PYTHON LAB BOOK

Python For Programmers

UCSC Extension Online

Lab 3 for range

Topics

- `range` operator
- `for` loop
- tuples

v.4.0

lab02_1.py
```
#!/usr/bin/env python
"""lab02_1.py Inputs two integers and determines whether
the first is a multiple of the second. """

while True:  # True/False are keywords.
    try:
        number1 = int(raw_input("Number please: "))
        break
    except ValueError:
        print "Please try again."

while True:
    try:
        number2 = int(raw_input("Number please: "))
        break
    except ValueError:
        print "Please try again."

if number1 % number2 == 0:
    print '%d is a multiple of %d' % (number1, number2)
else:
    print '%d is not a multiple of %d' % (number1, number2)

"""
$ lab02_1.py
Number please: 8
Number please: 2
8 is a multiple of 2
$ lab02_1.py
Number please: 18
Number please: 17
18 is not a multiple of 17
$ """
```

lab02_2.py

```
#!/usr/bin/env python
"""lab02_2.py Displays the octal and hexadecimal
representation of a number"""

while True:
    try:
        number = int(raw_input("Number please: "))
        break
    except ValueError:
        print "Please try again."

print "Octal = %#o Hexadecimal = %#x" % (number, number)

"""
$ lab02_2.py
Number please: 17
Octal = 021 Hexadecimal = 0x11
$

"""
```

lab02_3.py

```
#!/usr/bin/env python
""" lab02_3.py I'm thinking-of-a-number game. """

print "Think of a number between 1 and 10 and I'll try to guess it."
high = 10
low = 1
guesses = 0
while high > low:
    guesses += 1
    guess = (high + low)/2
    print 'Is your number %d?' % guess
    while True:
        answer = raw_input("""Please press:
        'y' for yes
        'n' for no
        """)
        answer = answer[0].lower()
        if answer == 'y' or answer == 'n':
            break
        print 'Please follow directions.'
    if answer == 'y':
        print 'Hurray! Only', guesses, "guesses."
        break

    while True:
        answer = raw_input("""No?  Then please press:
        'h' if %d is higher than your number
        'l' if %d is lower than your number
        """ % (guess, guess))
        answer = answer[0].lower()
        if answer == 'l' or answer == 'h':
            break
        print 'Please follow directions'

    if answer == 'l':
        low = guess + 1
    else:
        high = guess - 1

"""
$ lab02_3.py
Think of a number between 1 and 10 and I'll try to guess it.
Is your number 5?
Please press:
        'y' for yes
        'n' for no
        n
No?  Then please press:
        'h' if 5 is higher than your number
        'l' if 5 is lower than your number
        h
Is your number 2?
Please press:
        'y' for yes
        'n' for no
        y
Hurray! Only 2 guesses.
$
"""
```

```
>>> range(10)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> range(5, 10)

[5, 6, 7, 8, 9]

>>> range(2, 11, 2)

[2, 4, 6, 8, 10]

>>> range(10, 0, -1)

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

>>>
```

`range([start=0,] almost_end[, increment=1])`

All the same:

```
range(10)
range(0, 10)
range(0, 10, 1)
```

for_loop.py
```
#!/usr/bin/env python
""" Demonstrates a for loop """

numbers = range(5)
print numbers
for num in numbers:
    print "%d * 2 = %d" % (num, num * 2)

"""
OUTPUT:
$ for_loop.py
[0, 1, 2, 3, 4]
0 * 2 = 0
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8

(rest of output is below)
"""
# Use xrange(5) with a for loop.  It works with
# for/in to generate the numbers one at a time:

for num in xrange(5):
    print "%d * 2 = %d" % (num, num * 2)

"""
(output continued)

0 * 2 = 0
1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
$
"""
```

Lab 03

1. How would you produce the following using the `range` operator?

   ```
   [3, 6, 9, 12]
   [-10, 100, 210]
   -1, -3, -5, -7,
   ```

   Notice that the last one has no square brackets.

2. Produce this output using `range` and `for`:

   ```
   10, 9, 8, 7, 6, 5, 4, 3, 2, 1, BLASTOFF!!!
   ```

3. Try this in the interpreter:

   ```
   for ch in "Howdy":
      print ch

   for num in 2, 4, 16:
      print num
   ```

   Strings and comma-separated objects, maybe in ()'s, called "tuples", are "sequences" and can be iterated with the `for` and `in`.

   And try this:

   ```
   for thing in (2, "hat", (0, 1)):
       print thing
   ```

   A tuple can contain any sort of object, even nested tuples.

4. Use a for loop and a tuple of strings to produce:

   ```
   Hi ya Manny!
   Hi ya Moe!
   Hi ya Jack!
   ```

   Can you do it without duplicating any code?

5. (Optional) Print the decimal equivalent of a binary string that is given by the user:

   ```
   Binary string: 1011
   Decimal equivalent: 11
   ```

   Try it using a for-loop and a while-loop.

   Then, (not optional), at the interpreter prompt, type:

   ```
   help(int)
   ```

   So, what is the easiest way to do this exercise?