Scientific Programming with Python

Lecture 2: Imperative Programming

UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

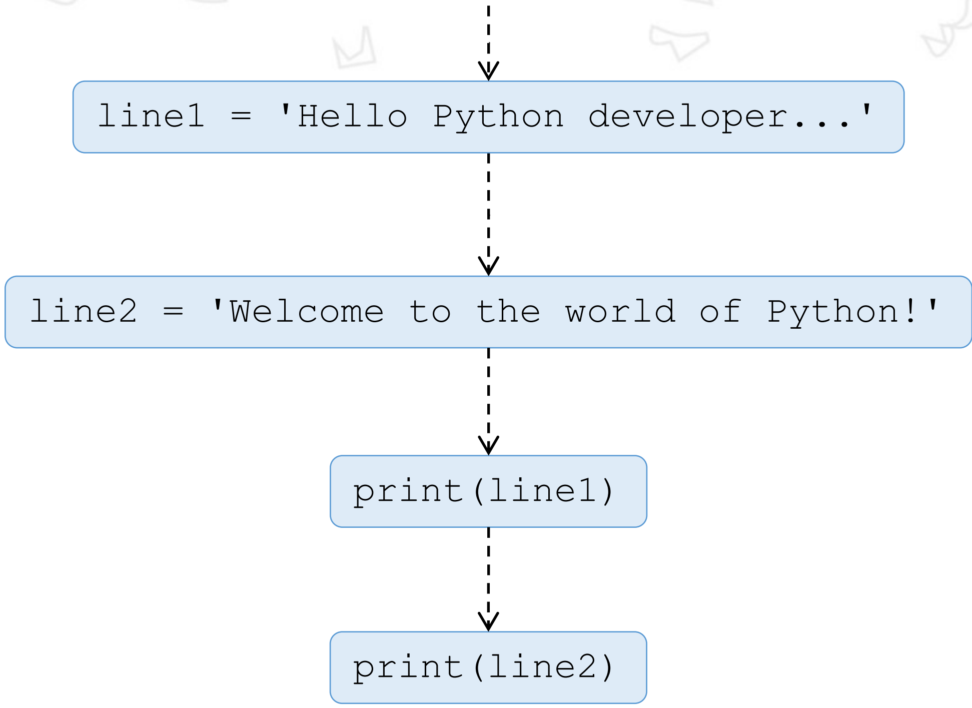
Lecture 2: Imperative Programming

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

Python program

A Python program is a sequence of Python statements

- Stored in a text file called a Python module
- Executed using an IDE or “from the command line”



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

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

hello.py

```
$ python hello.py
Hello Python developer…
```

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

```
>>> name = input('Enter your name: ')
Enter your name: Michael
>>> name
'Michael'
>>> ========= RESTART =============
>>>
Enter your first name:
```

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

input.py

Built-in function `eval()`

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 1: Use type conversion
- Solution 2: Use function `eval()`
  - Takes a string as input and evaluates it as a Python expression

```
>>> age = input('Enter your age: ')
Enter your age: 18
>>> age
'18'
>>> int(age)
18
>>> eval('18')
18
>>> eval('age')
'18'
>>> eval('[2,3+5]')
[2, 8]
>>> eval('x')
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in
<module>
    eval('x')
  File "<string>", line 1, in
<module>
NameError: name 'x' is not defined
>>>
```

