Defining and Working wit Functions

May 17, 2024

… to the only wise God!!!

1 Functions (Re-usable Statements)

Sometimes, you write a sequence of statements to perform a task, and you realize you often perform that task; may be in the same program, or in other programs.

You don't want to keep writing (or copying) same sequence of statements anytime you need to perform such task.

You may give the sequence of statements a *cute* name; so that anytime that task is to be performed, you simply call that name and the entire sequence of statements in the code get executed!

Just like you assign a value to a name to make a variable; you assign a sequence of statements to a name make a function.

Such a named sequence of statements is known as function!

Benefits of writing functions:

```
-> Saves typing efforts
-> Reduces the risk of mistakes
-> Help to organize your codes
-> Makes your code more readable
```

Take note, functions (named sequence of statements) are objects just as variables (named values) are objects.

The installed (base)Python itself thrives on a lot of in-built functions such as print, input, range, min, max, len, float, int, str etc.

Functions created (defined) by users (i.e. not built into Python during installation) are known as user-defined functions.

1.1 Function Definition

Function Header: The first line (*header*) of a function definition starts with the keyword *def*, followed by the name of the function (as chosen by you), followed by a comma-separated list of function argument(s) enclosed in parenthesis, and finally, a colon.

If the function requires no arguments, an empty parenthesis () is typed.

Example: Function to compute stock tank oil in place (STOIIP) in an oil reservoir

$$N = \frac{7758Ah\phi(1 - S_{wi})}{B_{oi}}$$

[1]:
```
# stoiip: name chosen for the function

def stoiip(area, thickness, poro, sw, boi):      # header
    N = (7758*area*thickness*poro*(1-sw))/boi    # body begins on this line
    N = round(N,2)
    return N                                     # body ends here
```

Function Body: The sequence of statements (to be executed whenever the function is called) is known as the *body* of the function, and is written in subsequent indented lines after the function's header.

Function Return Value: A statement specifying the value(s) to be returned by the function must be included with the keyword *return*

Function Arguments: The arguments of a function are the values that that it would need to execute its sequence of statements when called. When defining a function, placeholders (variable names with no values assigned) corresponding to these arguments are named, listed and used in expressions.

When the function is called, actual values for the arguments are specified (more on this, soon).

1.1.1 Defining functions with default arguments

In some cases, some arguments may have default values – a standard (often-used) value, to be used if value is not provided in the function call.

Such default value is to be assigned to the concerned argument name at the point of defining the function.

All such default arguments should only be listed after all un-defaulted arguments have been listed.

Example: a function to compute the density of a real gas (in lb/ft3)

$$\rho_g = \frac{2.7P\gamma_g}{zT}$$

While the equation above works for gas density at any given values of pressure, P and temperature, T; often, engineers want to compute gas density specifically at standard values of pressure ($14.7psia$) and temperature (520^0 $Rankine$). At these values, the z-factor, z also takes a standard value of 1.0.

In this example, the function might be defined to accommodate these default values.

However, the user may still specify other values for these default arguments when calling the function (more on this, soon).

[2]:
```
def gas_density(gravity, pressure = 14.7, temperature = 520, z = 1):
    density = (2.70*pressure*gravity)/(z*temperature)
    return round(density, 4)
```

[ ]:
```

```