An Overview of Python Programming

Lesson 2: Functions and Conditionals

Nick Featherstone
feathern@colorado.edu

Useful Resources

- Github for this class:
  - https://github.com/ResearchComputing/Python_Overview_Fall2017
- Free online text:
  - How to Think Like a Computer Scientist (HTLCS)
  - http://openbookproject.net/thinkcs/python/english3e/
- Official Python Website:
  - www.python.org
- Paperback Textbook:
  - Python Programming: An Introduction to Computer Science
  - 2nd ed., J. Zelle
- Be sure and see “formatting_numbers.py” in today’s folder

Outline

- Functions
- Conditionals
- Recursion

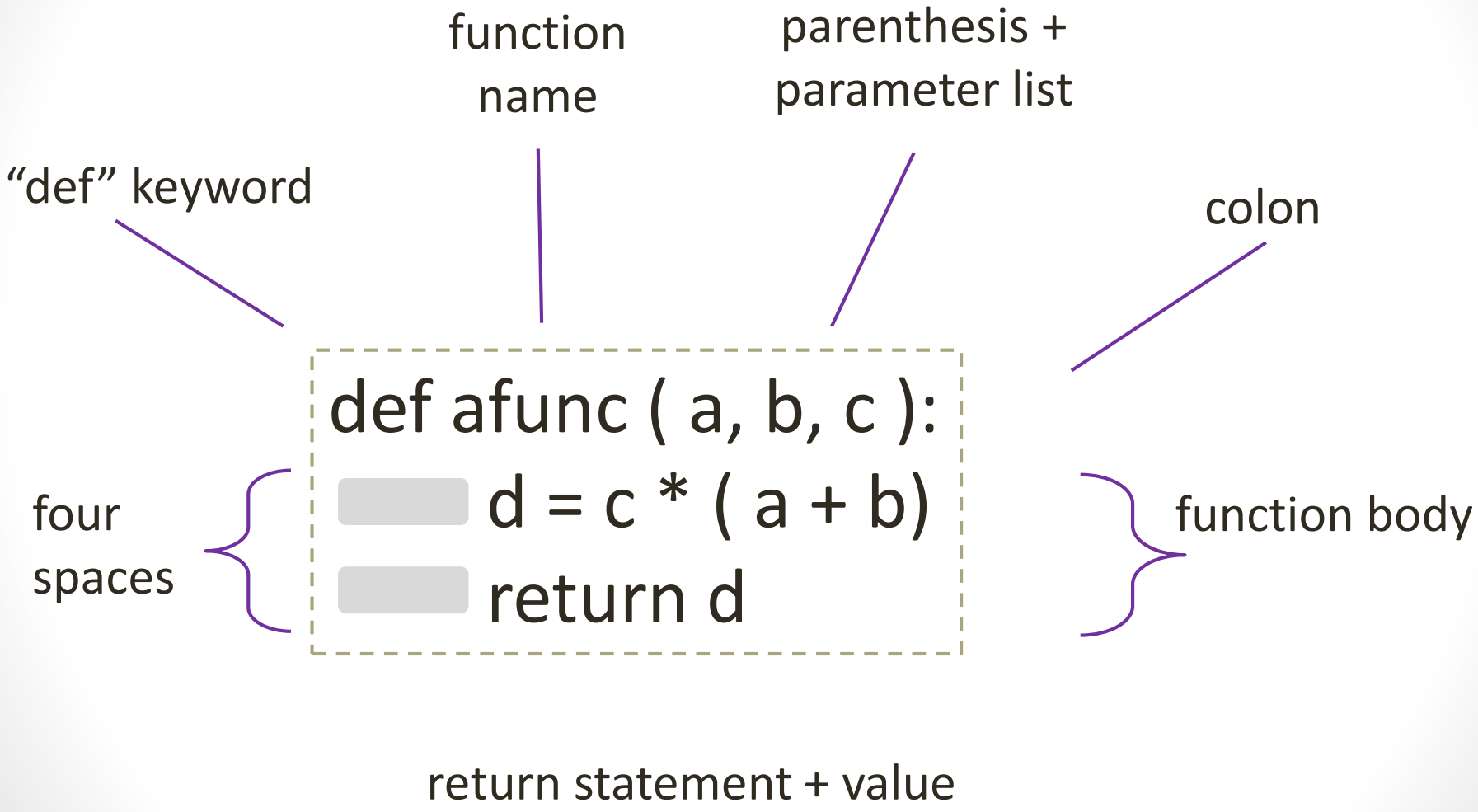
Function Definitions in Python

- Functions must be defined before they can be called
- Definition syntax:

```
def afunc ( a, b, c = 1):
    d = c * ( a + b)
    return d
```

