CS A131: Lecture 7

Nadia Ahmed

Orange Coast College, Computer Science

CS A131

Lecture 7: Overview

- Functions
  - Terms and concepts
  - Scope rules
  - Scope example
- Storing Functions in Modules

Functions

Review: Terms and Concepts

- Function parameters
  - formal parameters holding the data supplied to a function
- Function Definition
  - defines the behavior in function body
- Local variables
  - variables defined locally in a function body
- Function call
  - expression invoking a function with supplied arguments
- Function arguments
  - arguments passed to a function call (initial values for parameters)
- Return value
  - result computed by a function call

Functions

- *Scope* of an identifier
  - Portion of the program where the identifier can be referenced
  - aka accessibility, visibility
- Scope rules
  - Global variables: *file* scope
    - Declaration outside any function (at global level)
    - Scope in entire source file after declaration
  - Function parameters: *function* scope
    - Declaration in function parameter list
    - Scope limited to this function body (entirely)
  - Local variables: *block* scope
    - Declaration inside a compound statement (i.e. function body)
    - Scope limited to this compound statement block (entirely)

Scope Rules: Example

```python
x = 5
y = 7

def square (a):
  s = a*a
  return s

def add_y(x):
  s = x + y
  return s

def main():
  z = square(x)
  z = add_y(z)
  print( "z =  %d " % z)

main()
```

- Global variables
- Function definition
  - Local variable

- Function definition
  - Local variable

- Function definition
  - Local variable
- Main function call

Scope Rules: Example

```
1  x = 5
2  y = 7
3
4  def square(a):
5    s = a*a
6    return s
7
8  def add_y(x):
9    s = x + y
10   return s
11
12 def main():
13   z = square(x)
14   z = add_y(z)
15   print("z = %d" % z)
16
17 main()
```

Scope of global variable x: lines 2 through 17

Scope Rules: Example

```
1  x = 5
2  y = 7
3
4  def square(a):
5    s = a*a
6    return s
7
8  def add_y(x):
9    s = x + y
10   return s
11
12 def main():
13   z = square(x)
14   z = add_y(z)
15   print("z = %d" % z)
16
17 main()
```

Scope of global variable y: lines 3 through 17

Scope Rules: Example

```
1   x = 5
2   y = 7
3
4   def square(a):
5     s = a*a
6     return s
7
8   def add_y(x):
9     s = x + y
10    return s
11
12  def main():
13    z = square(x)
14    z = add_y(z)
15    print("z = %d" % z)
16
17  main()
```

Scope of parameter a: lines 5 through 6

Scope Rules: Example

```
 1  x = 5
 2  y = 7
 3
 4  def square(a):
 5    s = a*a
 6    return s
 7
 8  def add_y(x):
 9    s = x + y
10    return s
11
12  def main():
13    z = square(x)
14    z = add_y(z)
15    print("z = %d" % z)
16
17  main()
```

Local variables are independent!
(unless their scopes are nested)

- Scope of local variable `s`: lines 5 through 6
- Scope of local variable `s`: lines 9 through 10
- Scope of local variable `z`: lines 13 through 15

Scope Rules: Example

```
1  x = 5
2  y = 7
3
4  def square(a):
5    s = a*a
6    return s
7
8  def add_y(x):
9    s = x + y
10   return s
11
12 def main():
13   z = square(x)
14   z = add_y(z)
15   print("z = %d" % z)
16
17 main()
```

Scope of parameter `x`: lines 9 through 10

Scope Rules: Example

```python
1  x = 5
2  y = 7
3
4  def square(a):
5    s = a*a
6    return s
7
8  def add_y(x):
9    s = x + y
10   return s
11
12 def main():
13   z = square(x)
14   z = add_y(z)
15   print("z = %d" % z)
16
17 main()
```

Shadowing! In nested scopes, inner scope takes precedence!

- Scope of global variable `x`: lines 2 through 17
- Scope of parameter `x`: lines 9 through 10

Storing Functions in Modules

- A module is a file that contains Python code.
- Divide large programs into modules.
- Usually a module will hold your function definitions
- Helpful if you have written a set of functions that are needed in several different programs
- Import the module in each program that needs to call one of the functions.
  - module's file name should end in `.py`
  - If the module's file name does not end in `.py` you will not be able to import it to other programs
  - A modules' name cannot be a python keyword
  - Must use the keyword `import` *module_name* to use the function in the module

Modules: `circle.py`

```python
# The circle module has functions that perform
# calculations related to circles.
import math

# The area function accepts a circle's radius as an
# argument and returns the area of the circle.
def area(radius):
  return math.pi * radius**2

# The circumference function accepts a circle's
# radius and returns the circle's circumference.
def circumference(radius):
  return 2 * math.pi * radius
```

Modules: `rectangle.py`

```python
# The rectangle module has functions that perform
# calculations related to rectangles.

# The area function accepts a rectangle's width and
# length as arguments and returns the rectangle's area.
def area(width, length):
  return width * length

# The perimeter function accepts a rectangle's width
# and length as arguments and returns the rectangle's
# perimeter.
def perimeter(width, length):
  return 2 * (width + length)
```

Modules: geometry.py 1/2

```python
import circle
import rectangle

# Constants for the menu choices
AREA_CIRCLE_CHOICE = 1
CIRCUMFERENCE_CHOICE = 2
AREA_RECTANGLE_CHOICE = 3
PERIMETER_RECTANGLE_CHOICE=4
QUIT_CHOICE = 5

def main():
  choice = 0
  while choice != QUIT_CHOICE:
    display_menu()
    choice = int(input('Enter your choice: '))
    # Perform the selected action.
    if choice == AREA_CIRCLE_CHOICE:
      radius = float(input("Enter the circle's radius: "))
      print('The area is', circle.area(radius))
    elif choice == CIRCUMFERENCE_CHOICE:
      radius = float(input("Enter the circle's radius: "))
      print('The circumference is %f' % circle.circumference(
          radius))
    elif choice == AREA_RECTANGLE_CHOICE:
```

Modules: `geometry.py` 2/2

```
...
      width = float(input("Enter the rectangle's width: "))
      length = float(input("Enter the rectangle's length: "))
      print('The area is %f' % rectangle.area(width, length))
    elif choice == PERIMETER_RECTANGLE_CHOICE:
      width = float(input("Enter the rectangle's width: "))
      length = float(input("Enter the rectangle's length: "))
      print('The perimeter is %f' % rectangle.perimeter(width,
          length))
    elif choice == QUIT_CHOICE:
      print('Exiting the program... ')
    else:
      print('Error: invalid selection.')

def display_menu():
    print(' MENU')
    print('1) Area of a circle')
    print('2) Circumference of a circle')
    print('3) Area of a rectangle')
    print('4) Perimeter of a rectangle')
    print('5) Quit')

main()
```