Imperative Programming

Kenn H. Kim, Ph. D.

School of Business
Clemson University

1

Topics in This Week

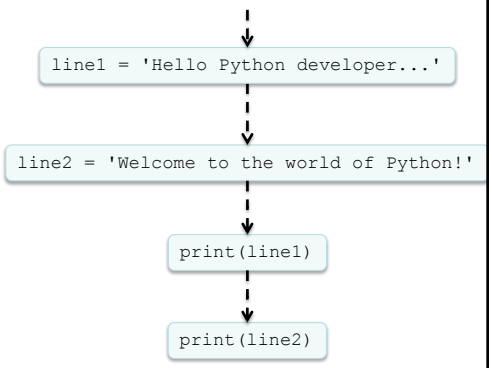
- Python Programs
- Interactive Input/Output
- One-Way and Two-Way `if` Statements
- `for` Loops
- User-Defined Functions

2

Python Program

A Python program is a sequence of Python statements

- Stored in a text file (e.g. hello.py) called a Python module
- Executed using an IDE or "from the command line"



hello.py

```
line1 = 'Hello Python developer...'
line2 = 'Welcome to the world of Python!'
print(line1)
print(line2)
```

```
$ python hello.py
Hello Python developer…
Welcome to the world of Python!
```

3

Built-in Function `print()`

Function `print()` prints its input argument

- The argument can be any object: an integer, a float, a string, a list, …
  - Strings are printed without quotes and "to be read by people", rather than "to be interpreted by Python",
- The "string representation" of the object is printed

```
>>> print(0)
0
>>> print(0.0)
0.0
>>> print('zero')
zero
>>> print([0, 1, 'two'])
[0, 1, 'two']
```

4

Built-in Function `input()`

Function `input()` requests and reads input from the user interactively

- It's (optional) input argument is the request message
- Typically used on the right side of an assignment statement

When executed:

1. The input request message is printed
2. The user enters the input
3. The *string* typed by the user is assigned to the variable on the left side of the assignment statement

```
first = input('Enter first name: ')
last = input('Enter last name: ')
line1 = 'Hello' + first + '' + last + '…'
print(line1)
print('Welcome to the world of Python!')
```

```
>>> name = input('Enter your name: ')
Enter your name: Michael
>>> name
'Michael'
>>> ========= RESTART =============
>>>
Enter your first name: Michael
Enter your last name: Lee
'Hello Michael Lee'
'Welcome to the world of Python!'
```

5

Built-in Function `input()`

Function `input()` evaluates anything the user enters as a **string**

What if we want the user to interactively enter non-string input such as a number?

- Solution: Use int()

```
>>> age = input('Enter your age: ')
Enter your age: 18
>>> age
'18'
>>> int(age)
18
```

6

Exercise (Notebook)

CLEMSON College of BUSINESS

Write a program that:

1. Requests the user's name
2. Requests the user's age
3. Computes the user's age one year from now and prints the message shown

```
>>>
Enter your name: Marie
Enter your age: 17
Marie, you will be 18 next year!
```

7

Execution Control Structures

CLEMSON College of BUSINESS

- Execution control structures are programming language statements that control which statements are executed, i.e., the execution flow of the program
- The if statements (e.g. one-way, two-way, etc.) are, more specifically, conditional structures

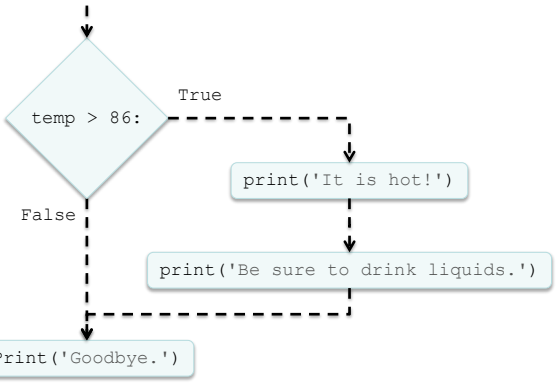
8

One-Way `if` Statement

```
if <condition>:
    <indented code block>
<non-indented statement>
```

```
if temp > 86:
    print('It is hot!')
    print('Be sure to drink liquids.')
print('Goodbye.')
```

"The value of `temp` is 50."

