CS A131: Lecture 6

Nadia Ahmed

Orange Coast College, Computer Science

CS A131

Overview

- Functions
  - Introduction to function concepts
    - Function definition
    - Function call
- Simple functions
  - Example: `squareEx.py`
- Hierarchy of functions
  - Example `cylinder.py`

Introduction to Functions

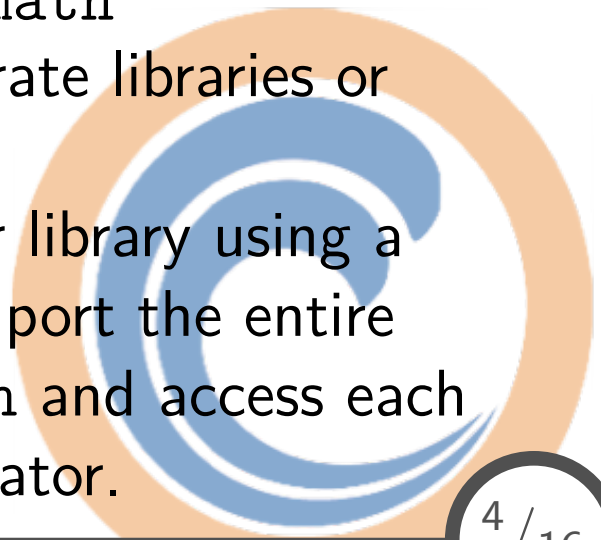
- Functions are often called *modules*
- They are like miniature programs that can be combined to form larger programs.
- They allow complicated programs to be divided into manageable task portions.

Predefined Functions

- `from math import` *function_name*
- `import math`
- Examples

math library	
`abs(x)` or `math.abs(x)`	`math`
`ceil(x)` or `math.ceil(x)`	`math`
`cos(x)` or `math.cos(x)`	`math`
`exp(x)` or `math.exp(x)`	`math`
`floor(x)` or `math.floor(x)`	`math`
`pow(x,y)` or `math.pow(x,y)`	`math`
`sqrt(x)` or `math.sqrt(x)`	`math`

- Predefined functions are organized into separate libraries or toolkits
- You can import the function from the proper library using a `from math import sqrt` statement. To import the entire library you can alternatively state `from math` and access each function from that library using the dot operator.



User Defined Functions

Introduction to Functions

- Important programming concepts
  - Hierarchy
  - Encapsulation
  - Information hiding
  - Divide and conquer
- Software reuse
  - Don't reinvent the wheel!
- Program composition
  - python program = Set of functions
    - starting point: function named `main`
  - Libraries = Set of functions
    - predefined functions (often written by somebody else)

Functions

python programming language distinguishes 2 constructs around functions

1. Function definition
   - definition of the function behavior
   - comprised of code statements
2. Function call
   - invocation of a function

Functions

Function definition

- defines the statements executed by the function
- may use local variables for the computation
- returns result value via `return` statement (if any)
- use indentation to delineate function body
- Example:

```
def Square(p):
  r = p * p
  return r
```

Functions

Function call

- Function call
  - expression invoking a function
  - supplies arguments for formal parameters
  - invokes the function
  - result is the value returned by the function
- Example

```
a = 42
b = Square(a)
```

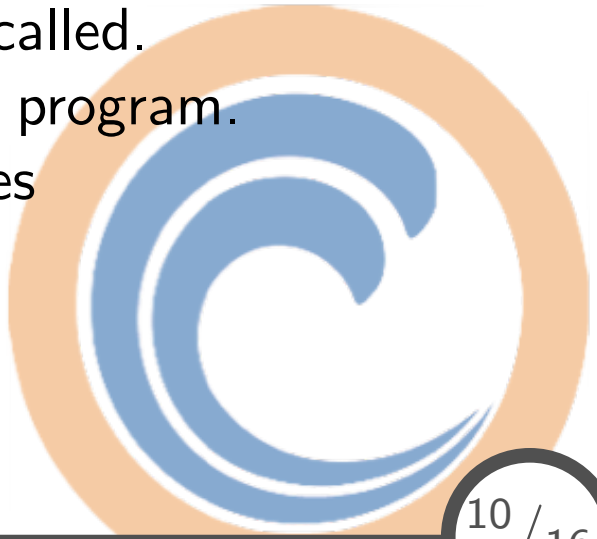
  - function `Square` is called
  - argument `a` is passed for parameter `p` (by value)
  - value returned by the function is assigned to `b`

