

10/23/07



- * if else
- * returns
- * input



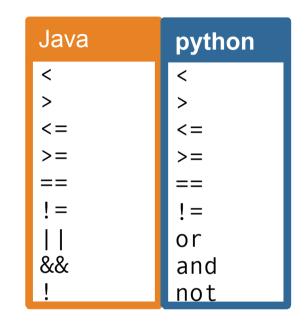
>>> if

Like many things in the transition from Java to Python, curly braces are replaced with colons and whitespace, the parentheses are dropped and &&, || and ! change.

Translator.java

```
1 // 1 for english
2 // 2 for german
3 int translator = 1;
4 if (translator == 1) {
    english();
5 } else if (translator == 2) {
    german();
8 } else {
9 System.out.println("None");
10 }
```





Notice: "else if" becomes "elif"

translator.py

```
1 translator = 1
2 if translator==1:
3 english()
4 elif translator==2:
5 german()
else:
        print "None"
```

>>> strings

Just like in Java, strings are objects. Here are some things you can do with them:

| string methods | | |
|--------------------------------------|------------------------|------------------|
| <pre>s.capitalize()</pre> | "wow".capitalize() | => "Wow" |
| <pre>s.endswith(<str>)</str></pre> | "wow".endswith("w") | => True |
| <pre>s.find(<substr>)</substr></pre> | "wow".find("o") | => 1 |
| s.islower() | "wow".islower() | => True |
| s.isupper() | "wOw".isupper() | => False |
| s.lower() | "wOw".lower() | => "wow" |
| <pre>s.split(<str>)</str></pre> | "hmm wow".split(" ") | => ["hmm","wow"] |
| <pre>s.startswith(<str>)</str></pre> | "hmm".startswith("hm") | => True |
| s.strip() | " ack ".strip() | => "ack" |
| s.swapcase() | "wOw".swapcase() | => "WoW" |
| s.upper() | "wow".upper() | => "WOW" |

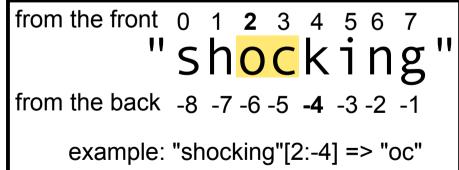


>>> strings as sequences

As we saw with loops, python has certain sequence types such as lists. Strings are also sequences. Indexes start with 0 the left and -1 on the right.

sequence operations seq[<index>] "look!"[3] => "k" "look!"[-1] => "!" seq[<start>:<end>] "shocking"[4:] => "king" "shocking"[3:5] => "ock" "shocking"[-3:] => "ing" len(<seq>) len("whoa") => 4

Indexing





>>> return

Returns in python are super easy. Simply "return <value>" instead of "return <value>;" and forget about the types.

funky.py

| 1 | def funky(s): |
|----|--|
| 2 | if len(s)<=3: |
| 3 | <pre>return s.lower()</pre> |
| 4 | else: |
| 5 | <pre>return s.upper()</pre> |
| 6 | |
| 7 | s1 = funky("wow") |
| 8 | print s1 #"wow" |
| 9 | <pre>if len(s)<=3: return s.lower() else: return s.upper() s1 = funky("wow") print s1 #"wow" s2 = funky("whoa")</pre> |
| 10 | print s2 #"WHOA" |



>>> input() vs. raw_input()

There are two ways of getting input. The first is input(). It takes in input until enter is hit and then tries to interpret it into python. However, this way only works well for numbers.

```
>>> x = input("yes? ")
yes? y
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
    File "<string>", line 1, in <module>
NameError: name 'y' is not defined
>>> x = input("yes? ")
yes? 2
>>> print x
2
>>> x = input("num? ")
num? 2.0
>>> print x
2.0
```



The second way is to use raw_input() which returns the entire line as a string. Once this is done, the string can be split into smaller strings and changed to the desired type.

inputs.py

10

```
1
  #take a number in
2
  x = input("x?")
3
  print x
4
5
  #take a sentence to tokenize it
6
  sent = raw input("sentence: ")
7
  for w in sent.split(" "):
8
      print "word: " + w
9
```

>>> igpay atinlay

scott @ yossarian ~ \$ python translate.py

Translators: 1. Angry.

2. Pig Latin.

Which translator would you like to use? $\underline{2}$

What would you like me to translate? Look! Pig latin!

Translated:
Ooklay! Igpay atinlay!





© 2007 Scott Shawcroft, Some Rights Reserved

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

Python® and the Python logo are either a registered trademark or trademark of the Python Software Foundation. Java[™] is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.