PYTHON LAB BOOK

Python For Programmers

UCSC Extension Online

Lab 2 Input

Topics

- Input from `stdin`
- Factory functions
- Catching an exception:
- yet another `else`
- Formatted strings
- Integer division issue

v.4.0

lab01_3.py
```
#!/usr/bin/env python
"""lab01_3.py -- produces strings with embedded strings."""

print """He said "Hello World".
She said 'Hello Sky'.
She said "He said 'Hello World'".
"""

"""
$ lab01_3.py
He said "Hello World".
She said 'Hello Sky'.
She said "He said 'Hello World'".
$
"""


```

lab01_4.py
```
#!/usr/bin/env python
"""lab01_4.py produces:
10 9 8 7 6 5 4 3 2 1 BLASTOFF!!!
"""
count = 10
while count > 0:
    print count,
    count -= 1
print 'BLASTOFF!!!'

"""

$ lab01_4.py
10 9 8 7 6 5 4 3 2 1 BLASTOFF!!!
$

"""
```

lab01_5.py

```
#!/usr/bin/env python
"""lab01_5.py -- produces a christmas tree pattern of stars."""

line = 0
stars = 1
blanks = 12
while line < 10:
    i = 0
    while i < blanks:
        print ' ',
        i += 1
    i = 0
    while i < stars:
        print '*',
        i += 1
    print
    line += 1
    stars += 2
    blanks -= 1

"""
$ lab01_5.py
                        *
                      * * *
                    * * * * *
                  * * * * * * *
                * * * * * * * * *
              * * * * * * * * * * *
            * * * * * * * * * * * * *
          * * * * * * * * * * * * * * *
        * * * * * * * * * * * * * * * * *
      * * * * * * * * * * * * * * * * * * *
$
"""

```

lab01_5x.py

```
#!/usr/bin/env python
"""lab015x.py -- produces a christmas tree pattern of stars.
The easy way. """


line = 0
stars = 1
blanks = 12
while line < 10:
    print '  ' * blanks + '* ' * stars
    line += 1
    stars += 2
    blanks -= 1

"""
$ lab015x.py
                        *
                      * * *
                    * * * * *
                  * * * * * * *
                * * * * * * * * *
              * * * * * * * * * * *
            * * * * * * * * * * * * *
          * * * * * * * * * * * * * * *
        * * * * * * * * * * * * * * * * *
      * * * * * * * * * * * * * * * * * * *
$
"""
```

get_input.py
```
#!/usr/bin/env python
"""Demonstrates input."""

answer = raw_input('What is your favorite number? ')
print 'You like', answer, '?'
print 'You really like ' + answer + '?'
number = int(answer)
if number < 10:
    print number, 'is small.'
elif number >= 1000:
    print number, 'is big.'
else:
    print number, 'is medium.'

print 'But you like ' + str(number) + '!'
"""
$ get_input.py
What is your favorite number? 8
You like 8 ?                     <-  space before the '?'!
You really like 8?
8 is small.
But you like 8!
$
Notes:

int(x)   -> converts to integer
str(x)   -> converts to string
float(x) -> converts to float

$ get_input.py
What is your favorite number? x
You like x ?
You really like x?
Traceback (most recent call last):
  File "get_input.py", line 7, in ?
    number = int(answer)
ValueError: invalid literal for int(): x
$


"""
```

try_input.py

```
#!/usr/bin/env python
"""Demonstrates catching an exception on error."""

answer = raw_input('What is your favorite number? ')
print 'You like', answer, '?'
print 'You really like ' + answer + '?'
try:
    number = int(answer)
except ValueError:
    print answer, 'is not a number!'
else:
    if number < 10:
        print number, 'is small.'
    elif number >= 1000:
        print number, 'is big.'
    else:
        print number, 'is medium.'

    print 'But you like ' + str(number) + '!'
"""
$ try_input.py
What is your favorite number? 44
You like 44 ?
You really like 44?
44 is medium.
But you like 44!
$ try_input.py
What is your favorite number? xx
You like xx ?
You really like xx?
xx is not a number!
$
"""
```