Value returning functions vs. void functions

- Value returning functions: have a return type
  - Return a value of a specific data type using the `return` statement
- Void functions: do not a have a return type
  - Do *not* use a `return` statement to return a value

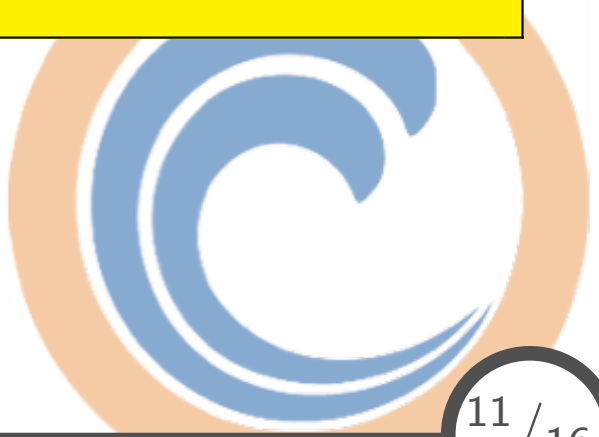
Functions

- python programming language distinguishes 2 constructs
  - Function definition
    - a function declaration with a function body
    - definition of the function behavior
  - Function call
    - invocation of a function
- python program rules
  - A function must be defined before it can be called.
  - A function must be defined exactly once in a program.
  - A function may be called any number of times



Program example: squareEx.py

```
##################################################
# squareEx.py: example demonstrating functions
# author: Nadia Ahmed
# modifications:
# 07/23/16  NA initial version
##################################################

#function definition
def square(p):
    r = p * p
    return r
#end of square
```



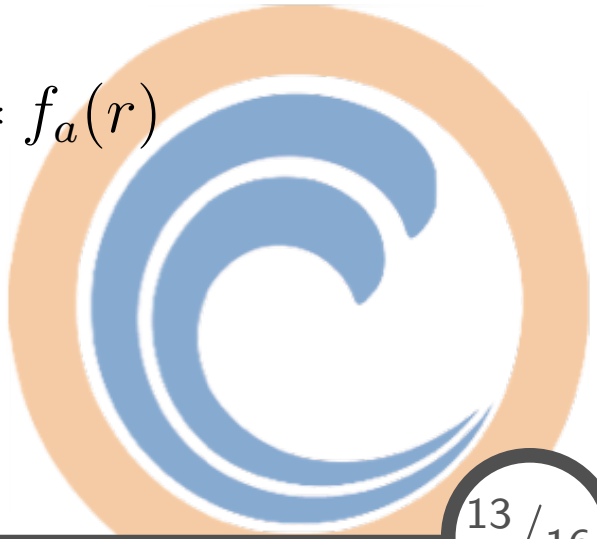
Program example: squareEx.py

```
#main function
def main():
  #input section
  a = int(input("Please enter a value for the argument: "))

  #computation section
  b = square(a)

  #output section
  print("The square of %d is %d" %(a,b))
#end of main

main()
```

Functions

- Hierarchy of Functions
  - functions call other functions
- Example: Cylinder calculations
  - given radius and height
  - calculate surface and volume

Circle constant	$\pi = 3.14159265...$
Circle perimeter	$f_p(r) = 2 * \pi * r$
Circle area	$f_a(r) = \pi * r^2$
Cylinder surface	$f_s(r, h) = f_p(r) * h + 2 * f_a(r)$
Cylinder volume	$f_v(r, h) = f_a(r) * h$



Program example: cylinderEx.py

```
##########################################
# cylinderEx.py: cylinder functions
# author: Nadia Ahmed
# modifications:
# 07/23/16  NA initial version
##########################################

#cylinder functions
def pi_():

  return 3.1415927


def CircleArea(r):

  return pi_()*r*r

...
```

Program example: cylinderEx.py

```
...
def CirclePerimeter(r):

  return 2*pi_()*r


def Surface(r,h):

  side = CirclePerimeter(r) *h
  lid = CircleArea(r)

  return side+2*lid


def Volume(r,h)

  return CircleArea(r)*h
```

Program example: cylinderEx.py

```
...
# main function
def main ():

  #input section
  r = float(input("Please enter the radius: "))
  h = float(input("Please enter the height: "))

  #computation section
  s = Surface(r, h)
  v = Volume(r, h)

  #output section
  print("The surface area is %f" % s)
  print("The volume is %f " % v)
#end of main

main()
```