

CIS 125 Principles of Programming Logic

Functions/Modules

There are two main types of functions:

1. **Built-in function** – these already exist; they are built-into the programming language.
2. **User-defined function** (or modules or procedures) – these do not already exist; you create them for a specific purpose.

Both types share some common characteristics:

- Both modularize a specific task of a program.
- Both are created so code can be re-used (write once, maintain in one location, run many times)
- Both are called - you call/run them from the main, or some part of the program.
- Both can have data passed into them and out of them (or no data passed in or out).

Examples of **built-in functions** (sometimes called a method):

- Math: pi, round, sin, tangent, random number (useful for games, simulations, statistics, computer security/encryption), etc.
- Formatting: change 6543.9 to 6,543.90
- String: length (find length of string), count how many of a certain character is in a paragraph, convert string to all upper case or initial cupper case, extract a portion of a string, replace certain characters in a string with other characters, etc.
- Many languages have 500+ built-in library functions.

Python – Built-In / Library Function Examples

Nearly every language has a large assortment of built-in functions - functions with a specific purpose you can call to perform a task without having to code the details yourself. Built-in functions can be very useful. Python has the following standard built-in library functions.

abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Python has many of additional libraries (also referred to as services and modules) that add hundreds more built-in functions. Below are a few of these libraries/modules/services.

String	Numbers	SSL
Re (regular expression)	Fileinput	Email
Codecs	Random	JSON
Datetime	OS.path	HTMLParser
Calender	Zipfile	HTMLlib
Array	Csv	XM.dom
Pprint	Hashlib	cmd
Fmmatch	Crypt	MacOS

Just the string library has 12 functions within it. The math library has over 40.

Built-In Function Examples

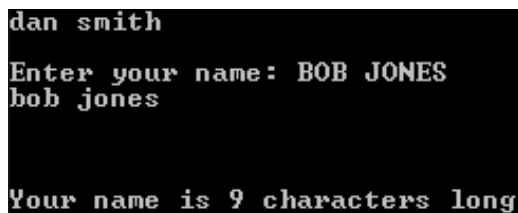
Example: Python string functions

```
name1 = "DAN SMITH"
print(name1.lower(), "\n")

name2 = input('Enter your name: ').lower()
print(name2, "\n\n\n")

print("Your name is", len(name2), "characters long")
```

Output:

A screenshot of a terminal window showing the output of the Python string function example. The output is as follows:
dan smith

Enter your name: BOB JONES
bob jones

Your name is 9 characters long

Example: Time function (waits two seconds)

```
import time
print("Thinking...")
time.sleep(2)
print("Done")
```

In the example above **time** is the **library** being imported and **sleep** is the **function** be used inside the library. **2** is a value being passed into the function.

Example: Calendar function (2015 and 10 are being passed into the function and the calendar is being passed out)

```
import calendar
cal = calendar.month(2015, 10)
print ("Here is the calendar:")
print (cal)
```

Here is the calendar:

```
October 2015
Mo Tu We Th Fr Sa Su
                1 2 3 4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Example: Math functions

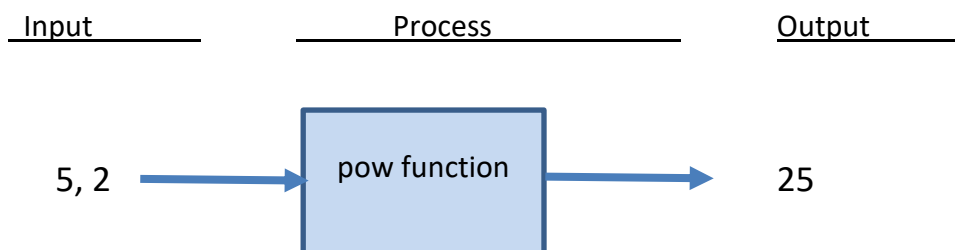
```
import math

total1 = math.pow(5,2)
total2 = math.sqrt(36)
total3 = math.pi * 2.5

print ("Total1 = ", total1)
print ("Total2 = ", total2)
print ("Total3 = ", total3)
```

Output:

```
Total1 = 25.0
Total2 = 6.0
Total3 = 7.853981633974483
```



Example: Displaying current date

