

CS A131: Lecture 7

Nadia Ahmed

Orange Coast College, Computer Science

CS A131



Lecture 7: Overview

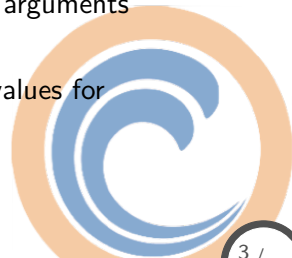
- Functions
 - Terms and concepts
 - Scope rules
 - Scope example
- Storing Functions in Modules



Functions

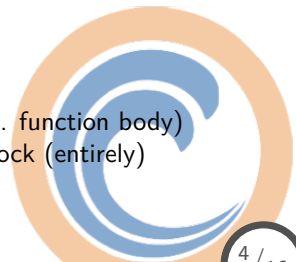
Review: Terms and Concepts

- 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

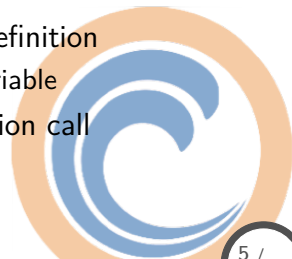
- *Scope* of an identifier
 - Portion of the program where the identifier can be referenced
 - aka accessibility, visibility
- Scope rules
 - Global variables: *file* scope
 - Declaration outside any function (at global level)
 - Scope in entire source file after declaration
 - Function parameters: *function* 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

```
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()
```

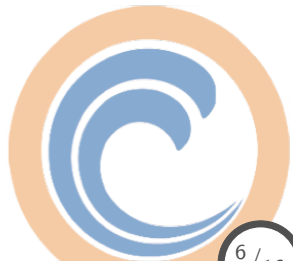
- Global variables
- Function definition
 - Local variable
- Function definition
 - Local variable
- Function definition
 - Local variable
- Main function call



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()
```

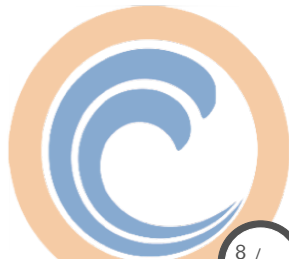
Scope of global variable y: lines 3 through 17



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

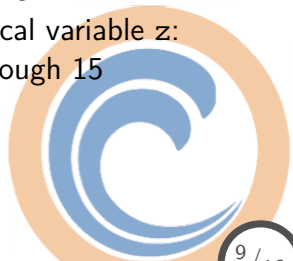


Scope Rules: Example

```
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)

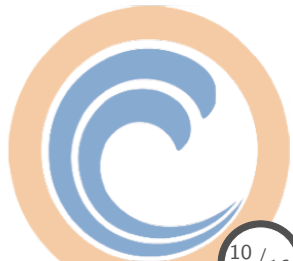
- Scope of local variable `s`:
lines 5 through 6
- Scope of local variable `s`:
lines 9 through 10
- Scope of local variable `z`:
lines 13 through 15



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

```
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()
```

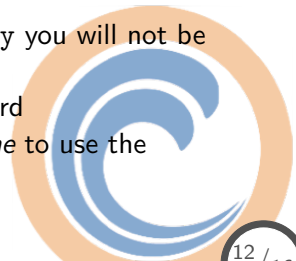
Shadowing! In nested scopes, inner scope takes precedence!

- 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

```
1  # The circle module has functions that perform  
2  # calculations related to circles.  
3  import math  
4  
5  # The area function accepts a circle's radius as an  
6  # argument and returns the area of the circle.  
7  def area(radius):  
8      return math.pi * radius**2  
9  
10 # The circumference function accepts a circle's  
11 # radius and returns the circle's circumference.  
12 def circumference(radius):  
13     return 2 * math.pi * radius
```

Modules: rectangle.py

```
1  # The rectangle module has functions that perform  
2  # calculations related to rectangles.  
3  
4  # The area function accepts a rectangle's width and  
5  # length as arguments and returns the rectangle's area.  
6  def area(width, length):  
7      return width * length  
8  
9  # The perimeter function accepts a rectangle's width  
10 # and length as arguments and returns the rectangle's  
11 # perimeter.  
12 def perimeter(width, length):  
13     return 2 * (width + length)
```

Modules: geometry.py 1/2

```
1 import circle
2 import rectangle
3
4 # Constants for the menu choices
5 AREA_CIRCLE_CHOICE = 1
6 CIRCUMFERENCE_CHOICE = 2
7 AREA_RECTANGLE_CHOICE = 3
8 PERIMETER_RECTANGLE_CHOICE=4
9 QUIT_CHOICE = 5
10
11 def main():
12     choice = 0
13     while choice != QUIT_CHOICE:
14         display_menu()
15         choice = int(input('Enter your choice: '))
16         # Perform the selected action.
17         if choice == AREA_CIRCLE_CHOICE:
18             radius = float(input("Enter the circle's radius: "))
19             print('The area is', circle.area(radius))
20         elif choice == CIRCUMFERENCE_CHOICE:
21             radius = float(input("Enter the circle's radius: "))
22             print('The circumference is %f' % circle.circumference(
                radius))
23         elif choice == AREA_RECTANGLE_CHOICE:
```

Modules: geometry.py 2/2

```
1  ...
2      width = float(input("Enter the rectangle's width: "))
3      length = float(input("Enter the rectangle's length: "))
4      print('The area is %f' % rectangle.area(width, length))
5  elif choice == PERIMETER_RECTANGLE_CHOICE:
6      width = float(input("Enter the rectangle's width: "))
7      length = float(input("Enter the rectangle's length: "))
8      print('The perimeter is %f' % rectangle.perimeter(width,
9      length))
10     elif choice == QUIT_CHOICE:
11         print('Exiting the program... ')
12     else:
13         print('Error: invalid selection.')
14
15 def display_menu():
16     print(' MENU')
17     print('1) Area of a circle')
18     print('2) Circumference of a circle')
19     print('3) Area of a rectangle')
20     print('4) Perimeter of a rectangle')
21     print('5) Quit')
22
23 main()
```