```
#-----------------------------------------------------------------
# File prime.py finds the first 100 prime numbers
#-----------------------------------------------------------------
# Author: Thomas R Turner
# E-Mail: trturner@uco.edu
# Date:   January, 2019
#-----------------------------------------------------------------
from math import sqrt
def isPrime(c) :
   factor=3
   maxfactor=sqrt(c)
   while factor<=maxfactor :
      if c%factor==0 :
         return False
      factor=factor+2
   return True
primecount=100
print("%5d" % (2),end="")
count=1
candidate=3
while count<primecount :
   if isPrime(candidate) :
      count=count+1
      print("%5d" % candidate,end="")
      if count%10==0 :
         print()
   candidate=candidate+2
```

File ~tt/cs4023/python/prime.py

```
$ python3 prime.py
    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
   73   79   83   89   97  101  103  107  109  113
  127  131  137  139  149  151  157  163  167  173
  179  181  191  193  197  199  211  223  227  229
  233  239  241  251  257  263  269  271  277  281
  283  293  307  311  313  317  331  337  347  349
  353  359  367  373  379  383  389  397  401  409
  419  421  431  433  439  443  449  457  461  463
  467  479  487  491  499  503  509  521  523  541
```

Execution of prime.py

Notes:

1. Blocks are indented
2. Subprograms begin with **def**
3. Semicolons are not used to terminate statements
4. Print statements
   4.1. Formatted similar to C write.
   4.2. The argument **end=””** suppresses the new line character.
   4.3. The general form is: **print(“%***format specification***” % (***variables* **|** *values***))**

```
#-----------------------------------------------
# Author: Thomas R Turner
# E-Mail: trturner@uco.edu
# Date:   April, 2019
#-----------------------------------------------
# Usage
# python3 args.py one two three four
#-----------------------------------------------
# Output
# one
# two
# three
# four
#-----------------------------------------------
# Copyright: April, 2019 by Thomas R Turner.
# Do not reproduce without permission from Thomas R Turner
#-----------------------------------------------
from sys import argv
def main() :
   for i in range(1,len(argv)):
      print(argv[i])
#-----------------------------------------------
# Start the program
#-----------------------------------------------
main()
```

File ~tt/cs4023/python/args.py

```
$ python3 args.py one two three four
one
two
three
four
```

Execution of args.py

```
#-----------------------------------------------------
# File readfile.py
# - opens a file
# - reads the file
# - prints the contents to the display
# - closes the file
#-----------------------------------------------------
# Author: Thomas R Turner
# E-Mail: trturner@uco.edu
# Date: April, 2019
#-----------------------------------------------------
infile=open("i01.dat","r")
line=infile.readline()
while line !="" :
   print(line)
   line=infile.readline()
infile.close()
```

File ~tt/cs4023/python/readfile.py

```
one, two, buckle my shoe
three, four, shut the door
five, six, pick up sticks
seven, eight, lay them straight
```

File ~tt/cs4023/ python/i01.dat

```
$ python3 readfile.py
one, two, buckle my shoe

three, four, shut the door

five, six, pick up sticks

seven, eight, lay them straight
```

Execution of readfile.py

```
#-----------------------------------------------------
# File files.py
# - opens a file
# - reads the file
# - writes the file
# - closes the file
#-----------------------------------------------------
# Author: Thomas R Turner
# E-Mail: trturner@uco.edu
# Date: April, 2019
#-----------------------------------------------------
infile=open("i01.dat","r")
outfile=open("o01.dat","w")
line=infile.readline()
while line !="" :
   outfile.write("%s" % (line))
   line=infile.readline()
outfile.close()
infile.close()
```

File ~tt/cs4023/python/readwrite.py

```
$ python3 readwrite.py
```

Execution of readwrite.py

```
one, two, buckle my shoe
three, four, shut the door
five, six, pick up sticks
seven, eight, lay them straight
```

File ~tt/cs4023/python/o01.dat

```
#-------------------------------------------------------
# File exception01.py explores exceptions and handlers
#-------------------------------------------------------
# Author: Cay Horstman
#-------------------------------------------------------
try :
   filename = input("Enter filename: ")
   infile = open(filename, "r")
except IOError :
   print("Error: " + filename + " could not be opened.")
```

File ~tt/cs4023/python/exception01.py

```
$ python3 exception01.py
Enter filename: zafira
Error: zafira could not be opened.
```