```
import datetime
print ("Current Date: " ,datetime.datetime.now().strftime("%m/%d/%y"))
```

Example: random functions

```
import random

die1 = random.randrange(1,7)
die2 = random.randrange(1,7)
print ("Dice 1 roll: ", die1)

print ("Dice 2 roll: ", die2)
card_deck = ['2C', '2H', '2S', '2D', '3C', '3H', '3S', '3D']
random.shuffle(card_deck)
print ("Top card on deck: ", card_deck[0])
```

Example: replace function

```
text = "The quick brown fox jumps over the lazy dog."
text = text.replace("fox", "dog")
text = text.replace("lazy dog", "lazy fox")
print (text)
```

Output:

The quick brown dog jumps over the lazy fox.

Example: join function

```
list1 = ["The", "quick", "fox", "runs."]
sent = " ".join(list1)
print (sent)
```

Output:

The quick fox runs.

Example: Using math library to perform square root calculation

```
import math
feet = int(input('Enter height in feet at which you will drop object: '))
time2 = 2 * feet / 32.1522
time = math.sqrt(time2)
```

Example: Open external Operating System (OS) files from Python (there are three other methods)

```
import os
os.startfile("diagnose_power.pdf")
```

Example: Miscellaneous Built-In Functions

```
import math, random, datetime, os, calendar, time

name = input("Please enter your full name: ")
num1 = float(input("Please enter number for calculations: "))

areaC = math.pi * math.pow(num1,2)
squareRoot = math.sqrt(num1)
ran1 = random.randrange(1,num1)
cal = calendar.month(2017, 1)

os.system('cls')
print("The area of the circle is: %.2f" % areaC + ".")
print("Square root of number is: ", str(squareRoot) + ".")
print("Random number between 1 and", int(num1), "is: ", str(ran1) + ".")
print ("Current Date: " ,datetime.datetime.now().strftime("%m/%d/%y."))
print ("Your name is", len(name), "characters long.")
print (cal)
print ("Thinking...")
time.sleep(2)
print("Done")

card_deck = ['2C', '2H', '2S', '2D', '3C', '3H', '3S', '3D']
random.shuffle(card_deck)
print ("Top card on deck: ", card_deck[0])

os.startfile("diagnose_power.pdf")
```

Output:

Please enter your full name: John Smith
Please enter number for calculations: 25

The area of the circle is: 1963.50.
Square root of number is: 5.0.
Random number between 1 and 25 is: 20.
Current Date: 01/21/17.
Your name is 10 characters long.

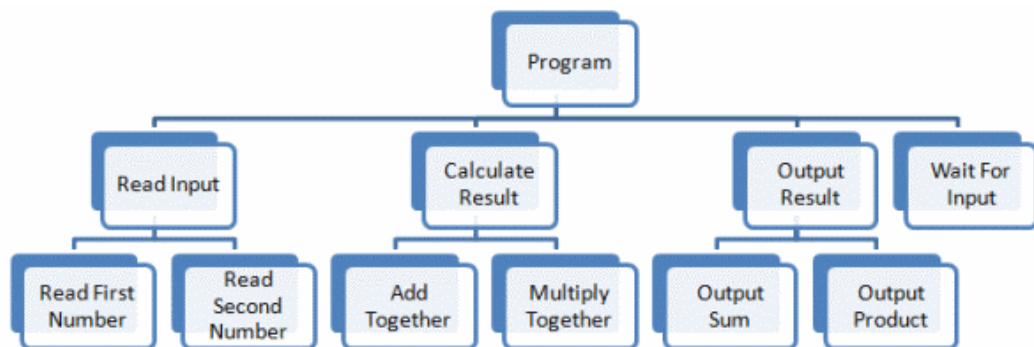
January 2017
Mo Tu We Th Fr Sa Su
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

Thinking....
Done
Top card on deck: 3C

User-Defined Functions

User-defined functions (or modules)

- A group of statements that exist within a program for the purpose of performing a specific task, e.g. you might have a module to: calculate overtime pay, calculate student GPA, display a menu etc.
- Allow for top-down design (or stepwise refinement)
 - Break program down into sub-components or sub-tasks, e.g. modules/functions
 - Determine if sub-tasks will have sub-tasks (decomposition)
 - This can be visually depicted with a “hierarchy chart”
 - Example from <http://www.multiwingspan.co.uk/vb08.php?page=proc>



- Benefits to functions:
 - Re-use code
 - Maintain re-used code in one location
 - Easier to read program
 - Team work and faster development – assign functions to different team members
 - Modular testing and isolate problems during testing
- Functions do not run until they are called. You define and name them first, then you can call/run them (if you need them).

Below are **four main ways a function may be used in Python** (used with pass by value model):

1. No data passed in, no data passed out
2. No data passed in, data passed out
3. Data passed in, no data passed out
4. Data passed, data passed out

Example: User-defined menu function (no data passed in; data being passed back to main)