Exercise

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!
```

```python
name = input('Enter your name: ')
age = int(input('Enter your age: '))
line = name + ', you will be ' + str(age+1) + ' next year!'
print(line)
```

Exercise

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**

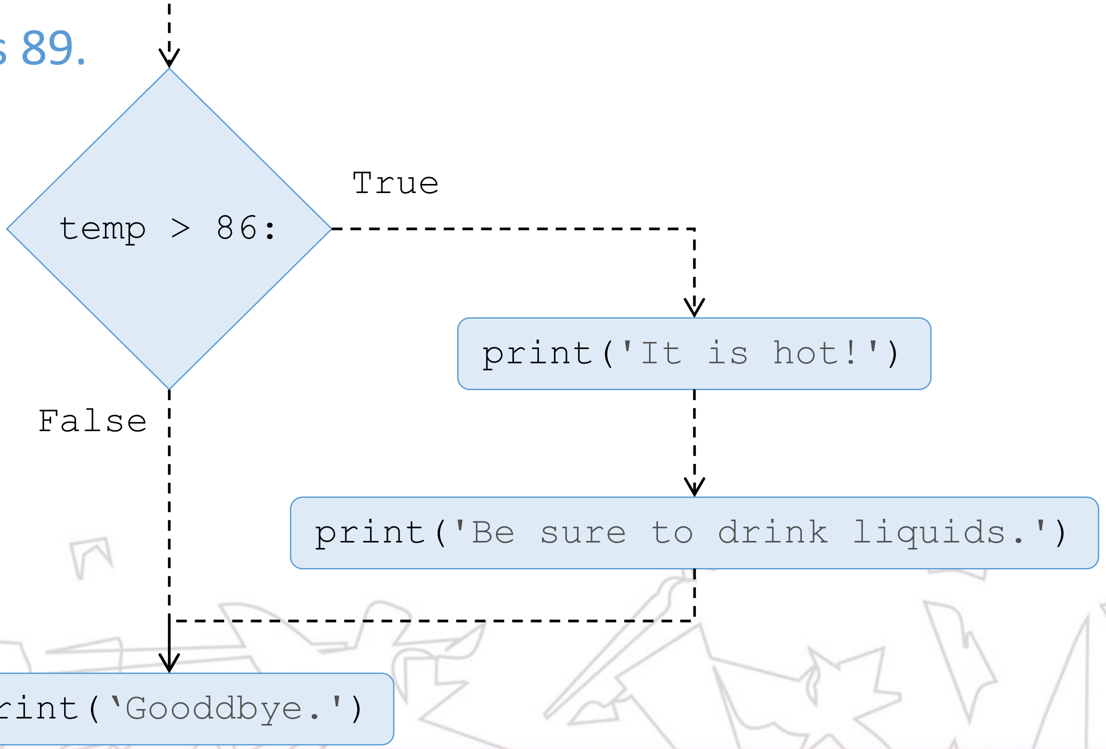
Need a way to execute a Python statement if a condition is true

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.')
```

Assume the value of `temp` is 89.



Examples

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..."`

```
>>> hits = 12
>>> shield = 0
>>> if hits > 10 and shield == 0:
        print("You're dead...")


You're dead...
>>> hits, shield = 12, 2
>>> if hits > 10 and shield == 0:
        print("You're dead...")



>>>
```

```
'



ear'
```

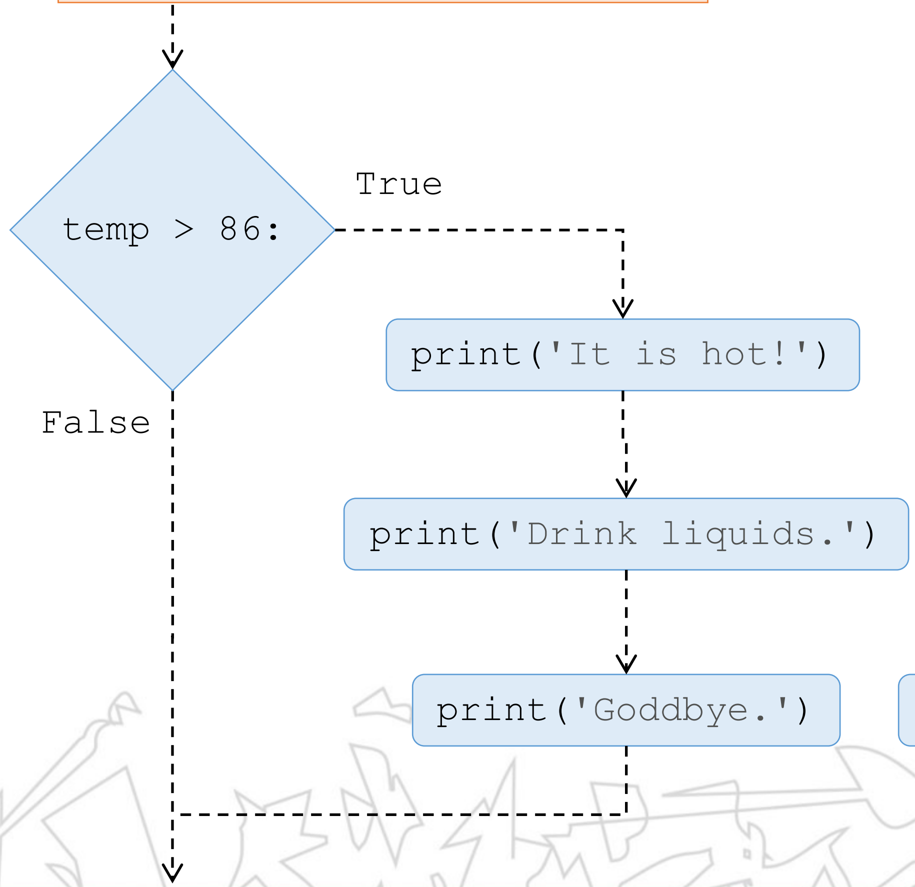
```
ity benefits')




ity benefits')
```

