



Lecture 2

expressions, variables, for loops

Special thanks to CS Washington CS 142

Except where otherwise noted, this work is licensed under:
<http://creativecommons.org/licenses/by-nc-sa/3.0>

Who uses Python?



- “Python is fast enough for our site and allows us to produce maintainable features in record times, with a minimum of developers”
-Cuong Do, Software Architect, YouTube.com

Expressions

- Arithmetic is for numeric values
 - Operators: + - * / % (plus ** for exponentiation)
 - Python built-in operators.
 - Precedence: () before ** before * / % before + -
 - Integers vs. real numbers
 - You may use // for integer division

```
>>> 1 + 1
2
>>> 1 + 3 * 4 - 2
11
>>> 7 // 2
3
>>> 7 / 2
3.5
>>> 7.0 / 2
3.5
```

Variables

- Variable can be used to store a value.
- Declaring
 - no type is written; same syntax as assignment
 - Try these Python commands in Python prompt

Python
<pre>x = 2 x = x + 1 print(x) x = x * 8 print(x) d = 3.2 d = d / 2 print(d)</pre>

Types

- Python is looser about types
 - Variables' types do not need to be declared
 - Variables can change types as a program is running

Value	Python type
42	int
3.14	float
"hi!"	str

- In Python prompt, try:

```
>>> x=42
>>> type(x)
<class 'int'>
>>> x=4.2
>>> type(x)
<class 'float'>
>>> x='4.2'
>>> type(x)
<class 'str'>
>>>
```

String Multiplication

- Python strings can be multiplied by an integer.
 - The result is many copies of the string concatenated (chained) together.

```
>>> "hello" * 3
"hellohellohello"

>>> print(10 * "yo ")
yo yo yo yo yo yo yo yo yo yo

>>> print(2 * 3 * "4")
444444
```

String Concatenation

- Integers and strings cannot be concatenated in Python.
 - Workarounds:
 - `str(value)` - converts a value into a string
 - `print value, value2` - prints value and value2, separated by a space

```
>>> x = 4
>>> print("Thou shalt not count to " + x + ".")
TypeError: cannot concatenate 'str' and 'int' objects

>>> print("Thou shalt not count to " + str(x) + ".")
Thou shalt not count to 4.

>>> print(x + 1, "is out of the question.")
5 is out of the question.
```

The for Loop

- for **name** in range(**max**):
- **statements**
- Repeats for values 0 (inclusive) to **max** (*exclusive*)

```
>>> for i in range(5):  
...     print(i)  
0  
1  
2  
3  
4
```


for Loop Variations

- for **name** in range(**min**, **max**):
- **statements**
- for **name** in range(**min**, **max**, **step**):
- **statements**
- Can specify a minimum other than 0, and a step other than 1

```
>>> for i in range(2, 6):  
...     print(i)  
2  
3  
4  
5  
>>> for i in range(15, 0, -5):  
...     print(i)  
15  
10  
5
```

Nested Loops

- Nested loops are often replaced by string * and +

```
.....1
....2
...3
..4
.4
5
```

Python

```
1 for line in range(1, 6):
2     print((5 - line) * "." + str(line))
```

For Loops Example: find the sum from 0 to 10

- Without a loop:

Python

```
1 sum = 0+1+2+3+4+5+6+7+8+9+10
2 print(sum)
```

- With a loop

Python

```
1 sum = 0
2 for value in range(0, 11):
3     sum = sum + value
4 print(sum)
```

Exercise

- Rewrite the Mirror lecture program in Python. Its output:

```
#=====#
|           <><>           |
|        <>.....<>        |
|     <>.....<>           |
|<>.....<>           |
|<>.....<>           |
|     <>.....<>           |
|        <>.....<>        |
|           <><>           |
#=====#
```

- Make the mirror resizable by using the variable "SIZE"

Exercise Solution

```
SIZE = 4

def bar():
    print("#" + "=" * (4 * SIZE) + "#")

def top():
    for line in range(1, SIZE + 1):
        # split a long line by ending it with \
        print("|" + (-2 * line + 2 * SIZE) * " " + \
              "<>" + (4 * line - 4) * "." + "<>" + \
              (-2 * line + 2 * SIZE) * " " + "|")

def bottom():
    for line in range(SIZE, 0, -1):
        print("|" + (-2 * line + 2 * SIZE) * " " + \
              "<>" + (4 * line - 4) * "." + "<>" + \
              (-2 * line + 2 * SIZE) * " " + "|")

# main
bar()
top()
bottom()
bar()
```

Concatenating Ranges

- Ranges can be concatenated with +
 - However, you must use the “list()” command
 - Can be used to loop over a disjoint range of numbers

```
>>> list(range(1, 5)) + list(range(10, 15))
[1, 2, 3, 4, 10, 11, 12, 13, 14]

>>> for i in list(range(4)) + list(range(10, 7, -1)):
...     print(i)
0
1
2
3
10
9
8
```

Exercise Solution 2

- `SIZE = 4`
- `def bar():`
- `print "#" + 4 * SIZE * "=" + "#"`
- `def mirror():`
- `for line in list(range(1, SIZE + 1)) + list(range(SIZE,`
- `0, -1)):`
- `print("|" + (-2 * line + 2 * SIZE) * " " + \`
- `"<>" + (4 * line - 4) * "." + "<>" + \`
- `(-2 * line + 2 * SIZE) * " " + "|")`
- `# main`
- `bar()`
- `mirror()`
- `bar()`