paint.py

```
#!/usr/bin/env python
"""Demonstrates formatted output, and both uses of '%'
1.  string replacement
2.  modulo
"""
PER_GALLON = 200      # A can of paint covers 200 square feet
sq_ft = 0
while sq_ft == 0:
    input = raw_input("Number of square feet to paint: ")
    if not input:   # or if input == '':
        print "Thank you anyway."
        break
    try:
        sq_ft = int(input)
    except ValueError:
        print "Please give a whole number."
else:
    cans = sq_ft/PER_GALLON
    if sq_ft % PER_GALLON > 0:
        cans += 1
        print "You need %.1f cans so you'd better buy %d %s."\
              % (float(sq_ft)/PER_GALLON, cans,
                 "can" if cans==1 else "cans")
    else:
        print "You need exactly %d %s." % (cans,
                                         "can" if cans==1 else "cans")
"""
$ paint.py
Number of square feet to paint: 250
You need 1.2 cans so you'd better buy 2 cans.
$ paint.py
Number of square feet to paint: 201
You need 1.0 cans so you'd better buy 2 cans.
$ paint.py
Number of square feet to paint: 400
You need exactly 2 cans.
$ paint.py
Number of square feet to paint:
Thank you anyway.
$"""
```

Formatting Strings

```
"string with % language" % (argument for every % and *)
```

	OPTIONAL				
%	**flag**	**m**	**.**	**n**	**?**
	minimum width all conversions			*precision*	*conversion character*
	reads the width from the arguments *			* reads the precision from the arguments	
-	left justify			for integers - forces `n` digits	`d, i` `o, x, X`
+	sign on positive values				
␣	blank instead of plus			for strings - maximum characters	`s, r`
0	pad with zeros				
#	prepends: `0` for `%#o` `0x` or `0X` for `%#x` or `%#X`			for floats - digits to right of decimal	`f, e, E`
				general floats - significant figures	`g, G`
				%% makes one %	%

Notes:

- `%s` and `%r` always work.
- `r"\n"` is a `raw` string and makes `\n`.
  Backslashes are not escaped if you prepend a `r`.

```
Printing 12.3456
      "|%d|" % (12.3456) -> |12|
      "|%o|" % (12.3456) -> |14|
      "|%X|" % (12.3456) -> |C|
     "|%#o|" % (12.3456) -> |014|
     "|%#x|" % (12.3456) -> |0xc|
     "|%#X|" % (12.3456) -> |0XC|
     "|%+d|" % (12.3456) -> |+12|
     "|%6d|" % (12.3456) -> |    12|
    "|%06d|" % (12.3456) -> |000012|
    "|%-6d|" % (12.3456) -> |12    |
     "|%1d|" % (12.3456) -> |12|
      "|%f|" % (12.3456) -> |12.345600|
      "|%g|" % (12.3456) -> |12.3456|
    "|%.1f|" % (12.3456) -> |12.3|
    "|%.2f|" % (12.3456) -> |12.35|
  "|%10.1f|" % (12.3456) -> |      12.3|
 "|%-10.1f|" % (12.3456) -> |12.3      |
   "|%*.*f|" % (5, 2, 12.345599999999999) -> |12.35|

Printing "Croque Monsieur"
      "|%s|" % ("Croque Monsieur") -> |Croque Monsieur|
    "|%20s|" % ("Croque Monsieur") -> |     Croque Monsieur|
   "|%-20s|" % ("Croque Monsieur") -> |Croque Monsieur     |
     "|%5s|" % ("Croque Monsieur") -> |Croque Monsieur|
    "|%.5s|" % ("Croque Monsieur") -> |Croqu|

-----
```

```
          Division Issue

          >>> 3/10
          0
          >>> from __future__ import division
          >>> 3/10
          0.29999999999999999
          >>> 3//10
          0       <--- //  2 slashes will always mean integer division.
```

Lab 02 (Demo: indenting in the interpreter and `pass`.)

1. Ex 2.7 Deitel: Write a program that reads in two integers and determines whether the first is a multiple of the second.

2. Write a program that requests a number from the user and then prints out the hexadecimal and octal representations of the number.

3. (Optional) Write a program that asks the user to think of a number between 1 and 10. The program then tries to guess the number and asks the user if the computer's guess is right. If the guess is not right, ask the user if the guess is high or low and then take another guess. My output looks like:

```
$ 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 3?
Please press:
        'y' for yes
        'n' for no
        n
No?  Then please press:
        'h' if 3 is higher than your number
        'l' if 3 is lower than your number
        l
Is your number 4?
Please press:
        'y' for yes
        'n' for no
        y
Hurray!
$
```