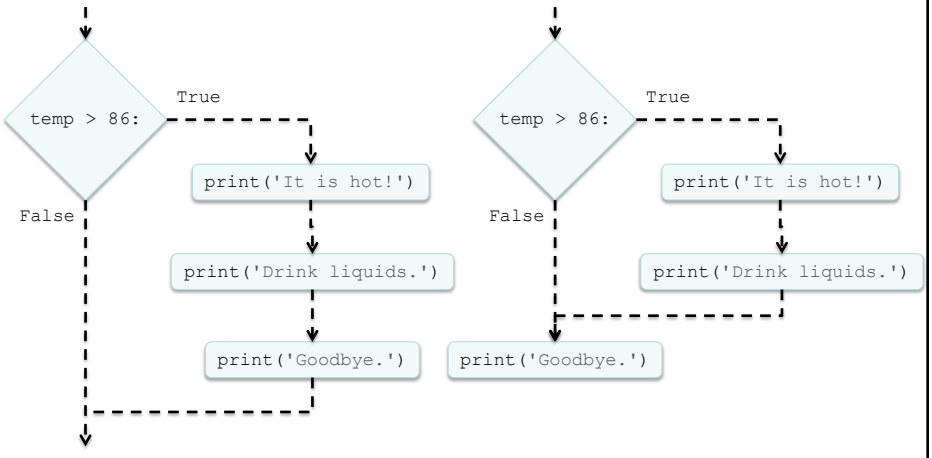


9

Indentation is Critical!

```
if temp > 86:
    print('It is hot!')
    print('Drink liquids.')
    print('Goodbye.')
```

```
if temp > 86:
    print('It is hot!')
    print('Drink liquids.')
print('Goodbye.')
```



10

Exercises (Notebook)

Write corresponding if statements:

a) If `age` is greater than 62 then print `'You can get Social Security benefits'`

b) If string `'large bonuses'` appears in string `report` then print `'Vacation time!'`

c) If `hits` is greater than 10 and `shield` is 0 then print `"You're dead..."`

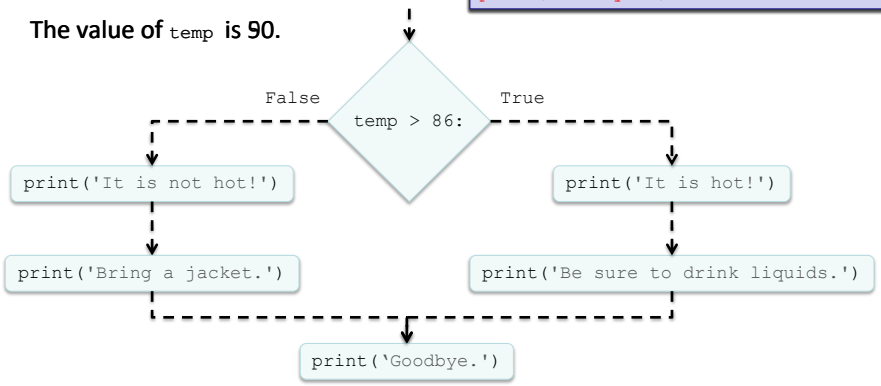
11

Two-Way `if` Statement

```
if <condition>:
    <indented code block 1>
else:
    <indented code block 2>
<non-indented statement>
```

```
if temp > 86:
    print('It is hot!')
    print('Be sure to drink liquids.')
else:
    print('It is not hot.')
    print('Bring a jacket.')
print('Goodbye.')
```

The value of `temp` is 90.



12

Exercise (Notebook)

Write a program that:

1) Requests the user's name
2) Requests the user's age
3) Prints a message saying whether the user is eligible to vote or not

```
>>>
Enter your name: Marie
Enter your age: 17
Marie, you can't vote.
>>>
============RESTART================
>>>
Enter your name: Marie
Enter your age: 18
Marie, you can vote.
>>>
```

13

Execution Control Structures

- The one-way and two-way if statements are examples of execution control structures, esp. conditional structures.
- Iteration structures are execution control structures that enable the repetitive execution of a statement or a block of statements
- The for loop statement is an iteration structure that executes a block of code for every item of a sequence

14

For Loop

Executes a block of code for every item of a sequence

- If sequence is a string, items are its characters (single-character strings)

```
name  =  'A  p  p  l  e'

char  =  'A'
char  =     'p'
char  =        'p'
char  =           'l'
char  =              'e'
```

```
>>> name = 'Apple'
>>> for char in name:
        print(char)


A
p
p
l
```

15

For Loop

Executes a code block for every item of a sequence

- Sequence can be a string, a list, …
- Block of code must be indented

```
for <variable> in <sequence>:
    <indented code block >
<non-indented code block>
```

```
for word in ['stop', 'desktop', 'post', 'top']:
    if 'top' in word:
        print(word)
print('Done.')
```

```
word  =  'stop'
word  =     'desktop'
word  =        'post'
word  =           'top'
```

```
>>>
stop
desktop
top
Done.
```

16

Exercise (Notebook)

Write a "spelling" program that:

1) Requests a word from the user
2) Prints the characters in the word from left to right, one per line

```
===========RESTART===============
>>>
Enter a word: omnipotent
The word spelled out:
o
m
n
i
p
o
t
e
n
t
>>>
```

17

Built-in Function `range()`

Function range() is used to iterate over a sequence of numbers in a specified range

- To iterate over the n numbers 0, 1, 2, …, n-1
  ```
  for i in range(n):
  ```
- To iterate over the range j, j+1, j+2, …, n-1
  ```
  for i in range(j, n):
  ```
- To iterate over the range with step c: j, j+c, j+2c, j+3c, …, n-1
  ```
  for i in range(j, n, c):
  ```

```
for i in range(j, n, 1):
for i in range(j, n):
```

```
>>> for i in range(2, 16, 10):
        print(i)


2
12
>>>
```

18

Exercise (Notebook)

