Python Basics!

lists and loops

CS101 Lecture #7

2016-09-14

Administrivia

Administrivia

- Homework #3 is due Friday Sep. 16.
- Midterm #1 will be Monday Oct. 3, covering through Lecture #11. (evening)

Warmup Quiz

Question #1

```
a = 1
def fun(a,b):
    return a + b
a = fun( a,a ) + a
```

What is the final value of `a`?

A 2

B 3 ⋆

C 4

Question #1 (Worked)

```
a = 1
def fun(c,b):
    return c + b
a = fun( a,a ) + a
```

Question #2

```
x = 10
if ((x/2) < 5) or ((x%3) == 1):
    x = x + 2
if (x != 10) or ((x**2) <= 144):
    x = x * 2
```

What is the final value of x?

A `10`
B `12`
C `20`
D `24` ⋆

Question #3

```
def fun(x):
    if x and x:
        return not x
    return x or x

x = fun(True) or fun(False)
```

What is the final value of x?

A `True`

B `False` ⋆

Question #4

```
def fun(a,b):
    if len(a)+len(b)>5:
        return (a+b)[0:5]
  return (b+a)+str(len(a))

x = fun(”abc”,”def”) + fun(”gh”,”ij”)
```

What is the final value of x?

A ’abcdefijgh4’

B ’defabcghij4’

C ’abcdeijgh’

D ’abcdeijgh4’ ⋆

Question #5

The following code should increment x if the hundreds place contains a zero:

```
def fun(x):
    if x < 100 or ???:
        return x+1
    return x
```

What should replace the `???` to complete the code? Assume x is an integer.

A `x.string(3) == '0'`

B `str(x)[-3] == '0'` ⋆

C `((x//100) % 10) == 0` ⋆

D None of the above.

Container Data Types

Example

```
colors = [ 'red', 'yellow', 'blue',
           'jale', 'ulfire' ]
for color in colors:
    print( color.title() )
```

`list` data type

- The `list` type represents an ordered collection of items.
- `list` is an iterable and a container.
- Containers hold values of any type (doesn't have to be the same).

`list` statement

- We create a `list` as follows:
  - opening bracket [
  - one or more comma-separated data values
  - closing bracket]

`list` statement

- `lists` work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( len(x) )
```

list statement

- But strings are immutable (we can't change contents without creating a new string):

```
s = "good advise"
s[9] = 'c'                          # nope

s = s[:9] + 'c' + s[9:]             # this way
```

`list` statement

- We can change `list` content—they are mutable.

```
x = [ 4,1,2,3 ]
x[3] = -2
x.append(5)
del x[1]
x.sort()
```

← item assignment

Loops

Loops

- We frequently need to process each value in a set of values.
- Two kinds: `while` and `for`

Example: *while* Loop

```
number = 10
while number > 0:
    print(number)
    number = number - 1
print('Blast off!')
```

Defining loops: *while*

- A `while` loop has only:
  - the keyword `while`
  - a logical comparison (`bool`-valued result)
  - a **block** of code

Example

The following code should increment x if the hundreds place contains a zero:

```
x = 3
while x > 0:
    print(”Hello”)
    x -= 1
```

How many times is ’Hello’ printed?

A zero
B once
C twice
D thrice
E four times

String comparison methods

- These produce Boolean output.

`isdigit()`	Does a string contain only numbers?
`isalpha()`	Does a string contain only text?
`islower()`	Does a string contain only lower-case letters?
`isupper()`	Does a string contain only upper-case letters?

Example: String comparison

```
answer = input( 'How do you feel?  ' )
if not answer.isalpha():
    print( "I don't understand." )
else:
    print( "Ah, you feel %s." % answer )
```

Exercise

Write a program for a user to create a new password. The program should accept a password attempt from the user and check it with the function `validate_password`. If the password is valid, the program ends. If the password is invalid, the program asks for a new attempt, repeating until the user enters a valid password.

Solution

```
pwd = input(”Enter a password:  ”)
while not validate_password(pwd):
    pwd = input(”INVALID!  Try again:  ”)
print(”Your password is valid.”)
```

Infinite loops

- Make sure that your code always has a way to end!

```
while True:
    print('Hello!')
```

Infinite loops

- Make sure that your code always has a way to end!

```
while True:
    print('Hello!')
```

- Use `Ctrl+C` to break free.

Accumulator pattern

- Design patterns are common structures we encounter in writing code.
- The accumulator pattern uses an accumulator variable to track a result inside of a loop:

```
i = 0
sum = 0
while i <= 4:
    sum += i
    i += 1
```

Example

```
i = 0
sum = 0
while i <= 4:
    sum += i
    i += 1
```

What is the value of `sum`?

A `6`
B `10`
C `15`
D None of the above.

Example

```
i = 0
sum = 0
while i < 7:
    if (i % 2) == 1:
        sum += i
        i += 1
```

What is the value of `sum`?

A 9
B 12
C 16
D 21

Exercise

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

$$12145 \rightarrow 1 + 2 + 1 + 4 + 5 \rightarrow 13$$

Solution (*while*)

```
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
```

Reminders

Reminders

- Homework #3 is due Friday Sep. 16.
- Midterm #1 will be Monday Oct. 3. (evening)