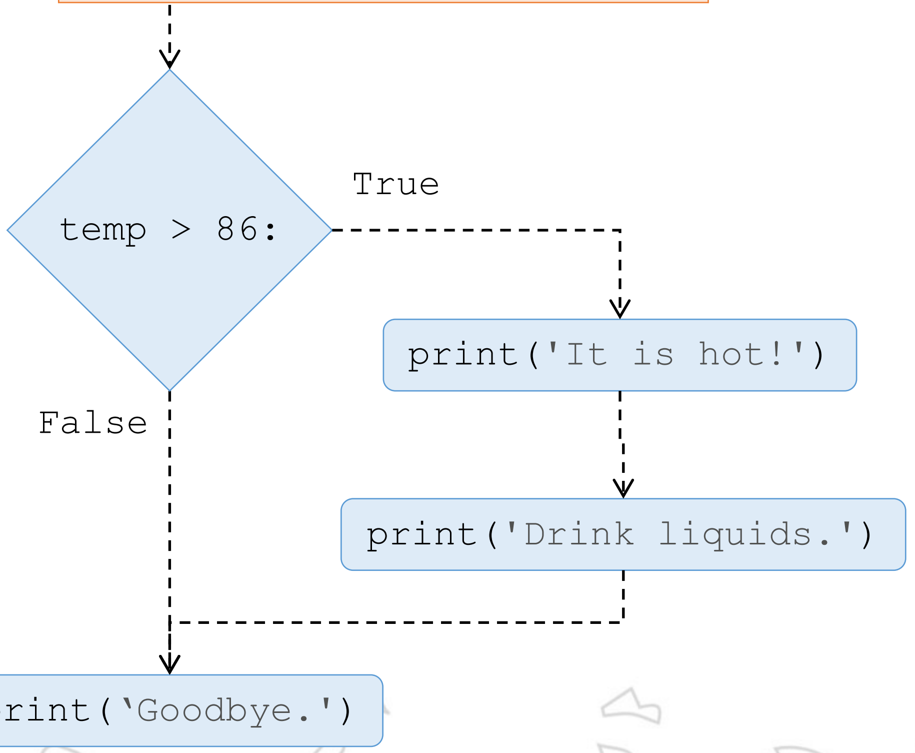
UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

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.')
```



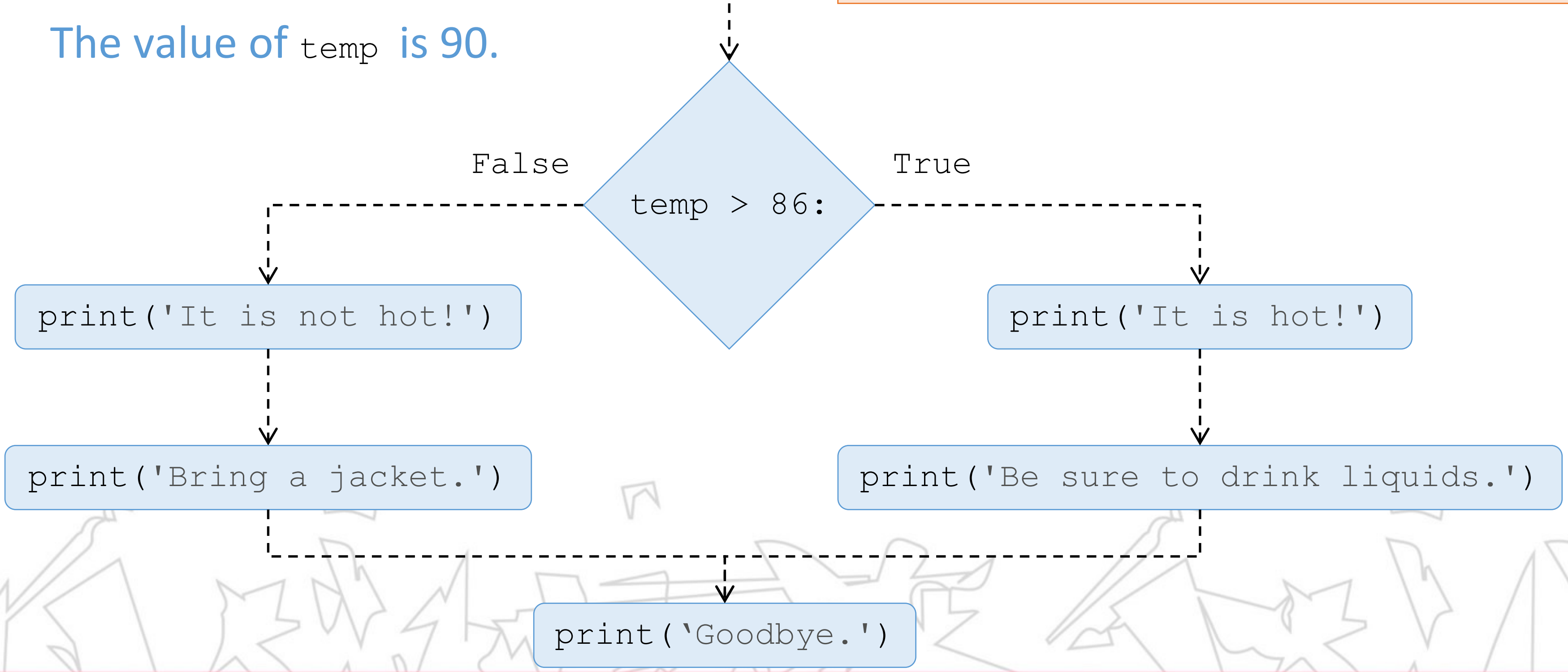


Two-way if statement

