Python 3 Workshop
Revised by Xiannong Meng based on the work of Profs Hyde and Peck
Spring 2019

Start **Idle3** either from the Linux (Windows, Mac) menu, or at the Linux command line.

1. The interactive prompt >>> is very useful for a few lines of a program, however, we soon want to enter a larger program. To enter a Python program, select **File->New Window** to open an editor window from inside the **Idle3** men. Then enter the following in the new window being careful with punctuation and indenting.

```
# Python program with a main function and big_number function
def big_number(n):  # n is input
    return 2**n     # 2**n is output

def main():
    x = int(input('Enter integer value: '))
    y = big_number(x)
    print('2 to the power of', x, 'is', y)
```

Save your file as `workshop01.py`.

To run your program select **Run->Run Module** in the editor window. Then type `main()` at the >>> prompt. Try 100 for input.

FUNCTION BASICS | *We'll dedicate a lecture to functions, but here are a couple of quick points:*

- We often have to "teach" python new commands by grouping together a bunch of existing commands
    - For example, "Spread Peanut Butter" really consists of "stick knife in peanut butter jar", "raise knife at angle to get peanut butter out", "lay part of knife with peanut butter flat on bread", "move knife in a motion parallel to the bread while gently pressing down", etc.
- To do that, we create a **function** – something that starts with `def`.
- The word that follows `def` is the function's name.
- Everything indented under that name belongs to the function. Whenever we type `main()` at the prompt, it calls the `main()` function which in turn calls all those functions, following the logic you put in.
- A function can have input and output. Input is usually what goes in the parentheses of the function. Output is usually what comes after the word `return`. We'll talk about this a lot more in the future.

The `input` statement prints the string `'Enter integer value: '` to the screen then waits for the user to enter a value and press Return key. The value returned is a string and the built-in function `int()` converts the value to an int.

You can also feed multiple values into a function (this is how you input into a function). For example:

```
def add_two_numbers(a, b):
```

```
    return a + b

def main1():
    a = int( input('Enter integer value: ') )
    b = int( input('Enter integer value: ') )
    sum = add_two_numbers(a, b)
    print('The sum is ', sum)
```

Problem 1: Write a `main1()` function in the same file (`workshop1.py`) that inputs the temperature of the day in Fahrenheit as a float number and converts it to Celsius. The conversion is done by the following formula Celsius = (Fahrenheit – 32) * 5 / 9. It then prints a message something similar to the following, if the input is 29.

```
29 degree Fahrenheit is -1.67 degree Celsius.
```

The decimal number printed by your program may see many more digits after the decimal point. We will handle this issue later. Try your program with different inputs.

2. Enter the following program and run it.

```
import random

def main2():
    quote = input('Type in your favorite quote : ')
    print('Hello, this is your quote : "' + quote + '"')
    leng = len(quote)
    if leng % 2 == 0: # even
        half = leng // 2
    else:
        half = (leng + 1) // 2
    pick = random.randint(0,1) # a random int either 0 or 1
    if pick == 0:
        first = quote[0:half]
        second = quote[half:]
    else:
        first = quote[half:]
        second = quote[0:half]
    print('Your quote has a length of ', leng, ' random pick ', pick)
    print('I can reconstruct your quote as "' + first + second + '"')
```

This program has several features.
First, the `import random` line allows the program to use functions from a **standard library** called `random`. A standard library in Python consists of a number of helpful, well-designed functions that prevent you from recreating them yourself.

We use the `random` library's `randint(x, y)` function which generates a random number in the range of `x` and `y`, inclusive.

Second, a string of character can be sliced into portions, just like Python lists. The operation
`first = quote[0:half]`

means that the variable first receives the portion of the quote from location 0 through half-1, while
`second = quote[half:]`
means that the variable second receives the portion of the quote from half all the way to the end.

Third, the condition `leng % 2 == 0` is to check if a value `leng` is multiple of `2`. Modulo operation `x % n` is first dividing `n` into `x`, then taking the remainder. If the remainder is 0, `x` is multiple of `n`.

The program must have a colon at the end of the "if" and "else" lines and you must consistently indent the parts under them.

Run the program a few times with your favorite quotes, or any text strings to observe the behavior of the program. Make sure you understand what is going on.

Problem 2: Rewrite the program to ask the user to enter a string that has length of multiple of 3, that is 3, 6, 9, 12, … If the length of the quote is not multiple of 3, print a statement indicating we can't process the quote. If the length of the quote is multiple of 3, generate a random number that is either 0, 1, or 2. Based on the random value, you reconstruct the string in three different ways of connecting the string using each of the three partial strings.

3. In the next problem, we are going to use **nested if statements**. It means that you can put if statements inside of other if statements. For example, look at this function that determines whether you can add a class:

```
def add_a_class(adviser_response, room_in_class):
    if adviser_response == 'yes':
    # statement below is only run if the adviser_response is yes
        if room_in_class > 0:
            print('you successfully added the class.')
        else:
            print('sorry, there is no room.')
    else:
        print('sorry, you need permission from your adviser.')
```

Now that you have multiple functions in your file workshop1.py, you can comment out the calls to the function `main(), main1(), main2()` and the likes. You can run the program, then type the following at the Python prompt to get sense how the function `add_a_class()` behaves.

```
>>>add_a_class('yes', 3)
>>>add_a_class('yes', 0)
>>>add_a_class('no', 2)
```

If you want to call other functions you wrote, you can simply call its name, e.g.,

```
>>> main1()
>>> main2()
```

Problem 3: Write `a main3()` function that inputs the number of toppings (1, 2, or 3) and the size (`'large'` or `'small'`) of a pizza the customer orders and prints out the price depending on the below table. The number of toppings is an `int` and the size is a string. Note you will need to use nested `if` statements, i.e., `if` statements inside of `if` statements.

Size\Toppings	1	2	3
small	$8.00	$8.50	$8.75
large	10.00	$10.60	11.00

4. Using Python's built-in help. Use `dir(module_name)` to see a listing of all the functions in a module (Python library). Try the following at the `>>>` interactive prompt.

```
>>> import random   # import the module random
>>> dir(random)
```

Use `help(module_name)` to see the manual page on a module. Try

```
>>> help(random)
```

Use `help(module_name.function_name)` to see a short description of the function. Try

```
>>> help(random.choice)
```

You may use `dir()` and `help()` on a variable that has been assigned a value. Try

```
>>> s2 = 'Bison'
>>> dir(s2)
>>> help(s2.lower)
```