CS1301-Intro to Computing

Day 3

Review

- Programs must be designed before they are written
  - Algorithm: set of well-defined logical steps that must be taken to perform a task
- Order of Operations: PEMDAS+R+L
- Division: float - /, floor division - //, modulus - %
- The development environment for the Python interpreter is IDLE.
  - the debugger is Debug Control, accessed from the Shell

Functions

- A function is a named sequence of statements that belong together.
  - used to organize programs into chunks.
  - can be any number of statements inside the function
  - names follow the variable name syntax
  - A header line which begins with the keyword **def** and ends with a colon.
  - A body consisting of one or more Python statements, each indented the same amount — the Python style guide recommends 4 spaces — from the header line.

Functions – Built in

- print()
  - displays a value on the screen.

```
>>> print("Hello World")
Hello World
```

- input()
  - getting input from the user to assign to a variable
  - Default result is a str

```
>>> n = input("Please enter your name: ")
Please enter your name: Kearse
>>> print(n)
Kearse
```

Functions – User defined

- Function definition has two parts
  - A **header** line which begins with the keyword **def** and ends with a colon **:**.
  - A **body** consisting of one or more Python statements, each indented (4 spaces) under the function header.

- syntax for a **function definition** is:

```
def NAME( PARAMETERS ):
    STATEMENTS
```

Function name

An identifier by which the function is called

Arguments

Contains a list of values passed to the function

```
def name(arguments):
    statement
    statement
    ...
    return value
```

Indentation

Function body must be indented

Function body

This is executed each time the function is called

Return value

Ends function call & sends data back to the program

Function - header

- A function definition must declare any input values (parameters) that are needed for it to perform its task.
  - Parameters are variables/expressions inside the parenthesis of the header

- The syntax of declaring a function with parameters
  - `def function_name(par1, par2, …, parN):`

- The syntax of declaring a function without parameters:
  - `def function_name():`

Function – void vs value-returning

- All **Python functions return** the value **None** (void), unless there is an explicit **return** statement with a **value** (value-returning).
  - Functions can only return one value

Function – void vs value-returning

- void
  - functions that return the value None

- value returning
  - functions with an explicit **return** statement with a value

Function – void

Python10.2.py

```
#define return function
def square(x):
    print(x*x)

print(square(4))
```

Run Python10.2

```
"C:\Users\DK\Desktop\Python
16
None
```

The function does not return anything. Hence output is None

Function - value-returning

example.py

square

```
1  def square(x):
2      return x*x
3
4  print(square(3))
5
```

return is used to get the output outside the function body

Debug I/O | Exceptions | Python Shell | Messages | OS Commands

Debug I/O (stdin, stdout, stderr) appears below

```
9
```

Line 1 Col 0 - [User]

Function - calling

- A function definition does nothing by itself.
- To use a function it must be called.
- The order of arguments should match the order or parameters

- The syntax of calling a function with arguments:
  `function_name(arg1, arg2, arg3, …, argN)`

- The syntax of calling a function without arguments:
  `function_name()`

Function - calling

After executing the body of the greet() function, control jumps back to the point where it left off and resumes the execution from there

```
import datetime

def greet():
      print("Hello !")
      print("Today is", datetime.datetime.now())


print("Before calling greet()")
greet()
print("After calling greet()")
```

When greet() function is called control jumps to the body of greet() function

After Class

1. Review lecture notes and code from today
2. Begin working on Homework 1
3. Read textbook Chapter 5 and Chapter 6
4. Practice Python

Time to program

1. Launch Canvas
   - Select this course KEARSE- CS1301-A/C
   - Click Files - Lecture notes - **03notes.py, 03code.py**
   - **download the files to your CS1301 folder**
2. Launch IDLE
   - Click on the File - Open
   - Navigate to your CS1301 folder
   - Click on the downloaded file for today
   - Click the Open button