MY470 Computer Programming

Writing and Calling Functions in Python

Week 4 Lab, MT 2017

Defining and Calling Functions

Defining a function

```
def *function_name*(*list of parameters*):
    *body of function*
```

Calling a function

```
*function_name*(*arguments*)
```

Functions can take 0 or more arguments and return 0 or more values!

Functions Take Arguments by Reference

In [1]:
```
def change_list(alist):
    alist.append(0)

mylist = [1, 2, 3]
change_list(mylist)
print(mylist)
```

```
[1, 2, 3, 0]
```

In [2]:
```
def change_list(alist):
    '''Takes a list and returns another list of the same length
    that looks like [0, 0, 0, ...].'''
    alist = [0]*len(alist)  # Creates a new local reference for alist

mylist = [1, 2, 3]
change_list(mylist)
print(mylist)
```

```
[1, 2, 3]
```

In [3]:
```
# Exercise: Rewrite the function definition and call above to accomplish what the
 function intends to do
```

Using Docstrings (String Literals) to Specify Functions

In [4]:
```
def f(x, y):
    '''Demonstrates the importance of providing specification for functions.
    Assumes x and y any type.
    Returns nothing.'''
    pass

help(f)
```

```
Help on function f in module __main__:

f(x, y)
    Demonstrates the importance of providing specification for functions.
    Assumes x and y any type.
    Returns nothing.
```

Using Functions Instead of Copy-Pasting Code

In [5]:
```
# Consider the following code:

# Print the name and profession of famous dead scientists:
print('Alan Turing was a mathematician.')
print('Richard Feynman was a physicist.')
print('Marie Curie was a chemist.')
print('Charles Darwin was a biologist.')
print('Ada Lovelace was a mathematician.')
print('Werner Heisenberg was a physicist.')
```

```
Alan Turing was a mathematician.
Richard Feynman was a physicist.
Marie Curie was a chemist.
Charles Darwin was a biologist.
Ada Lovelace was a mathematician.
Werner Heisenberg was a physicist.
```

In [6]:
```
# Exercise: Rewrite the code using a function and a suitable data structure.
```

Using Functions For General Cases

In [7]:
```
# Exercise: Write a Python function that checks if a string is a palindrome.
# A palindrome is a word or a phrase that reads the same backward as forward.
# For example, redder, nurses run, dad...
```

Using Functions to Improve Legibility

In [8]:
```
# Consider the following code:

# You are given two points in 2-D space
x = (1, 1)
y = (5, 4)

# Calculate the area of the circle if one of the points is the circle center and t
he other is on the perimeter
# and then calculate the side of the square with the same area
r_sq = (x[0] - y[0])**2 + (x[1] - y[1])**2
area = 3.14*r_sq
sq_side = area**0.5
print(sq_side)
```

```
8.860022573334675
```

In [9]:
```
# Exercise: Rewrite the code below using functions to make it easier to read.
```

Using Functions Inside List Comprehensions

In [10]:
```
def square_half(x):
    '''Assumes x is numeric. Estimates the square of x/2.'''
    return (x/2)**2

lst = [square_half(i) for i in range(10)]
print(lst)
```

```
[0.0, 0.25, 1.0, 2.25, 4.0, 6.25, 9.0, 12.25, 16.0, 20.25]
```

In [11]:
```
# Exercise: Using a function and a list comprehension, create a new list that has
 the numbers from
# testlist if they are positive and None otherwise

testlist = [-1, 0, 2, 178, -17.2, 12, -2, -3, 12]
```

In [12]:
```
# Exercise: Using a function and a list comprehension, create a new list that incl
udes the result
# from dividing each number from testlist1 by the corresponding number in testlist
2;
# For the cases when the divisor is 0, the new list should include None

testlist1 = [-1, 0, 2, 178, -17.2, 12, -2, -3, 12]
testlist2 = [0, 5, 0, 2, 12, 0.5, 0, 0.25, 0]
```