python™

10/23/07

>>> Overview

* 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.

Java	python
`<`	`<`
`>`	`>`
`<=`	`<=`
`>=`	`>=`
`==`	`==`
`!=`	`!=`
`||`	`or`
`&&`	`and`
`!`	`not`

Translator.java

```java
// 1 for english
// 2 for german
int translator = 1;
if (translator == 1) {
    english();
} else if (translator == 2) {
    german();
} else {
    System.out.println("None");
}
```

Notice: "`else if`" becomes "`elif`"

translator.py

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

python™

>>> strings

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

string methods

`s.capitalize()`	`"wow".capitalize()`	`=> "Wow"`
`s.endswith(<str>)`	`"wow".endswith("w")`	`=> True`
`s.find(<substr>)`	`"wow".find("o")`	`=> 1`
`s.islower()`	`"wow".islower()`	`=> True`
`s.isupper()`	`"wOw".isupper()`	`=> False`
`s.lower()`	`"wOw".lower()`	`=> "wow"`
`s.split(<str>)`	`"hmm wow".split(" ")`	`=> ["hmm","wow"]`
`s.startswith(<str>)`	`"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

```
from the front  0  1  2  3  4  5  6  7
               "s  h  o  c  k  i  n  g"
from the back  -8 -7 -6 -5 -4 -3 -2 -1
```

example: "shocking"[2:-4] => "oc"

python™

>>> return

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

funky.py

```
def funky(s):
    if len(s)<=3:
        return s.lower()
    else:
        return s.upper()

s1 = funky("wow")
print s1   #"wow"
s2 = funky("whoa")
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

```
#take a number in
x = input("x? ")
print x

#take a sentence to tokenize it
sent = raw_input("sentence: ")
for w in sent.split(" "):
    print "word: " + w


```

>>> igpay atinlay

```
scott @ yossarian ~ $ python  translate.py
Translators:
1. Angry.
2. Pig Latin.

Which translator would you like to use? 2

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

Translated:
Ooklay! Igpay atinlay!
```