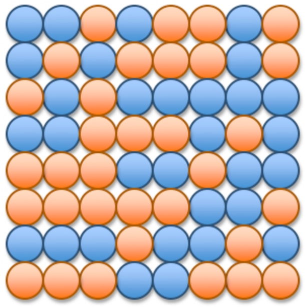


UNIT 3B Conditionals and Introduction to Lists

Data Types

- Booleans
  `True   False`
- Relational Operations
  `==   !=   <   >   <=   >=`
  Example: `while (x < 100):`
- Logical Operations

```
and    or    not
grade >= 80 and grade < 90
grade < 0 or grade > 100
grade >= 80 or grade < 90  →  True
grade < 0 and grade > 100  →  False
```

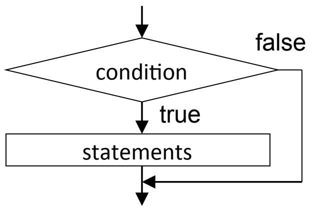
George Boole, 1815-1864

`if` statement

Format:

`if` ***condition*** **`:`**

  statement_list



```
if (x % 2 == 0):
   print("x is even!")
```

`if/else` statement

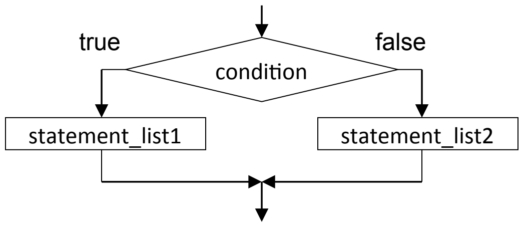
Format:

`if` ***condition*** **`:`**

  statement_list1

`else:`

  statement_list2

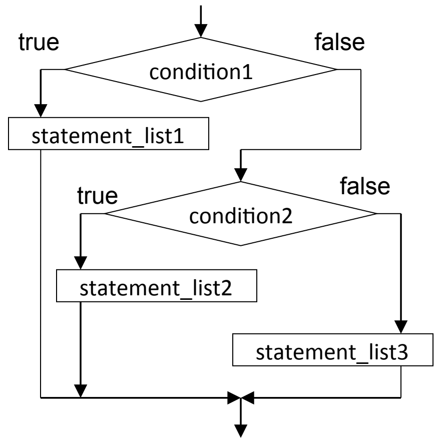


```
if (x % 2 == 0):
      print("x is even!")
else:
      print("x is odd!")
```

Flow chart: `if/elif/else` statement

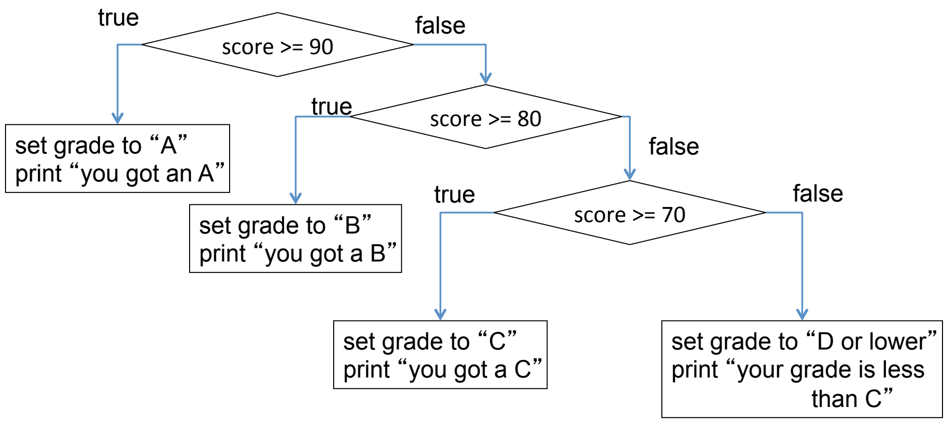
Format:

```
if  condition1 :
    statement_list1
elif  condition2 :
    statement_list2
else:
    statement_list3
```



5

Grader for Letter Grades



6

Nested if statements

```
def grader(score):
    if score >= 90:
        grade = "A"
        print("You got an A")
    else:
        if score >= 80:
            grade = "B"
            print("You got a B")
        else:
            if score >= 70:
                grade = "C"
                print("You got a C")
            else:
                grade = "D or lower"
                print("Your grade is less than C")
    return grade
```

7

Equivalently

```
def grader2(score):
    if score >= 90:
        grade = "A"
        print("You got an A")
    elif score >= 80:
        grade = "B"
        print("You got a B")
    elif score >= 70:
        grade = "C"
        print("You got a C")
    else:
        grade = "D or lower"
        print("Your grade is less than C")
    return grade
```

8

What's wrong?

```
def grader2(score):
    if score >= 90:
        grade = "A"
        print("You got an A")
    if score >= 80:
        grade = "B"
        print("You got a B")
    if score >= 70:
        grade = "C"
        print("You got a C")
    else:
        grade = "D or lower"
        print("Your grade is less than C")
    return grade
```

(DRAW A FLOWCHART.)

Recursive Solution

A recursive algorithm is an algorithm that uses a simpler version of itself as part of its solution.

Input: two non-negative integers x and y
Algorithm:
1. If y is equal to 0, return x as the GCD.
2. Otherwise,
   <u>return the GCD of y and (x modulo y) as the GCD.</u>

Output: the GCD of the initial x and y

Recursive Solution using Python

```
def gcd2(x, y):
    if y == 0:
        return x
    else:
        return gcd2(y, x % y)
```

This is recursive since `gcd2` calls itself.
More about recursion soon.

