Functions

A grouping of predefined statements for repeated operations.

A ***function*** is a named series of statements.

A ***function definition*** consists of the function's name and a block of statements.

A ***function call*** is an invocation of the function's name, causing the function's statements to execute.



A function may return one value using a ***return statement***.



def compute\_square(num\_to\_square):



return num\_to\_square \* num\_to\_square



num\_squared = compute\_square(7)



print(f'7 squared is {num\_squared}')



* 7 squared is 49



a = int(input())



def compute\_square(a):



b = a \*\* 2



return b



print(f'{a} squared is {compute\_square(a)}')



* 7 squared is 49



A function can return only one item, not two or more (though a list or a tuple with multiple elements could be returned).

**A return statement** may appear at any point in a function, not just as the last statement. A function may also contain multiple return statements in different locations.

A parameter is like a variable definition. Upon entering the function, the parameter is bound to the argument object provided by the call, creating a shared reference to the object. Upon return, the parameter can no longer be used.

Call a function named calc\_calories() passing the value 21 as an argument.

calc\_calories(21)



An argument is an expression like 21, num\_calories, num\_calories + 20, etc.



A parameter is like a variable definition. It cannot be an expression.



userNum + 5

print(get\_birthday\_age(42), get\_birthday\_age(20))

a comma creates a space between the two values, so the entire output is 43 21.

A function may have multiple parameters separated by commas.

A function definition with no parameters must still have the parentheses, as in: def calc\_something():

(99, 44+5) is legit argument.

### Hierarchical function calls

A function's statements may include function calls, known as ***hierarchical function calls*** or ***nested function calls***.

### Printing from a function

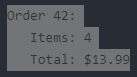
A function with no return statement is called a ***void function***, and such a function returns the value None.

old = int(input())

items = int(input())

price = float(input())

def print\_summary(old, items, price):

 print(f'Order {old}:')

print(f' Items: {items}')

print(f' Total: ${price:.2f}')

print\_summary(old, items, price)

### Dynamic and static typing

def add(x, y):

return x + y

1. add() function using two integer arguments, as in add(5, 7), which returns a value of 12.
2. add('Tora', 'Bora'), would concatenate the two strings and return 'ToraBora'

The function's behavior of adding together different types is a concept called ***polymorphism***.

( 5 \* 5)=25

( ‘x’ \* 5) = xxxxx

dynamic typing determines type of object in Python

C, C++, Java use static typing where each object must be defined

Polymorphism refers to how an operation depends on the involved object types. The behavior of an operator, such as + or \*, depends on the type of the operands.

***Modular development*** is the process of dividing a program into separate modules that can be developed and tested separately and integrated into a single program.

A general guideline (especially for beginner programmers) is that a function's definition usually shouldn't have more than about 30 lines of code, although this guideline is not a strict rule.

Redundant code can be replaced by multiple calls to one function.

Redundancy refers to writing the same code in multiple places. A programmer may want to write that code in a function, then call the function from multiple places in a program.

### Calling functions in expressions

y = square\_root(square\_root(16.0)) The inner square\_root(16.0) evaluates to 4.0. Then, the outer

square\_root(4.0) evaluates to 2.0, which y is then assigned with.

# Function stubs

To assist with the incremental development process, programmers commonly introduce ***function stubs***, which are function definitions whose statements haven't been written yet.

One approach is to use the ***pass*** keyword, which performs no operation except to act as a placeholder for a required statement.