FEU Institute of Technology

(INTEGRATIVE PROGRAMMING AND TECHNOLOGIES)

EXERCISE

5

(Working with Functions)

Guillermo Justine ROme	Name of Professor
Data Performed	Date Submitted

Functions

Just like a value can be associated with a name, a piece of logic can also be associated with a name by defining a function.

```
>>> def square(x):
...     return x * x
...
>>> square(5)
25
```

The body of the function is indented. Indentation is the Python's way of grouping statements.

The `...` is the secondary prompt, which the Python interpreter uses to denote that it is expecting some more input.

The functions can be used in any expressions.

```
>>> square(2) + square(3)
13
>>> square(square(3))
81
```

Existing functions can be used in creating new functions.

```
>>> def sum_of_squares(x, y):
...     return square(x) + square(y)
...
>>> sum_of_squares(2, 3)
13
```

It is important to understand, the scope of the variables used in functions.

Lets look at an example.

```
x = 0
y = 0
def incr(x):
    y = x + 1
    return y
incr(5)
print x, y
```

Variables assigned in a function, including the arguments are called the local variables to the function. The variables defined in the top-level are called global variables.

Changing the values of `x` and `y` inside the function `incr` won't effect the values of global `x` and `y`.

But, we can use the values of the global variables.

```
pi = 3.14
def area(r):
    return pi * r * r
```

When Python sees use of a variable not defined locally, it tries to find a global variable with that name.

However, you have to explicitly declare a variable as `global` to modify it.

```
numcalls = 0
def square(x):
    global numcalls
    numcalls = numcalls + 1
    return x * x
```

Problem 7: How many multiplications are performed when each of the following lines of code is executed?

```
print square(5)
print square(2*5)
```

Problem 8: What will be the output of the following program?

```
x = 1
def f():
    return x
print x
print f()
```

Problem 9: What will be the output of the following program?

```
x = 1
def f():
    x = 2
    return x
print x
print f()
print x
```

Problem 10: What will be the output of the following program?

```
x = 1
def f():
        y = x
        x = 2
        return x + y
print x
print f()
print x
```

Problem 11: What will be the output of the following program?

```
x = 2
def f(a):
    x = a * a
    return x
y = f(3)
print x, y
```

Functions can be called with keyword arguments.

```
>>> def difference(x, y):
...     return x - y
...
>>> difference(5, 2)
3
>>> difference(x=5, y=2)
3
>>> difference(5, y=2)
3
>>> difference(y=2, x=5)
3
```

And some arguments can have default values.

```
>>> def increment(x, amount=1):
...     return x + amount
...
>>> increment(10)
11
>>> increment(10, 5)
15
>>> increment(10, amount=2)
12
```

There is another way of creating functions, using the `lambda` operator.

```
>>> cube = lambda x: x ** 3
>>> fxy(cube, 2, 3)
35
>>> fxy(lambda x: x ** 3, 2, 3)
35
```

Notice that unlike function defination, lambda doesn't need a `return`. The body of the `lambda` is a single expression.

The `lambda` operator becomes handy when writing small functions to be passed as arguments etc. We'll see more of it as we get into solving more serious problems.

Built-in Functions

Python provides some useful built-in functions.

```
>>> min(2, 3)
2
>>> max(3, 4)
4
```

The built-in function `len` computes length of a string.

```
>>> len("helloworld")
10
```

The built-in function `int` converts string to ingeter and built-in function `str` converts integers and other type of objects to strings.

```
>>> int("50")
50
>>> str(123)
"123"
```

Problem 12: Write a function `count_digits` to find number of digits in the given number.

```
>>> count_digits(5)
1
>>> count_digits(12345)
5
```

Problem 8

```
x = 1
def f():
    return x
print (x)
print (f())
```

```
YTHON\ACTIVITI
1
1

c:\Users\Justi
```

Problem 9

```
x = 1
def f():
    x=2
    return x
print (x)
print (f())
print (x)
```

```
1
2
1

C:\Users\J
```

Problem 10

```
# NUMBER 10
x = 1
def f():
    y=x
    x=2
    return x
print (x)
print (f())
print (x)
```

```
  File "c:/Users/Justine Guillermo/Google Drive/PROG/PYTHON/ACTIVITIES/lab5.p
y", line 27, in <module>
    print (f())
  File "c:/Users/Justine Guillermo/Google Drive/PROG/PYTHON/ACTIVITIES/lab5.p
y", line 23, in f
    y=x
UnboundLocalError: local variable 'x' referenced before assignment
```

Problem 11

```
# NUMBER 11
x=2
def f(a):
    x=a*a
    return x
y=f(3)
print (x, y)
```

```
"c:/Users/J
2 9

C:\Users\Ju
```

Problem 12

```
def count_digits(n):
    ctr=0
    if (n>=10000):
        return 5
    elif (n>=1000):
        return 4
    elif (n>=100):
        return 3
    elif (n>=10):
        return 2
    elif (n>=1):
        return 1
print('number of digits',count_digits(5))
```

```
C:/Users/Justine Gui
number of digits 1
```

```
def count_digits(n):
    ctr=0
    if (n>=10000):
        return 5
    elif (n>=1000):
        return 4
    elif (n>=100):
        return 3
    elif (n>=10):
        return 2
    elif (n>=1):
        return 1
print('number of digits',count_digits(12345))
```

```
"c:/Users/Justine Guillerm
number of digits 5
```