- Several pieces to this…

Function Definitions in Python



Calling Functions

```
def afunc ( a, b, c ):
    d = c * ( a + b)
    return d

myval = afunc(1,2,4)
```

- Functions may be called once defined
- Value of “d” assigned to “myval” via return statement

Exercise 1

- Write a function that accepts two parameters and returns the difference between those two parameters.
- In your main program, call this function with various parameter combinations

Exercise 2

- Write a function that accepts two parameters:
  - name : a string value
  - age : an ‘int’ value
- It should return a single string value:
  - msg : a string with value “{name} is {age} years old.”
- Hint: use the “str” type conversion function

Exercise 2

- Let's look at exercise2_solution.py

Calling Functions

```
def afunc ( a, b, c = 1):
    d = c * ( a + b)
    e = c * ( a – b)
    return d,e


myval1, myval2 = afunc(1,2)
```

- Multiple scalar values may be returned.
- Separate values with commas
- d → myval1 e → myval2

Calling Functions

```
def afunc ( a ):
    print ( a)


afunc(2)
g = afunc(2)
print(g)
```

Open and run “nonetype.py”

- Functions need not return a value
- Even if no “return” statement, functions will return Nonetype
- Nonetype: empty datatype; prints as “None”

Optional Parameters

```
def afunc ( a, b, c = 1):
    d = c * ( a + b)
    return d

myval = afunc(1,2)
myval2 = afunc(1,2,c=2)
```

- Optional parameters indicated by setting a default value
- c does not have to be passed to afunc
- Defaults to value of 1

Optional Parameters

```
def afunc ( a, b = 1, c = 1):
    d = c * ( a + b)
    return d
```

- Optional parameters can be specified implicitly by position (no "=" needed)

afunc(3,b=2)	afunc(3,2,c=1)	afunc(3,2,1)
afunc(3,2)	afunc(3,b=2,c=1)	*equivalent function calls*

Pass by Value or Reference?

- General rule of thumb
  - Scalar variables behave as though passed by value
  - Most everything else behaves as though passed by reference
- Open and run pass_by_reference_or_value.py

Scope

- Scope behaves “as expected” in python.
- Variables defined within a function are invisible to the program unit that called the function.
- If a variable is undefined, and its value is accessed, Python will check the program unit where the function was defined and use its value (if it exists)
- Open and run “scope.py”

