Functions

- **Simpler code** - Easier to read and understand
- **Code reuse** - Reduces code duplication
- **Better testing** - Testing and debugging becomes simpler Faster development
- **Faster development** - Functions can be used for common tasks across programs
- **Easier to work in teams** - Assign different functions to different programmers

Basic Syntax:

```
def function_name(): #Function Head
    statement       #
    statement       # Function Block
    statement(s)    #
```

Calling Functions

```
# First define the function
def print_message():
    print("Hello World!")

# Call the print_message function
print_message()
```

Multiple Functions

```
# First define the main function
def main():
    print_message()

# Next define the print_message function
def print_message():
    print("Hello World!")

# Call the main function
main()
```

Arguments/Parameters

Argument – data passed into a function when called

Parameter – variable that receives data passed into a function

Basic Syntax

```
#Function definition
def function_name(variable1, variable2, etc):
    statement
    statement
    statement(s)

# Function call
function_name(value1, value2, etc.)
```

Example: (this is a stupid example tbh)

```
# First define the main function
def main():
    message = "Hello MIS 304"
    print_message(message)
    print(message)

# Next define the print_message function
def print_message(msg_to_display):
    print(msg_to_display)
    msg_to_display= "The end"

# Call the main function
main()
```

Return Values

Basic Syntax

```
def function_name():
    statement
    statement
    statement(s)
    return expression

#To call the function
variable = function_name()
```

Example:

```
def get_number():
    number = int(input("Please enter a number: "))
    return number

def calculate_square(value):
    square = value ** 2
    return square
```

Local vs Global Variables

Local variable – can only be “seen” within the function

Global variable – can be “seen” by all functions in a program

```
value = 100 #Global
def main():
    name = "Caryn Conley" #Local in main

def print_message():
    message = "Caryn Conley" #Local print_message
```

Using global variables is generally a **bad idea**:

- Global variables make debugging difficult
- Functions using global variables usually depend on those variables
- Global variables make programs hard to understand

Solution:

- Create local variables
- Pass them as arguments to other functions

Global Constants

Global constant – global name that references a value that cannot be changed during program execution

- Avoids problems of global variables

Global constants in Python

- Cannot create true global constants in Python
- Can be simulated using global variables

Coding standards for constants

- Name should be all caps (e.g.NUMBER)
- Declare globally at top of program