Ceng 111 – Fall 2015
Week 9b

Functions

Credit: Some slides are from the “Invitation to Computer Science” book by G. M. Schneider, J. L. Gersting and some from the “Digital Design” book by M. M. Mano and M. D. Ciletti.
Functions in Python

```python
1 def function-name(parameter-1, ..., parameter-N):
2     statement-1
3     .
4     .
5     statement-M
```

- Syntax is important!
- Indentation is extremely important
Functions in Python

- Write a Python function that reverses a given number
  - Example: If 123 is given, the output should be 321

```python
def reverse(a):
    return int(str(a)[::-1])
```
Default Parameters in Python

We can now call `reverse_num` as `reverse_num()` in which case `Number` is assumed to be 123.

If we supply a value for `Number`, that value is used instead.

```python
1 def reverse_num (Number=123):
2     
3     str_num = str(Number)
4     print ”Reverse of”, Number, ”is”, str_num[::-1]
```

```python
1 def f(Str, Number=123, Bst=”Some”):
2     print Str, Number, Bst
```
While we are at it...

let's have a look at commenting in Python

```python
1 def reverse_num(Number=123):
2     "reverse_num: Reverse the digits in a number"
3     str_num = str(Number)
4     print "Reverse of", Number, "is", str_num[::-1]
```

There are two different ways to put comments in Python: (1) You can use `#` in which case the rest of the line is not interpreted. (2) You can enclose multiple lines like """ <lines of text> """. The comments that are written using the second option are basically documentation strings and available through the `help` page.
Nested Functions in Python

```python
1 def f(N):
2     Number = N
3     def g():
4         C = 20
5         return N * Number
6     print ”Number”, N, ”and its square:”, g()
```

- Function `g()` can access all the local variables as well as the parameters of function `f()`.
- Function `f()` cannot access the local variables of function `g()`!
- Function `g()` cannot be used before it is defined! For example, the second line could not have been `Number = 10 * g(10)`.
- The indentation is extremely important to understand which statement belongs to which function! For example, the last line is part of function `f()` since they are at the same indentation!
Today

- Functions

- Note: THE2 has been announced!

- Reminder:
  - midterm date: 9 December at 17:40.
Note

THE-2 is announced.

- Deadline: 18th of December, 23:59
- Not subject to extensions.
- First write a function for solving the first step.
Note

- Never ever share your code with others in part or as a whole.
  - Check your home directory permissions.
- You can only discuss high-level "pseudo-code" level descriptions. If you end up talking about if-statements and their details, you are doing something wrong.
- If you feel like you are in a grey area about what you are discussing with your friends, then you are probably doing something wrong.
- If you have difficulty with the problem, come to us for hints; cheating is not a solution.
- You are encouraged to share inputs and the solutions for those inputs in the newsgroup. However, we will not post any test cases, nor comment on the correctness of your solutions.
- Pay attention to the specifications and the details. They do matter, and your grade will be affected.
Global Variables in Python

To access variables in the global workspace, you should use “global <varname>”

```python
1 N = 10
2 def f():
3     global N
4     def g(Number):
5         C = 20
6         return N * Number
7     N = g(N)
```
Since you can nest functions in Python, understanding scope is important.
Functional Programming in Python

- List Comprehension

```python
[<expr> for <var> in <list>]
```

Example:

```python
[3*x for x in [1, 2, 3, 4, 5]]
```

\[ [3, 6, 9, 12, 15] \]
Functional Programming in Python

- **filter(function, list)**

```python
1 def Positive(N):
2     if N > 0: return True
3     return False
```

`filter(Positive, [-10, 20, -2, 5, 6, 8, -3])`  
\[20, 5, 6, 8]\n
- **map(function, list)**

```python
1 def Mod4(N):
2     return N % 4
```

`map(Mod4, range(1, 10))`  
\[1, 2, 3, 0, 1, 2, 3, 0, 1]\n
Functional Programming in Python

- `reduce(function, list)`

```python
1 def greater(A, B):
2     return A if A > B else B

reduce(greater, [1, 20, 2, -30])
```

20
Parameter passing in functions

```python
define f(x)
    x[0] ← "jennie"
    x ← ["suzy", "mary", "daisy"]
return x
```

... 

```python
s ← ["bob", "arthur"]
print f(s)
print s
...
```

```
<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>s</td>
</tr>
<tr>
<td>[&quot;bob&quot;, &quot;arthur&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>[&quot;suzy&quot;, &quot;mary&quot;, &quot;daisy&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Call by Value

```
<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>s</td>
</tr>
<tr>
<td>[&quot;suzy&quot;, &quot;mary&quot;, &quot;daisy&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>[&quot;suzy&quot;, &quot;mary&quot;, &quot;daisy&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Call by Sharing

CALL BY REFERENCE

```
<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>s</td>
</tr>
<tr>
<td>[&quot;suzy&quot;, &quot;mary&quot;, &quot;daisy&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>[&quot;suzy&quot;, &quot;mary&quot;, &quot;daisy&quot;]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

(b) ["suzy", "mary", "daisy"]
["suzy", "mary", "daisy"]

(c) ["suzy", "mary", "daisy"]
["jennie", "arthur"]
Parameter passing in functions in Python

```python
define f(x)
    x[0] ← "jennie"
    x ← ["suzy", "mary", "daisy"]
    return x

...  
s ← ["bob", "arthur"]
print f(s)
print s
... 
```

**Call by Sharing**

- MISTAKE

```python
(c) ["suzy", "mary", "daisy"]
    ["jennie", "arthur"]
```
Parameter passing in functions in Python

```python
1 def f(N):
2     N = N + 20
3
4 def g():
5     A = 10
6     print A
7     f(A)
8     print A

>>> g()
10
10
```
Parameter passing in functions in Python

```python
1 def f(List):
2     List[0] = 'A'
3
4 def g():
5     L = [1, 2, 3]
6     print L
7     f(L)
8     print L

>>> g()
[1, 2, 3]
['A', 2, 3]
```
Parameter passing in functions in Python

```
1 def f(List):
2     List = List[::-1]
3
4 def g():
5     L = [1, 2, 3]
6     print L
7     f(L)
8     print L
```

```python
>>> g()
[1, 2, 3]
[1, 2, 3]
```
Exercises

- Write a function that calculates the factorial of a number \textit{without recursion or iteration}.

- Write a function that calculates the average of the numbers in a list \textit{without recursion or iteration}. 