Logic: logical operators

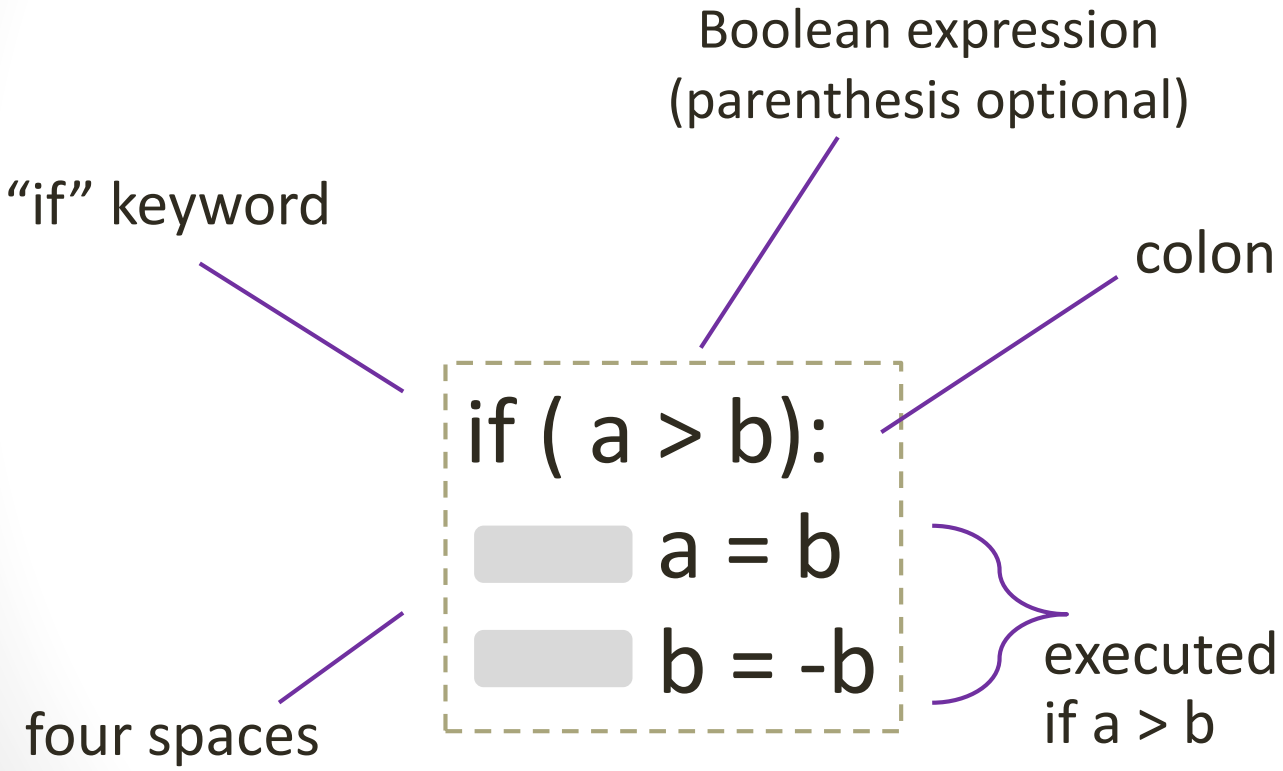
- Boolean expressions have value True or False in Python
- Boolean values can be combined to yield a Boolean expression via logical operators:
  - and
  - or
  - not
- Open and run logical_operators.py

Logic: comparison operators

- Boolean expressions have value True or False in Python
- Numeric values can be combined to yield a Boolean expression via comparison operators:
  - == “equals”
  - != “not equal”
  - > “ greater than”
  - >= “greater than or equal to”
  - < “less than”
  - <= “less than or equal to”
- Open and run comparison_operators.py
- Note that “==“ and “!=“ work with string values as well

Conditionals in Python 1: if

- Conditional syntax is similar to function definition syntax:



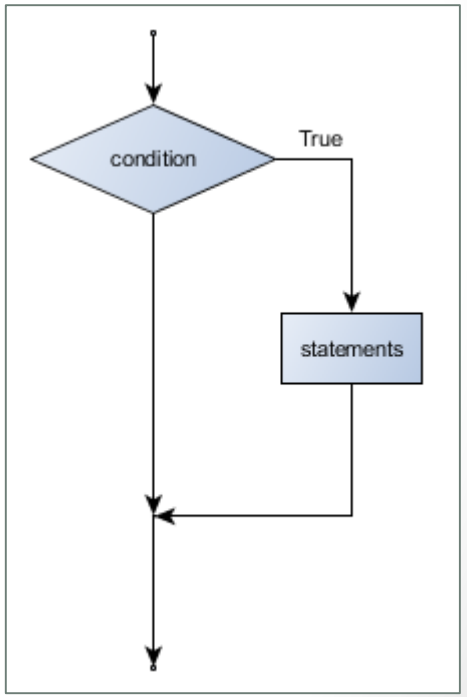


image credit: HTLCS

Exercise 3

- Write a function named ispositive that:
  - Accepts a single, numeric parameter
  - uses if (without else) to return
    - True if the input parameter is positive.
    - False otherwise

```
def ispositive(a):
    if ( expr ):
        statement 1
    statement 2
```