```
def menu():
    print ("MAIN MENU")
    print ("-----")
    print ("1. Print pay checks")
    print ("2. Look up employee")
    print ("3. Change benefits")
    print ("4. Exit")
    choice = int(input("Choose: "))
    return choice

choice = menu()
print("You chose menu option", choice)
```

Example: Function to calculate power (data being passed in, data being passed back)

Functions in Python can return a value to the module/part of the program that called them.

```
def power3( arg1, arg2 ):
    total = arg1 ** arg2
    return total

total = power3( 10, 3 ) # Call power3 function
print("Total: ", total)
```

Output:

Total: 1000

Variable Scope

- Functions introduce the concept of scope.
- Variables can have either local or global scope (i.e. where the variable is available).
- Default scope is local – variables only available inside the function they are created in
- Global variables and global constants should be avoided, if possible – they make program harder to read and diagnose errors in
- Different functions could have can have local variables with the same name
- In Python, variables are immutable objects (value changed in function not reflected in main unless passed back; like “pass by value”). Lists in Python are mutable objects (value changed in function is reflected in the main without being passed back).

Example: Python program that produces syntax error

```
def power3( arg1, arg2 ):
    total = arg1 ** arg2

power3( 10, 3 ) # Call power3 function
print("Total: ", total)
```

In the above example, because the total variable was not passed back to the main part of the program, the main part of the program does not know what it is and cannot print it. total is only known inside the function.

Example: Example of local variable scope and data being passed back

```
def power3( arg1, arg2 ):
    total = arg1 ** arg2;    # Local variable.
    print ("Inside the function local total: ", total)
    return total

total = power3( 10, 3 )    # Call power3 function
print ("Value of total outside function: ", total)
```

Output:

```
Inside the function local total: 1000
Value of total outside function: 1000
```

Example: Example of global variable scope (avoid the use of this)

```
def power3( arg1, arg2 ):
    global total
    total = arg1 ** arg2;    # Local variable.
    print ("Inside the function local total: ", total)

power3( 10, 3 )    # Call power3 function
print ("Value of total outside function: ", total)
```

Output:

```
Inside the function local total: 1000
Value of total outside function: 1000
```


Example: Data being passed into function, no data being passed back

```
def power3( arg1, arg2 ):
    total = arg1 ** arg2;      # raise arg1 to the power of arg2
    print ("Inside the function local total: ", total)

power3(3,2)
```

Output:

Inside the function local total: 1000

Example: Function with no data in and no data passed out

```
def menu():
    print ("MAIN MENU")
    print ("-----")
    print ("1. Print pay checks")
    print ("2. Look up employee")
    print ("3. Change benefits")
    print ("4. Exit")

menu()
```

Example2: Function with no data in, but data being passed out

```
def menu():
    print ("MAIN MENU")
    print ("-----")
    print ("1. Print pay checks")
    print ("2. Look up employee")
    print ("3. Change benefits")
    print ("4. Exit")
    choice = int(input("Choose: "))
    return choice

choice = menu()
print("You chose menu option", choice)
```

Example Power Function with User Input

```
def power3( arg1, arg2 ):
    total = arg1 ** arg2
    print ("Inside the function local total: ", total)
    return total

power3( 10, 3 ) # Call power3 function

# With user input
num1 = int(input("Enter number to raise to power: "))
num2 = int(input("Enter number for power: "))
power3(num1, num2) # Call power3 function
```

Sample run:

```
Inside the function local total: 1000
Enter number to raise to power: 10
Enter number for power: 3
Inside the function local total: 1000
```

Example: Data passed in, no data passed back (string being passed in)