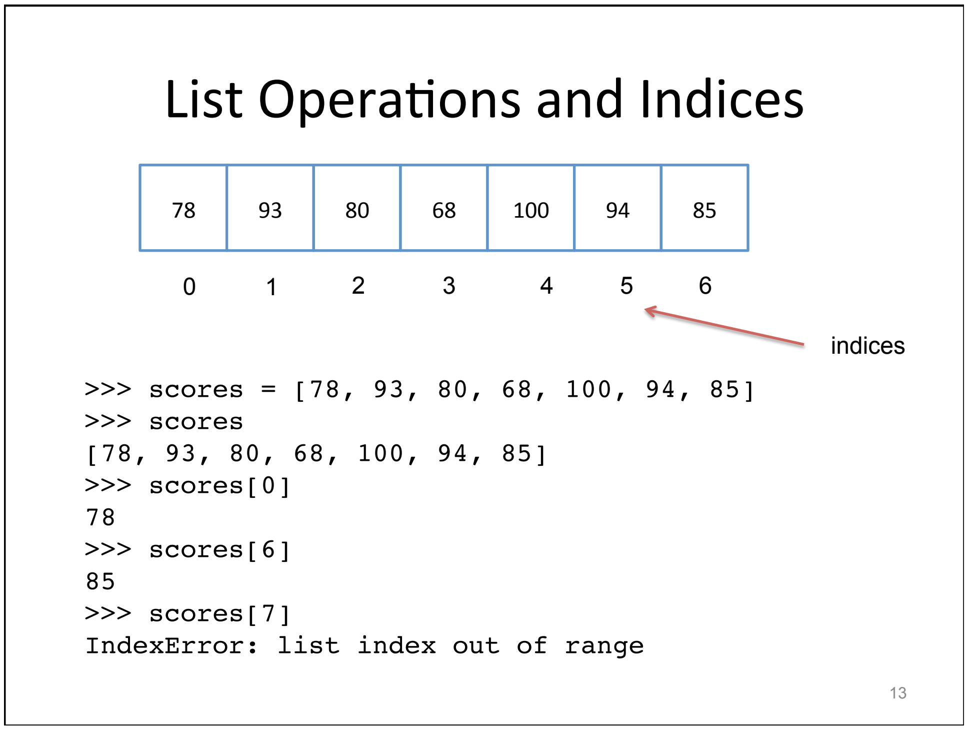
Representing Lists in Python

We will use a list to represent a collection of data values.

```
scores = [78, 93, 80, 68, 100, 94, 85]

colors = ['red', 'green', 'blue']
```

A list is an *ordered* sequence of values.



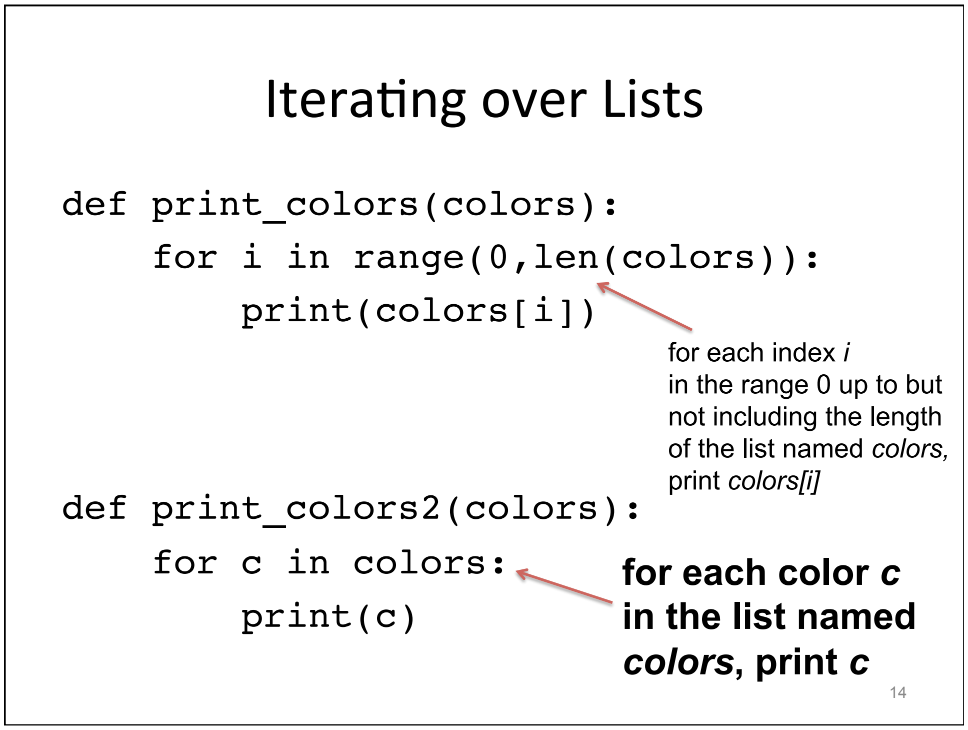
List Operations and Indices

78	93	80	68	100	94	85
0	1	2	3	4	5	6

indices

```
>>> scores = [78, 93, 80, 68, 100, 94, 85]
>>> scores
[78, 93, 80, 68, 100, 94, 85]
>>> scores[0]
78
>>> scores[6]
85
>>> scores[7]
IndexError: list index out of range
```

13



Iterating over Lists

```
def print_colors(colors):
    for i in range(0,len(colors)):
        print(colors[i])
```

for each index *i*
in the range 0 up to but
not including the length
of the list named *colors,*
print *colors[i]*

```
def print_colors2(colors):
    for c in colors:
        print(c)
```

for each color *c* in the list named *colors*, print *c*

14