Introduction to Programming: 2nd lesson – Functions

Basics of function:

# Define the function

def add\_three(input\_var):

output\_var = input\_var + 3

return output\_var

Headers:

* Defines the name of the function and its arguments.
* Every function header begins with def, which informs Python that the function will be defined.
* The argument is the name of the variable that will be used as input to the function, which always enclosed in parentheses that appear immediately after the name of the function.
* For every function, the parentheses enclosing the function arguments must be followed by a colon (:).

Body:

* Specifies the work that function operates.
* The function operators work by running all the indented lines from top to bottom:
  + Inputs the argument as input.
  + The function creates a new variable with the calculation assigned.
  + Then, the final line of code, known as return statement, just returns the value in the calculation assigned as the function’s output.

Running a function:

# Run the function with 10 as input

new\_number = add\_three(10)

# Check that the value is 13, as expected

print(new\_number)

13

Naming functions:

Only input lowercase letters with words separated by underscores (\_) instead of whitespaces ( ).

Complex calculation example:

Say you are helping a friend to calculate their weekly paycheck after taxes.

* They're in a 12% tax bracket (in other words, 12% of their salary is taken for taxes, and they only take home 88%), and
* They're paid hourly, at a rate of $15/hour.

The function below calculates the paycheck based on the number of hours worked. The function is more complicated than with the first example, because the function has more lines of code and comments. Similar to the example above, the function has a single argument (num hours). In the function body, we:

* Use the value for num\_hours to specify the value for a new variable pay\_pretax.
* Use the value of pay\_pretax to specify the value for a new variable pay\_aftertax.
* Return the value of the pay\_aftertax variable.

def get\_pay(num\_hours):

# Pre-tax pay, based on receiving $15/hour

pay\_pretax = num\_hours \* 15

# After-tax pay, based on being in 12% tax bracket

pay\_aftertax = pay\_pretax \* (1 - .12)

return pay\_aftertax

We call this function the same way we called the first function. The next code cell calculates the paycheck, based on working 40 hours. (After taxes, it is $528.)

# Calculate pay based on working 40 hours

pay\_fulltime = get\_pay(40)

print(pay\_fulltime)

528.0

To quickly calculate pay based on a different number of hours worked, you need to supply the function with a different number. For instance, say your friend works 32 hours. (Then, they get $422.40.)

pay\_parttime = get\_pay(32)

print(pay\_parttime)

422.4

Functions with multiple arguments:

def get\_pay\_with\_more\_inputs(num\_hours, hourly\_wage, tax\_bracket):

# Pre-tax pay

pay\_pretax = num\_hours \* hourly\_wage

# After-tax pay

pay\_aftertax = pay\_pretax \* (1 - tax\_bracket)

return pay\_aftertax

In the code cell below, we calculate the pay after taxes for someone who works 40 hours, makes $24/hour, and is in a 22% tax bracket.

higher\_pay\_aftertax = get\_pay\_with­\_more\_inputs(40, 24, .22)

print(higher\_pay\_aftertax)

748.8000000000001

The following code cell gives the same result as when we ran get\_pay(40), because hourly\_wage is set to 15, and tax\_bracket is set to 12%, which lines up with how we designed get\_pay.

same\_pay\_fulltime = get\_pay\_with­\_more\_inputs(40, 15, .12)

print(same\_pay\_fulltime)

528.0

Functions with no arguments:

# Define the function with no arguments and with no return

def print\_hello():

print(“Hello, you!”)

print(“Good morning!”)

# Call the function

print\_hello()

Hello, you!

Good morning!