```
def printme(str):
    print (str)

# Main - function calls
printme("This is the first line to be printed.")
printme("This is the second line.")
```

Output:

```
This is the first line to
This is the second line.
```

Example: Data being passed into function, no data being passed back with keyword arguments

```
def display_employee( name, age ):
    print ("Name: ", name, "\t Age: ", age)

display_employee( age=42, name="Daniel Smith" )
display_employee( age=35, name="Tina Jones" )
```

Output:

```
Name: Daniel Smith    Age: 42
Name: Tina Jones      Age: 35
```

Next Examples: Tuition Calculator - One Program Five Ways

Example: Simple Tuition Calculator – No Modules

```
credits = int(input("How many credit hours will you be taking?"))
amount_owed = credits*100+50
print("Your total tuition amount is", amount_owed, "dollars.")
```

Example: Simple Tuition Calculator – Function with no data passed in and no data passed back

```
def calculateAmountOwed():
    credits = int(input("How many credit hours will you be taking?"))
    amount_owed = credits*100+50
    print("Your total tuition amount is", amount_owed, "dollars.")

calculateAmountOwed()
```

Example: Simple Tuition Calculator – Function with one variable passed in and no data passed back

```
def calculateAmountOwed(credits):
    amount_owed = credits*100+50
    print("Your total tuition amount is", amount_owed, "dollars.")

credits = int(input("How many credit hours will you be taking?"))
calculateAmountOwed(credits)
```

Example: Simple Tuition Calculator – Function with global variable and user input inside function

```
def calculateAmountOwed():
    global amount_owed
    credits = int(input("How many credit hours will you be taking?"))
    amount_owed = credits * 100 + 50

calculateAmountOwed()
print("Your total tuition amount is", amount_owed, "dollars.")
```

Example: Simple Tuition Calculator - Function with one variable passed in and one variable passed back

```
def calculateAmountOwed(credits):
    amount_owed = credits*100+50
    return amount_owed

credits = int(input("How many credit hours will you be taking?"))
amount_owed = calculateAmountOwed(credits)
print("Your total tuition amount is", amount_owed, "dollars.")
```

Example: Calling a function from within a function in Python (generally avoid or use sparingly; may make program difficult to read/debug)

```
def getMess():
    message = input("Enter your message: ")
    printMess(message)

def printMess(message):
    print(message)

getMess()
```

Output:

```
Enter your message: Hello
Hello
```

Example: Menu system with functions

```
menu_chosen = 0

# functions
def menu(menu_chosen):
    print("MAIN MENU")
    print("-----")
    print("1. Print pay checks")
    print("2. Look up employee")
    print("3. Change benefits")
    print("4. Exit")
    menu_chosen = int(input("Chose menu option: "))
    return menu_chosen

def print_pay():
    print("\nPRINT PAYCHECKS")
    print("-----")
    return

def employee_lookup():
    print("\nEMPLOYEE LOOKUP")
    print("-----")
    return

# main program
menu_chosen = menu(menu_chosen)

if menu_chosen == 1:
    print_pay()
elif menu_chosen == 2:
    employee_lookup()
```

Output:

```
MAIN MENU
-----
1. Print pay checks
2. Look up employee
3. Change benefits
4. Exit
Chose menu option: 1

PRINT PAYCHECKS
-----
```

Example: Global variable example (use should be minimized)

```
def addHours():
    global hours
    hours_more = int(input('How many more hours did you work this week? '))
    hours = hours + hours_more
    print ("You've worked ", hours, " hours")

hours = 0
hours = int(input('How many hours did you work so far this week? '))
addHours()
print ("You've worked ", hours, " hours")
```

Output:

```
How many hours did you work so far this week? 20
How many more hours did you work this week? 20
You've worked 40 hours
You've worked 40 hours
```

Example: Python function returning a value (alternative to previous example)

```
def addHours(hours):
    hours_more = int(input('How many more hours did you work this week? '))
    hours = hours + hours_more
    return hours

hours = 0
hours = int(input('How many hours did you work so far this week? '))
newhours = addHours(hours)
print ("You've worked ", newhours, " hours")
```

Example: Python function returning a value