```
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.



Multi-way if statement

```
if <condition>:
    <indented code block 1>
elif <condition>:
    <indented code block 2>

. . .

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

```
print(”Letter Grades”)

score =float (input (“enter raw score”))

prefix=‘Your letter grade is’
if score < 60 :
        print(prefix, ‘F’)
elif   60 <= score  < 70:
        print(prefix, ‘D’)
elif   70 <= score  < 80:
        print(prefix, ‘C’)
elif   80 <= score  < 90:
        print(prefix, ‘B’)
else:
        print(prefix, ‘A’)

print('Goodbye.')
```

Exercise

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.
>>>
```

```
name = input('Enter your name: ')
age = eval(input('Enter your age: '))
if age < 18:
    print(name + ", you can't vote.")
else:
    print(name + ", you can vote.")
```

Execution control structures

- The one-way and two-way if statements are examples of execution control structures

- Execution control structures are programming language statements that control which statements are executed, i.e., the execution flow of the program

- The one-way, two-way and multi-way if statements are, more specifically, conditional structures

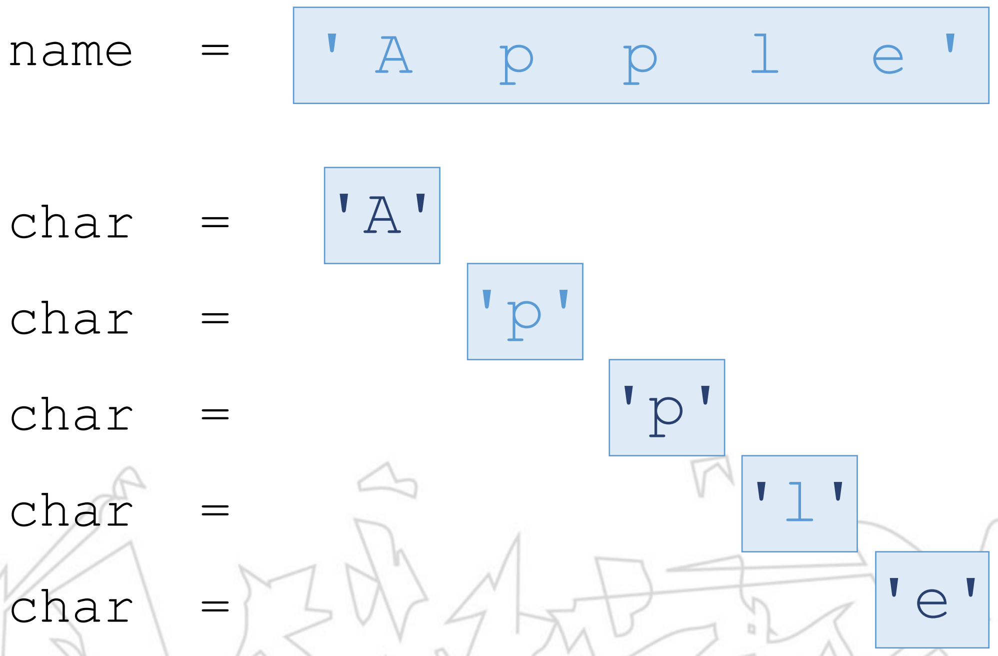
- Iteration structures are execution control structures that enable the repetitive execution of a statement or a block of statements

    - The for loop statement is an iteration structure that executes a block of code for every item of a sequence

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



A
p
p
l
```

`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.
```