Write for loops that will print the following sequences:

a) 0, 1, 2, 3, 4, 5, 6, 7, 8 , 9, 10
b) 1, 2, 3, 4, 5, 6, 7, 8, 9
c) 0, 2, 4, 6, 8
d) 1, 3, 5, 7, 9
e) 20, 30, 40, 50, 60

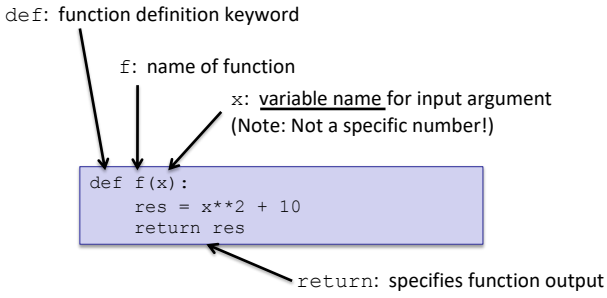
19

Defining New Functions

A few built-in functions we have seen:

- `abs()`, `max()`, `len()`, `sum()`, `print()`

New functions can be defined using `def`




```
>>> abs(-9)
9
>>> max(2, 4)
4
>>> lst = [2,3,4,5]
>>> len(lst)
4
>>> sum(lst)
14
>>> print()

>>> def f(x):
        res = x**2 + 10
        return res

>>> f(1)
11
>>> f(3)
19
>>> f(0)
10
```

* Note the difference between function `input()` and function input argument

20

print() versus return

```
def f(x):
    res = x**2 + 10
    return res
```

```
>>> f(2)
14
>>> 2*f(2)
28
```

Function returns value of `res` which can then be used in an expression

```
def f(x):
    res = x**2 + 10
    print(res)
```

```
>>> f(2)
14
>>> 2*f(2)
14
Traceback (most recent call last):
  File "<pyshell#56>", line 1, in
<module>
    2*f(2)
TypeError: unsupported operand
type(s) for *: 'int' and
'NoneType'
```

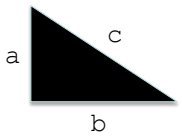
Function prints value of `res` but does not return anything

21

Defining New Functions

The general format of a function definition is

```
def <function name> (<0 or more variables>):
    <indented function body>
```



Let's develop function `hyp()` that:

- Takes two numbers as input (side lengths a and b of above right triangle)
- Returns the length of the hypotenuse c

```
>>> hyp(3,4)
5.0
>>>
```

```
import math
def hyp(a, b):
    res = math.sqrt(a**2 + b**2)
    return res
```

22

Exercise (Notebook)

Write function `hello()` that:

- takes a name (i.e., a string) as input
- prints a personalized welcome message

Note that the function does not return anything

```
>>> hello('Julie')
Welcome, Julie, to the world of Python.
>>>
```

```
def hello(name):
    line = 'Welcome, ' + name + ', to the world of Python.'
    print(line)
```

23

Exercise (Notebook)

Write function `rng()` that:

- takes a list of numbers as input
- returns the range of the numbers in the list

The range is the difference between the largest and smallest number in the list

```
>>> rng([4, 0, 1, -2])
6
>>>
```

```
def rng(lst):
    res = max(lst) - min(lst)
    return res
```

24

Comments and Docstrings

CLEMSON College of BUSINESS

Python programs should be documented

- So the developer who writes/maintains the code understands it
- So the user knows what the program does

Comments (start with #)

```
def f(x):
    res = x**2 + 10    # compute result
    return res         # and return it
```

Docstring (first line after function name)

```
def f(x):
    'returns x**2 + 10'
    res = x**2 + 10    # compute result
    return res         # and return it
```

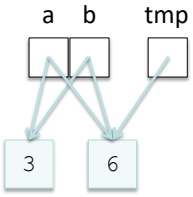
```
>>> help(f)
Help on function f in module
__main__:

f(x)

>>> def f(x):
        'returns x**2 + 10'
        res = x**2 + 10
        return res

>>> help(f)
Help on function f in module
__main__:

f(x)
    returns x**2 + 10

>>>
```

25

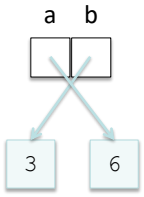
Swapping Values

CLEMSON College of BUSINESS

Have:



Want:




```
>>> a
3
>>> b
6
>>> tmp = b
>>> b = a
>>> a = tmp
```

29

Exercise (Notebook)

Write function `swapFS()` that:

- takes a list as input
- swaps the first and second element of the list, but only if the list has at least two elements

The function does not return anything

```
>>> mylst = ['one', 'two', 'three']
>>> swapFS(mylst)
>>> mylst
['two', 'one', 'three']
>>> mylst = ['one']
>>> swapFS(mylst)
>>> mylst
['one']
>>>
```

```
def swapFS(lst):
    if len(lst) > 1:
        lst[0], lst[1] = lst[1], lst[0]
```

32

We covered Imperative Programming

- Python Programs
- Interactive Input/Output
- One-Way and Two-Way `if` Statements
- `for` Loops
- User-Defined Functions

33