```
def addHours(hours):
    hours_more = int(input('How many more hours did you work this week? '))
    hours = hours + hours_more
    return hours

hours = 0
hours = int(input('How many hours did you work so far this week? '))
newhours = addHours(hours)
print ("You've worked ", newhours, " hours")
```

Output:

```
How many hours did you work so far this week? 30
How many more hours did you work this week? 10
You've worked 40 hours
```

Example: Word length function

```
def wordCheck(word):  
    length = len(word)  
    print("Your word is", length, "characters long.")  
  
word = input("Enter a word: ")  
wordCheck(word)
```

Example: Word length function method #2

```
def wordCheck(word):  
    length = len(word)  
    return length  
  
word = input("Enter a word: ")  
length = wordCheck(word)  
print("Your word is", length, "characters long.")
```

Output of previous two examples:

```
Enter a word: William  
Your word is 7 characters long.
```

Example: Functions call from a print statement

```
def greeting(name):  
    a = "Hello " + name  
    return a  
  
print(greeting("William"))
```

Output:

```
Hello William
```

Example: Functions call from an input statement (produces same output as previous example)

```
def greeting(name):  
    print("Hello " + name)  
  
greeting(input("What is your name? "))
```

Example: Function with data passed in, no data passed back, run three times

```
def compForm(x,y):  
    result = (10 * x) + (5 * y)  
    print("Result:", result)  
  
compForm(2,5)  
compForm(3,4)  
x=5  
y=6  
compForm(x,y)
```

Output:

```
Result: 45  
Result: 50  
Result: 80
```

Functions can return more than one value. However, they will be returned as a tuple (tuples and lists will be discussed more detail in the array chapter).

Example: Word length function method #2

```
def combine():  
    x = 1  
    y = 2  
    z = 3  
    return x, y, z  
  
a = combine()  
print(a)  
for i in a:  
    print(i)
```

Output:

```
(1, 2, 3)  
1  
2  
3
```

Formatted Output of User-Defined Function Passing Two Variables in and Passing Two Variables Back

```
def calcPay(hourly_rate, hours_worked):
    DEDUCTIONS = .25
    gross_pay = hourly_rate * hours_worked
    net_pay = gross_pay * DEDUCTIONS
    return gross_pay, net_pay

hourly_rate = float(input("Please enter employee hourly rate: "))
hours_worked = float(input("Please enter employee hours worked: "))
gross_pay, net_pay = calcPay(hourly_rate, hours_worked)
print("Employee gross pay: $%.2f\t net pay: $%.2f" % (gross_pay, net_pay))
```

Output:

```
Please enter employee hourly rate: 40
Please enter employee hours worked: 25
Employee gross pay: $1000.00 net pay: $250.00
```

Another Set of Examples of Four Main Ways to Pass Data to Functions

Example: Meal Calculator

```
# No Data In, No Data Out
def billCalc():
    meal = float(input("Please enter your meal cost: "))
    tip = float(input("Please enter your tip amount (e.g. .15 for 15%): "))
    bill = meal + (meal * tip)
    print("Your total bill comes to: $%.2f" % bill)

billCalc()
```

Sample run:

```
Please enter your meal cost: 100
Please enter your tip amount (e.g. .15 for 15%): .15
Your total bill comes to: $115.00
```

Example: Meal Calculator

```
# Data In, No Data Out
def billCalc(meal, tip):
    bill = meal + (meal * tip)
    print("Your total bill comes to: $%.2f" % bill)

meal = float(input("Please enter your meal cost: "))
tip = float(input("Please enter your tip amount (e.g. .15 for 15%): "))
billCalc(meal, tip)
```

* Same output *

Example: Meal Calculator

```
# Data In, Data Out
def billCalc(meal, tip):
    bill = meal + (meal * tip)
    return bill

meal = float(input("Please enter your meal cost: "))
tip = float(input("Please enter your tip amount (e.g. .15 for 15%): "))
bill = billCalc(meal, tip)
print("Your total bill comes to: $%.2f" % bill)
```

* Same output *

Example: Meal Calculator

