LING 388: Computers and Language

Lecture 8
Administrivia

- Next week I'm out of town
- The TA Patricia Lee will lead both classes
- You will do Python programming exercises in class
Python numbers revisited

- Python invisibly automatically converts to type `long int` from `int` (32 bit 2's complement), (page 64–65, section 3.5)

- explicit type coercion:
  1. `float()`
  2. `int()`
  3. `long()` Python 2.7
  4. `complex(r, i)`
  5. `complex(string)`
Python numbers revisited

• Note the difference in precision here

```python
>>> 2**float(90)
1.2379400392853803e+27
>>> 2**90
1237940039285380274899124224
```
Python numbers revisited

In mathematics, factorial is often denoted with an exclamation (!). Factorial of a whole number \( n \) is defined as \( n! = n(n-1)(n-2)\ldots(1) \).

```python
# factorial.py
# Program to compute the factorial of a number
# Illustrates for loop with an accumulator

def main():
    n = input("Please enter a whole number: ")
    fact = 1
    for factor in range(n,1,-1):
        fact = fact * factor
    print("The factorial of", n, "is", fact)

main()
```

Python 2.7

```python
>>> type(input())
12
<class 'int'>
```

Python 3.x

```python
>>> type(input())
34
<class 'str'>
```

```python
>>> a = 2
>>> print(input())
a
>>> print(eval(input()))
a
2
```
Python `eval()`

- Think of `eval(input())` as being the same as the user typing directly at the Python console ...

```python
>>> a = 2.0
>>> eval(input("Enter number "))
Enter number 2
2
>>> eval(input("Enter number "))
Enter number a
'a'
>>> eval(input("Enter number "))
Enter number a+2
4.0
>>> a+2
4.0
>>> "a"
'a'
```
Python numbers revisited

```python
1def main():
2    n = int(input("Please enter a whole number: ")
3    fact = 1
4    for factor in range(n,1,-1):
5        fact = fact * factor
6    print("The factorial of", n, "is", fact)
7
8main()
```

```python
from sys import argv
1def main():
2    try:
3        n = int(argv[1])
4        fact = 1
5        for factor in range(n,1,-1):
6            fact = fact * factor
7        print("The factorial of", n, "is", fact)
8    except:
9        print("Error: please enter a whole number!")
10main()
```

$ python3 factorial2.py
Error: please enter a whole number!
$ python3 factorial2.py the
Error: please enter a whole number!
$ python3 factorial2.py 12.2
Error: please enter a whole number!
$ python3 factorial2.py 12
The factorial of 12 is 479001600
$
Python argv

- List of arguments from the command line: what's argv[0] then?

```python
from sys import argv
print(argv[0])
```

```
$ python3 test.py
test.py
$ python3 test.py 121 2
test.py
```

Python example

```python
# futval.py
#    A program to compute the value of an investment
#    carried 10 years into the future

def main():
    print "This program calculates the future value"
    print "of a 10 year investment."

    principal = input("Enter the initial principal: ")
    apr = input("Enter the annualized interest rate: ")

    for i in range(10):
        principal = principal * (1 + apr)

    print "The value in 10 years is: ", principal

main()
```

How about printing to 2 decimal places?

```plaintext
This program calculates the future value of a 10 year investment.
Enter the initial principal: 1000
Enter the annualized interest rate: .035
The value in 10 years is: 1410.590876062
```
# futval.py
# A program to compute the value of an investment
carried 10 years into the future

def main():
    print "This program calculates the future value"
    print "of a 10 year investment."

    principal = input("Enter the initial principal: ")
    apr = input("Enter the annualized interest rate: ")

    for i in range(10):
        principal = principal * (1 + apr)

    print "The value in 10 years is: ", format(principal,'10.2f')

main()
• $100 principal for 10 years at 4% (0.04) compound interest = $148.02

```
>>> principal = 100
>>> for years in range(10):
...     principal *= 1 + 0.04
...
>>> print(principal)
148.02442849183444
>>> print(format(principal, '.2f'))
148.02
>>> print(format(principal, '.3f'))
148.024
>>> print(format(principal, '.4f'))
148.0244
>>> print(format(principal, '.5f'))
148.02443
>>> print(format(principal, 'd'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Unknown format code 'd' for object of type 'float'
```
• Lots of ways: (Old way)

```python
>>> print "x is %.2f" % 5
x is 5.00
```

```python
>>> print "x is %.2f and %2d" % (5,5)
x is 5.00 and 5
```

notation comes originally from the C Programming Language:
sprintf function
http://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm
Formatted Output

• `%<width>.<precision><type>`
• `<type>` = d, f, s
• `<width>` = minimum number of characters; right-justified by default, -width => left-justified 0 = as wide as needed
• `<precision>` = number of places after decimal point

• e.g. 02d          two digits wide, pad with 0 if needed

Now use instead: `<template-string>.format(<values>)`
Formatted Output: examples

```python
>>> print(format(100, 'd'))
100
>>> print(format(9, 'X'))
9
>>> print(format(10, 'X'))
A
>>> print(format(10, 'x'))
a
>>> print(format(148, 'x'))
94
>>> 9*16+4
148
>>> print(format(148, 'o'))
224
>>> 2*64+2*8+4
148
```

Function `str()` coerces integer or float above into type 'str'
Formatted Output

• Newer way:
  • [https://docs.python.org/2/tutorial/inputoutput.html#fancier-output-formatting](https://docs.python.org/2/tutorial/inputoutput.html#fancier-output-formatting)
  • Use {} for each argument
    • (can be numbered, e.g. {0}, {1},... or referenced by keyword {x})

```python
>>> import math
>>> print 'The value of PI is approximately {0:.3f}'.format(math.pi)
The value of PI is approximately 3.142.

>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print '{0:10} {1:10d}'.format(name, phone)
...
Jack   ==>  4098
Dcab   ==>  7678
Sjoerd ==>  4127
```
Formatted Output

0 in {0:..} can be omitted but not the colon

{0:..} and {1:....} index into the format arguments
A note on string encoding

**Python 2.7**

```python
>>> x = "á"
>>> type(x)
<type 'str'>
>>> x = u"á"
>>> type(x)
<type 'unicode'>
>>> len(x)
1
>>> x = "á"
>>> len(x)
2
```

**Python 3.x**

```python
[>>> x = "á"
[>>> type(x)
<class 'str'>
[>>> x = u"á"
[>>> type(x)
<class 'str'>
[>>> len(x)
1
[>>> len(x)
2
```