Topic 1: Pre-defined and User-defined Functions

- **Learning outcomes:**
  - (a) Identify concepts of functions. (1 hour)

Table of Contents (TOC)

Type of function	
• pre-defined functions	• User-defined functions

Definition of pre-defined functions	Definition of user-defined functions
• A function that are readily available for use	• A function that is defined by a programmer

TYPE OF FUNCTIONS

PRE-DEFINED FUNCTIONS

- Definition: A function that are **readily available** for use.
- Example :
  - `input()` – allowing user input
  - `print()` – prints to the standard output device
  - `pow()` – returns the value of `x` to the power of `y`
  - `max()` – returns the largest item in an iterable

USER-DEFINED FUNCTIONS

- Definition: A function that is **defined by a programmer**.
- Created by the user to perform a specific task or to solve problems.
- Example:

```
def my function():
  print("Hello from a function")

my_function()
```

Function Definition

Function Call

Advantages of using functions

- Function makes your code **reusable**, **better organized**, and **more readable**.
- A function is a set of statements so they can be run more than once in a program.

Advantages of using functions: Reusable feature

Without function	User-defined function

a,b = 10,5
average = (a+b)/2
print("Average is: ", average)
 	
c,d =100,5
average = (c+d)/2
print("Average is: ", average)
 	
e,f =100,50
average = (e+f)/2
print("Average is: ", average)
 	
g,h = 9,50
average = (g+h)/2
print("Average is: ", average)
	def Average(a,b):
	    average = (a+b)/2;
	    print("Average is: ", average)
	
	Average(10,5)
	Average(100,5)
	Average(100,50)
	Average(9,50)

Example of pre-defined functions

```
print("Hello, World!")
```
← **print()** function to display String

```
input('Enter your name:')
```
← **input()** function allows user input

```
x = pow(4, 3)
```
← **pow( )** function returns the value of x to the power of y

```
a = (1, 2, 3, 4, 5)
x = sum(a)
```
← **sum()** function returns a value of the sum

Python has a set of pre-defined functions

For more information, visit https://www.w3schools.com/python/python_ref_functions.asp

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.
classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed

Example of user-defined functions

Example 1	Example 2

`def message():`
`  print("Hello")`
 	
`message()`
	`def my_function(fname):`
	`  print(fname + " Refsnes")`
	
	`my_function("Emil")`
	`my_function("Tobias")`
	`my_function("Linus")`

The characteristics of a user-defined functions

- It groups a block of code together so that we can call it by name.
- It enables us to pass values into the the function when we call it.
- It can return a value (even if None).
- When a function is called, it has its own namespace (function space). Variables in the function are local to the function only (and disappear when the function exits).

How user-defined function works

Figure 1 shows

- when you **invoke** a function, Python remembers the place where it happened and *jumps* into the invoked function;
- the body of the function is then **executed**;
- reaching the end of the function forces Python to **return** to the place directly after the point of invocation.

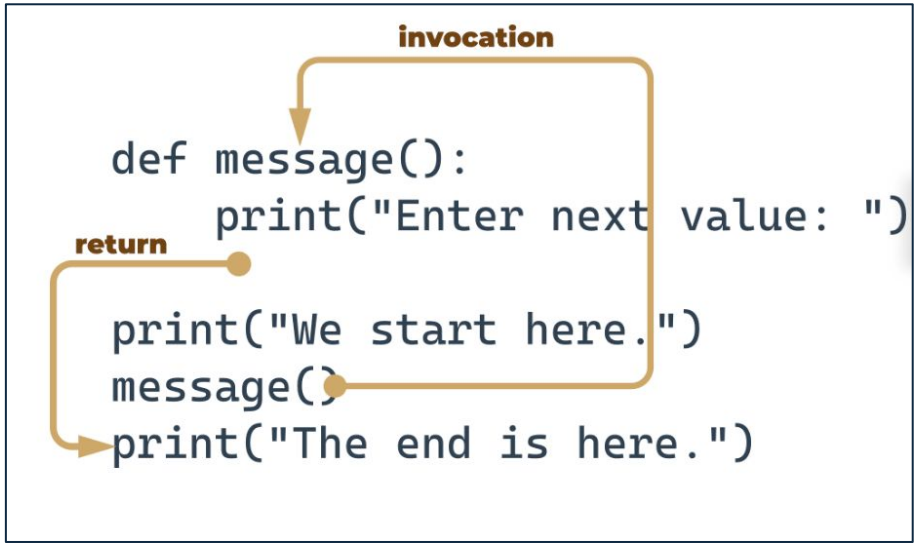


Figure 1

How user-defined function works

- In Python, calling a function before declaring it is not possible.
- Unlike compiled programming languages (such as Java), Python is an interpreted language, it does not allow invoking/ called a function before its definition.
- The order of function definitions matters, and you must define a function before you can call it.
- This is considered as a runtime error

```
def message():
    print("Enter a value: ")

print("We start here.")
message()
print("We end here.")
```