```
# No Data In, Data Out
def billCalc():
    meal = float(input("Please enter your meal cost: "))
    tip = float(input("Please enter your tip amount (e.g. .15 for 15%): "))
    bill = meal + (meal * tip)
    return bill

bill = billCalc()
print("Your total bill comes to: $%.2f" % bill)
```

* Same output *

Example: Function with data being passed in and data being passed out, called from print statement

```
def arith(x, y):
    total = x * (x + y + 2)
    return total

print("Total: ", arith(2, 2))
print("Total: ", arith(2, 3))
```

Output:

Total: 12

Total: 14

Another Program Written Five Ways

```
def calcPay():                                # Method 1: No data in, No data out
    DEDUCTIONS = .05
    TAX_RATE = .15
    hours = float(input("How many hours did you work this week? "))
    rate = float(input("What is your hourly pay rate? "))
    weekly_pay = (hours * rate) - ((hours * rate) * (DEDUCTIONS + TAX_RATE))
    print("Your weekly pay is: $", weekly_pay)

calcPay()
```

```
def calcPay():                                # Method 2: No data in, Data out
    DEDUCTIONS = .05
    TAX_RATE = .15
    hours = float(input("How many hours did you work this week? "))
    rate = float(input("What is your hourly pay rate? "))
    weekly_pay = (hours * rate) - ((hours * rate) * (DEDUCTIONS + TAX_RATE))
    return weekly_pay

weekly_pay = calcPay()
print("Your weekly pay is: $", weekly_pay)
```

```
# Method 3: Data in, No data out
def calcPay(DEDUCTIONS, TAX_RATE, hours, rate):
    weekly_pay = (hours * rate) - ((hours * rate) * (DEDUCTIONS + TAX_RATE))
    print("Your weekly pay is: $", weekly_pay)

DEDUCTIONS = .05
TAX_RATE = .15
hours = float(input("How many hours did you work this week? "))
rate = float(input("What is your hourly pay rate? "))
calcPay(DEDUCTIONS, TAX_RATE, hours, rate)
```

```
def calcPay(deductions, tax_rate):            # Method 4: Data in, Data out
    hours = float(input("How many hours did you work this week? "))
    rate = float(input("What is your hourly pay rate? "))
    weekly_pay = (hours * rate) - ((hours * rate) * (deductions + tax_rate))
    return weekly_pay

weekly_pay = calcPay(.05, .15)
print("Your weekly pay is: $", weekly_pay)
```

```
def calcPay(hours, rate):                    # Method 5: Data in; No data out; Two global
(avoid)
    global DEDUCTIONS
    global TAX_RATE
    weekly_pay = (hours * rate) - ((hours * rate) * (DEDUCTIONS + TAX_RATE))
    print("Your weekly pay is: $", weekly_pay)

DEDUCTIONS = .05
TAX_RATE = .15
hours = float(input("How many hours did you work this week? "))
rate = float(input("What is your hourly pay rate? "))
calcPay(hours, rate)
```

Passing Lists to Functions (Lists Covered Later)

Example: Calling a function and passing in a mutable object (list) - works like pass by Reference

```
def addChecking(a, b):  
    print ("Your current checking and saving balance are: ", a)  
    a[0] = a[0] + b  
  
bankAccounts = [540.34, 1250.00]  
addChecking(bankAccounts, 100)  
print ("Your new checking and saving balance are: ", bankAccounts)
```

Output:

```
Your current checking and saving balance are: [540.34, 1250.0]  
Your new checking and saving balance are: [640.34, 1250.0]
```

Example: Demonstration of immutable objects and a function

```
num1=2  
num2=3  
  
def changeMe(num1, num2):  
    num1=4  
    num2=5  
    print ("Inside the function: ", num1, num2)  
  
changeMe(num1, num2)  
print ("After the function: ", num1, num2)
```

Output:

```
Inside the function: 4 5  
After the function: 2 3
```

Example: Passing a mutable object into a function

```
num1=2  
num2=3  
nums=[num1, num2]  
print ("Before the function: ", nums)  
  
def changeMe(nums):  
    nums[0] = 4  
    nums[1] = 5  
    print ("Inside the function: ", nums)  
  
changeMe(nums)  
print ("After the function: ", nums)
```

Output:

```
Before the function: [2, 3]  
Inside the function: [4, 5]  
After the function: [4, 5]
```