Execution of exception01.py

```
#-------------------------------------------------------
# File exception02.py explores exceptions and handlers
#-------------------------------------------------------
# Author: Cay Horstman
#-------------------------------------------------------
try :
   raise BufferError("Stack Overflow")
except BufferError as exception :
   print("Error: " + str(exception))
```

File ~tt/cs4023/python/exception02.py

```
$ python3 exception02.py
Error: Stack Overflow
```

Execution of exception02.py

```
#-------------------------------------------------------------
# File string.py exercises various string operations
#-------------------------------------------------------------
# Author: Thomas R Turner
# E-Mail: trturner@uco.edu
# Date: April, 2019
#-------------------------------------------------------------
# Copyright April, 2019 by Thomas R Turner
# Do not reproduce without permission from Thomas R Turner
#-------------------------------------------------------------
christianname="Niklaus"
sirname="Wirth"
wholename=christianname+" "+sirname
print(wholename)
for letter in wholename :
   print(letter)
```

File ~tt/cs4023/python/string.py

```
tt@CS:~/cs4023/python$ python3 string.py
Niklaus Wirth
N
i
k
l
a
u
s

W
i
r
t
h
```

Execution of string.py

```
#-------------------------------------------------------------
# The selectionSort function sorts a list using the selection sort algorithm
#-------------------------------------------------------------
# Author: Cay Horstmann
#-------------------------------------------------------------
# Sorts a list, using selection sort
#-------------------------------------------------------------
def selectionSort(values) :
   for i in range(len(values)) :
      minPos = minimumPosition(values,i)
      temp = values[minPos]
      values[minPos] = values[i]
      values[i] = temp
#-------------------------------------------------------------
# Finds the smallest element in a tailrange of the list
#-------------------------------------------------------------
def minimumPosition(values,start) :
   minPos = start
   for i in range(start+1,len(values)) :
      if values[i] < values[minPos] :
         minPos = i

   return minPos
#-------------------------------------------------------------
```

File ~tt/cs4023/python/selectionsort.py

```
#-------------------------------------------------------
# This program demonstrates the selection sort algorithm
# by sorting a list that is filled with random numbers
#-------------------------------------------------------
# Author: Cay Horstmann
#-------------------------------------------------------
from random import randint
from selectionsort import selectionSort

n=20
values = []
for i in range(n) :
   values.append(randint(1,100))
print(values)
selectionSort(values)
print(values)
```

File ~tt/cs4023/python/selectiondemo.py

```
$ python3 selectiondemo.py
[60, 17, 11, 52, 15, 3, 56, 61, 98, 27, 98, 18, 11, 78, 43, 88, 3, 9, 66, 38]
[3, 3, 9, 11, 11, 15, 17, 18, 27, 38, 43, 52, 56, 60, 61, 66, 78, 88, 98, 98]
```

Execution of selectiondemo.py

```
#-------------------------------------------------------
# Models a tally counter whose value can be incremented, viewed, or reset
#-------------------------------------------------------
# Author: Cay Horstmann
#-------------------------------------------------------
class Counter :
   # Gets the current value of the counter.
   # @return the current value
   #
   def getValue(self) :
      return self._value
   #
   # Advances the value of this counter by 1
   #
   def click(self) :
      self._value = self._value + 1
   #
   # Resets the value of this counter to 0.
   #
   def reset(self) :
      self._value = 0
```

File ~tt/cs4023/python/counter.py

```
#-------------------------------------------------------
# This program exercises class Counter
#-------------------------------------------------------
# Author: Cay Horstmann
from counter import Counter

tally = Counter()  # Invoke the constructor
tally.reset()
tally.click()
tally.click()

result = tally.getValue()
print("Value:",result)

tally.click()
result = tally.getValue()
print("Value:",result)
```

File ~tt/cs4023/python/counterdemo.py

```
$ python3 counterdemo.py
Value: 2
Value: 3
```

Execution of counterdemo.py

Notes:

1. By convention, instance variables in Python start with an underscore to indicate that they should be private. Example: ***_value***
2. Member functions are called ***method***s
3. The first parameter of a method is called ***self***.
4. Instance variables are not explicitly declared. Instance variables are defined when a value is assigned to an instance variable.

```
def reset(self)
    self._value=0
```