Python

WEEK 2

Function - Deep Dive

AI Academy

COMPUTER PROGRAMMING WITH PYTHON

Instructor - James E. Robinson, III

Teaching Assistant - Travis Martin

AI Academy

NC STATE

LIGHTNING REVIEW

- Variables
- Input / Output
- Expressions
- Functions
- Conditional Control
- Looping
- Data Types

TOPICS COVERED

- Logging
- Functions
  - Parameters / Arguments
  - Call by reference
  - Return
- Scope
- Decorators

LOGGING

DEBUGGING YOUR CODE WITHOUT MAKING CHANGES IN CODE

STARTING WITH LOGGING

The following code imports the logging module and configures the module.

```
# import logging module
import logging

# config logging module
logging.basicConfig(format='%(asctime)s - %(message)s',level=logging.DEBUG)
```

LOGGING

DEBUGGING YOUR CODE WITHOUT MAKING CHANGES IN CODE

The logging module defines levels that allow one to filter messages.
https://docs.python.org/3/howto/logging.html

Using those levels, we use log statements:

```
logging.debug("We got here!")
logging.info("Started doing computation.")
logging.warning("Hmm, that felt a little off.")
logging.error("That directory did not exist. Try again.")
logging.critical("Something bad happened. It's you, not me.")
```

FUNCTIONS

A BLOCK OF CODE THAT RUNS WHEN CALLED

Simple Functions

```
def main():
    print('I have a message for you.')
    message()
    print('Goodbye!')

def message():
    print('I am Arthur')
    print('King of the Britons.')

#calling main function
main()
```

OUTPUT :

```
I have a message for you.
I am Arthur
King of the Britons.
Goodbye!
```

Note :

A function may or may not return a value.

A function may accept several or no parameters.

A function is also considered as an expression if it returns a value.

A function can be accessed from within another function.

AI Academy

NC STATE

ARGUMENTS / PARAMETERS

VALUES THAT YOU FEED A FUNCTION

Example

```
def add(x=0, y=0):
    return x+y

print( add(5, 10) )
```

OUTPUT : 15

```
print( add(15) )
```

OUTPUT : 15

```
print( add() )
```

OUTPUT : 0

Note :
A parameter is the variable listed inside the parentheses in the function definition.
An argument is the value that are sent to the function when it is called.
They can be of any data type.
Arguments can be variables too.

CALL BY REFERENCE

A FUNCTION CALL IS CALLED BY REFERENCE

Example

```
a = 10

def twice(x):
    x = 2*x

twice(a)

print(a)
```

OUTPUT : 10

Note :
Python utilizes a system known as “Call by Object Reference”
The arguments passed to the functions are a reference to the original values. Any changes to those values in a function will *NOT* be reflected to the original variable.

RETURN VALUE

A RETURN STATEMENT RETURNS A VALUE FROM THE CALLED FUNCTION

Example

```
def divide(dividend, divisor):
    quotient = dividend // divisor
    remainder = dividend % divisor

    return quotient , remainder
```

Note :

A return value can be any data type

A function can return multiple values

A return statement terminates the flow of the function

A return value can also be boolean

A missing or empty return statement results in a *None* value of type *NoneType* being returned

SCOPE

SCOPE RULES DETERMINE THE VISIBILITY OF NAMES AND VARIABLES

The LEGB rule
It is the order in which Python tries to find a name or variable:

1. **L**ocal (or function)
2. **E**nclosing (or nonlocal)
3. **G**lobal (or module, with the default being *__main__*)
4. **B**uilt-in

Keywords
Used to help specify scope

- global
- nonlocal

Note :
Multiple variables can exist with same variable name but different scope.

SCOPE

Example

```
num = 100

def func1():
    num = 150
    print(f'Inside func1 - {num}')

def func2():
    global num
    num += 150
    print(f'Inside func2 - {num}')

print(f'Initially - {num}')
func1()
print(f'After func1 - {num}')
func2()
print(f'After func2 - {num}')
```

OUTPUT :

```
Initially - 100
Inside func1 - 150
After func1 - 100
Inside func2- 250
After func2 - 250
```

Note:

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

DECORATOR

A FUNCTION THAT TAKES A FUNCTION AS ARGUMENT AND RETURNS A FUNCTION

Syntax

```
@log_function_call
def basic():
    # something
    return
```

Note :
@log_function_call is the decorator.
A decorator is syntactic sugar that allows one to augment functionality of existing functions.

Explanation

```
def log_function_call(func):
    def enhanced():
        logging.info(f"starting - {func.__name__}")
        return func()
    return enhanced
def basic():
    return
basic = log_function_call(basic)
basic()
```

AI Academy

NC STATE

EXERCISE

Problem

Using the NC Lottery Cash 5 data, create a list, by play, of the average value of the 5 balls. Graph the result.

Dataset available on the NC Education Lottery website (see "Download all past draws here." link at bottom) or a snapshot is available for download from moodle in case gambling websites are blocked from your provider.

WEEK SUMMARY

- Learned to debug using logging
- Learned how functions are used
- Learned about arguments and call by reference
- Learned about scope of variables
- Learned how to use decorators and their purpose

THANK YOU

FOR ADDITIONAL QUERIES OR DOUBTS CONTACT:
jerobins@ncsu.edu

AI Academy