay, total_hours

weeks = int(input('How many weeks of work? '))
total_pay, total_hours = calculate()
average_weekly_pay = total_pay / weeks
average_hourly_wage = total_pay / total_hours
print('Your AWP was $' + format(average_weekly_pay, ',.2f'))
print('Your AHW was $' + format(average_hourly_wage, ',.2f') + ' per hour')
```

How many local variables?

```
def calculate():
    total_pay = 0           # 1 Here
    total_hours = 0         # 1 Here
    index = 1               # 1 Here
    while index <= weeks:
        pay = int(input('Week ' + str(index) + ' pay: '))                 # 1 Here
        hours = int(input('Week ' + str(index) + ' hours worked: '))  # 1 Here
        total_pay += pay
        total_hours += hours
        index += 1
    return total_pay, total_hours

weeks = int(input('How many weeks of work? '))
total_pay, total_hours = calculate()
average_weekly_pay = total_pay / weeks
average_hourly_wage = total_pay / total_hours
print('Your AWP was $' + format(average_weekly_pay, ',.2f'))
print('Your AHW was $' + format(average_hourly_wage, ',.2f') + ' per hour')
```

Activity

```
name = 'NAME'

def process_name():
    name = input('Type your name: ')
    first_letter = name[0]
    after_first = name[1:]
    name = first_letter.upper() + after_first.lower()

process_name()

print('Hi there', name)
```

What will this print?

Input: jACOB

Activity

```
name = 'NAME'

def process_name():
    name = input('Type your name: ')
    first_letter = name[0]
    after_first = name[1:]
    name = first_letter.upper() + after_first.lower()
    print('Hi there', name)

process_name()
```

What will this print?

Input: jACOB

What will this print?

```
def process_name(name_to_process):
    first_letter = name_to_process[0]
    after_first = name_to_process[1:]
    name = first_letter.upper() + after_first.lower()

name = input('Type your name: ')
process_name(name)

print('Hi there', name)
```

Input: jACOB

What will this print?

```
def process_name(name_to_process):
    first_letter = name_to_process[0]
    after_first = name_to_process[1:]
    name_to_process = first_letter.upper() + after_first.lower()

name = input('Type your name: ')
process_name(name)

print('Hi there', name)
```

Input: jACOB

Activity

What will this print?

```
def process_name(name_to_process):
    first_letter = name_to_process[0]
    after_first = name_to_process[1:]
    return first_letter.upper() + after_first.lower()

name = input('Type your name: ')
name = process_name(name)

print('Hi there', name)
```

Input: jACOB

Activity

```
name = input('Type your name: ')

def process_name():
    first_letter = name[0]
    after_first = name[1:]
    name = first_letter.upper() + after_first.lower()

process_name()

print('Hi there', name)
```

What will this print?

Input: jACOB