Conditionals in Python 2: if / else

- Can add an “else” clause to our if statement

```
if ( a > b):
    a = b
    b = -b
else:
    b = a
    a = -a
```

executed if a > b

executed if a <= b

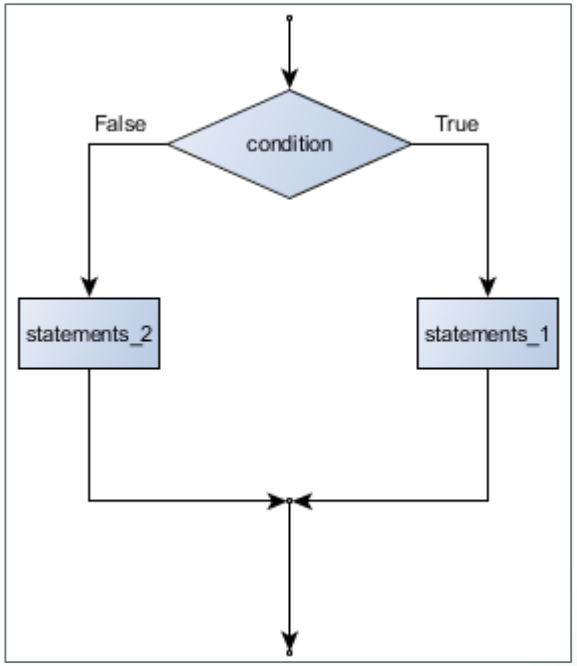


image credit: HTLCS

Conditionals in Python 3: elif

- Can also add an else-if clause(s) via “elif”

```
if ( x  <  y):
    a = -1
elif (x > y):
    a = 1
else:
    a = 0
```

executed if x < y

executed if x > y

executed if x = y

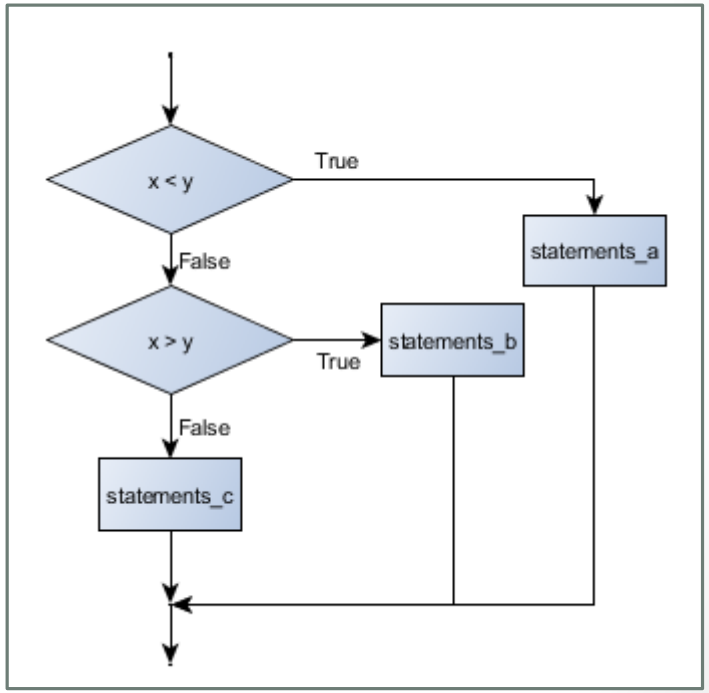


image credit: HTLCS

Exercise 4

- Using if/elif/else write a function that takes a number between 0 and 100 and returns the associated letter grade.
- e.g, grade(75) will return ‘C’

Recursion in Python

- Python allows the user to define recursive functions.
- No extra keywords needed.
- The function is recursive by virtue of calling itself:

```
def afunc ( parameters):
    if ( expr):
        statements 1
        return something
    else:
        statements 2
        afunc(new parameters)
```

Exercise 5 : Recursive Gynmastics

- Write a recursive function that:
  - Accepts a single integer parameter “n”
  - Prints all *odd* numbers from 1 through n
- Write a second function that prints all even numbers 2 through n, but in reverse (i.e., n, n-2, n-4, …, 2)
- Write a recursive function that computes n!

Next Time

- Lists, tuples, and dictionaries
- Iteration