Administrivia

- Homework 4 is due Friday
- Midterm 1 is February 29th at 7pm
REVIEW
s="ABcd"
if not s[0:2].isupper():
    if s[0]==s[2]:
        print(s[0])
    else:
        print(s[1])
else:
    if s[1]!=s[2]:
        print(s[-1])
    else:
        print(s[-2])
s=“abcd”
if not s.isalpha():
    print(s[0])
elif s.isupper():
    print(s[-1])
elif “ab” in s:
    print(s[-2])
else:
    print(s[1])
Exercise

• Validate password
  – At least 8 characters long
  – Upper and lower case characters
  – At least one non-alphabetic character
  – First three symbols must be distinct

• validate_password(“ABC”) → False
• validate_password(“AA9aaaaa”) → True
Solution

def validate_password(password):
    if not len(password) >= 8:
        return False
    elif password.isupper():
        return False
    elif password.islower():
        return False
    elif password.isalpha():
        return False
    elif password[0] == password[1]:
        return False
    elif password[1] == password[2]:
        return False
    elif password[0] == password[2]:
        return False
    else:
        return True
QUICK ASIDE
Shorthand

- $a+=b$ is shorthand for $a=a+b$
- $a-=b$ is shorthand for $a=a-b$
- $a*=b$ is shorthand for $a=a*b$
- $a/=b$ is shorthand for $a=a/b$
LOOPING
While loop

• Allows for *repeated execution* of code
• Execute a block over and over as long as a Boolean condition is True
• *Stop executing* if Boolean condition is False
While loop

• We create an **while loop** by typing:
  1. the keyword **while**
  2. a Boolean expression
  3. a **block** of code
x=3
while x>0:
    print("Hello")
    x-=1

How many times is "Hello" printed?

a) 0
b) 1
c) 2
d) 3
e) 4
Exercise

• Password creation:
  – Call validate password
  – Repeat until user inputs a valid password.
pwd=input("Enter a password: ")
while not validate_password(pwd):
    pwd=input("INVALID! Reenter: ")
print("Your password is valid")
Infinite loop

while True:
    print "Hello"

• **ALWAYS**: Statements *inside* the loop *must* change the loop condition!

• CTRL-C will stop the loop
Accumulator pattern

• Common and useful pattern to design programs

• *Accumulator* variable keeps track of result
  – Updated in each loop iteration
i=0
sum=0
while i<=4:
    sum+=i
    i+=1

a) 6
b) 10
c) 15
d) None of the other answers.
i=0
sum=0
while i<7:
    if (i%2)==1:
        sum+=i
        i+=1

a) 9
b) 12
c) 16
d) 21
Exercise

• Write a function to sum all of the digits in a number

$$\text{sum}(12145) \rightarrow 1+2+1+4+5 \rightarrow 13$$
Solution

def sum_digits(n):
    s= str(n)
    i=0
    result=0
    while i<len(s):
        result= result+ int(s[i])
        i=i+1
    return result
FOR LOOPS
For loop

• Loop construct to make our lives easier
• Used to iterate over *iterable* types
  – Example: strings (more to come)
• Step through a sequence “one at a time”
For loop

• We create an for loop by typing:
  1. the keyword for
  2. a loop variable (just a variable)
  3. the keyword in
  4. an iterable
  5. a block of code
Example

```python
my_string=“abcdefg”
for letter in my_string:
    print(letter)
```
def sum_digits(n):
    result=0
    for letter in str(n):
        result+=int(letter)
    return result
```python
s = "abcdefg"
t = ""
for c in s:
    t = c + t
```

What is the value of t?
a) "abcdefg"
b) "gfedcba"
c) "a"
d) "g"