Function Definition

Function Call

You must define a function before you can call it

```
print("We start here.")
message()
print("We end here.")

def message():
    print("Enter a value: ")
```

Function Call

Function Definition

Calling a function before declaring it is not possible

How to create a user-defined function

Step 1 — **Create Function Definition**
- Function Header
- Function Body

Step 2 — **Write Function Call**

```
def message():
    print("Enter a value: ")

print("We start here.")
message()
print("We end here.")
```

Example of a program

Exercise

Question 1: The `input()` function is an example of a:

a) user-defined function

b) pre-defined function

Question 2: The following snippet is an example of a:

```
def message():
    print("Hello")

message()
```

a) user-defined function

b) pre-defined function

Exercise

Question 3: What happens when you try to invoke a function before you define it?

```
hi()

def hi():
    print("hi!")
```

Question 4: The following snippet is an example of a:

```
x = pow(4, 3)
print(x)
```

a) user-defined function

b) pre-defined function

Summary

- Type of functions
- Definition of pre-defined function and user-defined functions
- Advantages of using functions
- Reusable feature
- Example of pre-defined functions
- Example of user-defined functions
- The characteristics of a user-defined functions
- How user-defined function works
- How to create user-defined function
- Exercise

Topic 1: Pre-defined and User-defined Functions

- **Learning outcomes:**

  (c) Identify the components of a function:

  - Function Header
  - Function Body
  - Function Call

(1 hour)

Table of Contents (TOC)

Introduction to Namespace

- Global space refers to the namespace that is outside of the function space.
- In this course we use the term global space and main space interchangeably.
- Moving forward, the main space is the term that we will use.

```
## Function space ##
def hi():
    print("hi!")
```

```
## Main space ##
hi()
```

