CS 110 - Functions and Return

Adriana Picoral (she/her/hers)

Find your NEW GROUP on the course website!

Announcements

- New groups
- Regrade requests are still open (you have 5 days to request a regrade from the time the grades are posted)
- Programming Assignment 5 - careful re hardcoding
- Style guide – functions
- D2L grades

Announcements

- Registration

Function Comments

- Important to document
  - What each function does
  - The expected ***type*** and ***purpose*** of each parameter variable
    - Otherwise, how would you know what to pass in?
- This should be done with a multi-line string

```
def validate_input(input_type, length):
    '''
    Asks the user for an input and exits if the input is not
    alphabetical, capitalized, and of the correct length.
    input_type: a string label for the input prompt and error message
    length: an int, representing the max length of the input string
    '''
    value = input('Enter ' + input_type + ': ')
    if not value.isalpha() or len(value) > length or not value[0].isupper():
        print('Invalid ' + input_type + '.')
        exit()

validate_input('first name', 15)
validate_input('middle initial', 1)
validate_input('last name', 30)

print('Valid name!')
```

main()

- ***No code should be without a function*** ***
- Standard practice in programming to have a **main()** function, which should be the first function to be called
- From this function other functions can be called
  - and others from those, and other from those, and so on
  - The **main** naming convention is standard practise across several programming languages, not just python

*** *with a few exceptions*

Activity

Use a main function

```
def validate_input(input_type, length):
    '''
    Asks the user for an input and exits if the input is not
    alphabetical, capitalized, and of the correct length.
    input_type: a string label for the input prompt and error message
    length: an int, representing the max length of the input string
    '''
    value = input('Enter ' + input_type + ': ')
    if not value.isalpha() or len(value) > length or not value[0].isupper():
        print('Invalid ' + input_type + '.')
        exit()

validate_input('first name', 15)
validate_input('middle initial', 1)
validate_input('last name', 30)

print('Valid name!')
```

Use a main function

```
def validate_input(input_type, length):
    '''
    Asks the user for an input and exits if the input is not
    alphabetical, capitalized, and of the correct length.
    input_type: a string label for the input prompt and error message
    length: an int, representing the max length of the input string
    '''
    value = input('Enter ' + input_type + ': ')
    if not value.isalpha() or len(value) > length or not value[0].isupper():
        print('Invalid ' + input_type + '.')
        exit()

def main():
    validate_input('first name', 15)
    validate_input('middle initial', 1)
    validate_input('last name', 30)
    print('Valid name!')

main()
```

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(temperature):
    if height > 98:
        return "fever"
    else:
        return "no fever"

statements . . .

category_1 = categorize(100)
category_2 = categorize(95)

statements . . .
```

What would this print?

```
def build_a_string():
    word_1 = 'this'
    word_2 = ' is'
    word_3 = ' awesome'
    words = word_1 + word_2 + (word_3 * 2)
    return words

something = build_a_string()
print(something)
```

What would this print?

```
def repeat(content, times):
    to_return = content * times
    return to_return

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

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)
```

The pythagorean theorem

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

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

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

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)
```

- Remember fractional exponents

$$x^{\frac{1}{n}} = \sqrt[n]{x}$$

Implement the pythagorean function

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

Implement the pythagorean function

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

```
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()
```