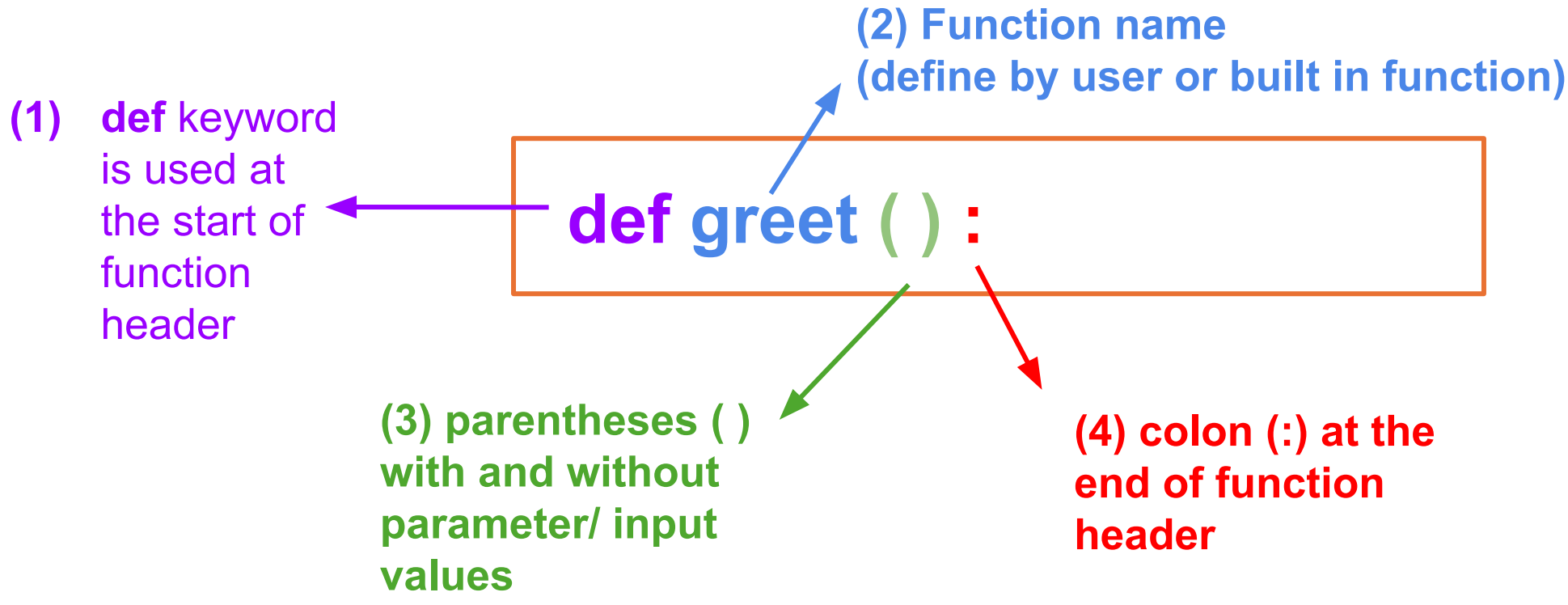
LECTURE NOTES

Three components of a function:

i. Function Header
ii. Function Body
iii. Function Call

i. Function Header

Four Elements in Function Header



Examples of Function Header

Function header without parameter

```
def status():

def message():
```

Function header with parameter

```
def find_total(a, b):

def display_lect(name):

def find_avg(no1, no2, no3):
```

- The function header name is using snake_case naming convention
- Parameter - Input value(s) for the function to process.
- Example - no1, no2 and no3 that will be processed to find the average

Exercise 1

Exercise 1.1

(i) Create a function header named `determine_grade` without parameter.

```
def __________ _
```

(ii) Create a function header named `display_message` without parameter.

```
_______ ________ () :
```

Exercise 1.1

(iii) Create a function header named `calc_total` that accepts two parameters, `a` and `b`.

```
_______ total(___, b):
```

(iv) Create a function header named `calc_sum` that accepts two parameters named `num1` and `num2`.

```
___ _________ (_____, _________):
```

Exercise 1.2

(i) This function named `display_msg()` is to display "Welcome back". Identify the **errors** from this snippet below.

```
## create function ##

def display_msg();
   print("Welcome back")

## main ##
msg()
```

Exercise 1.2

(i) This function named `display_msg()` is to display "Welcome back". Identify the **errors** from this snippet below.

```
  File "/tmp/ipython-input-2761624640.py",
    def display_msg();
                     ^
SyntaxError: expected ':'
```

```
NameError                                 Traceback
/tmp/ipython-input-1476903710.py in <cell line: 0>()
      5
      6 ## main ##
----> 7 msg()
      8
      9

NameError: name 'msg' is not defined
```

Exercise 1.2

(ii) This function named `display_msg()` is to display "Welcome back". Identify the error from this snippet below.

```
## create function ##

def display_msg():
print("Welcome back")

## main ##
display_msg()
```

Exercise 1.2

(iii) This function named `display_msg()` is to display "Welcome back". Identify the error from this snippet below.

```
## create function ##

def display_msg:
   print("Welcome back")

## main ##
display_msg()
```

Exercise 1.2

(iv) This function named `display_msg()` is to display "Welcome back". Identify the error from this snippet below.

```
## create function ##

defdisplay_msg():
   print("Welcome back")

## main ##
display_msg()
```

ii. Function Body

- The body of function consists of **one or more valid statements**.(Eg: to calculate max, to print value)

- Each statement must be indented with the **same indentation** to form a block.

ii) Function Body (without or with return values)

a) Function body without return value
(no **return** keyword)

a) Function body with return value
(must have **return** keyword)

Example of Function Body

(a) Function body without return value (no return keyword)

Function body

```
def avg():
    total = 100 + 20 + 30
    average = total / 3
    print (average)
```

```
def calcAvg(num1,num2,num3):
    total = num1 + num2 + num3
    avg = total/3
    print(total)
```

Function body

(b) Function body with return value (must have return keyword)

Function body with return keyword

```
def calcAvg(num1,num2,num3):
    total = num1 + num2 + num3
    avg = total/3
    return avg
```

Function body with return keyword

```
def greet(message):
    return message
```

iii) A. Function Call

- Function without function call is meaningless.
- Therefore, we need to create function call.
- To call a function, use the **function name** followed by parenthesis(), with or without arguments.

iii) A. Function Call - continued

- Once the function is invoked (called) in main program space, the parameters inside the parentheses () is called **arguments**.
- The number of arguments **must match with** parameters of the function.
- Arguments must have values

```
## create function to add two numbers ##
def add_numbers(a, b):
    result = a + b
    return result

## main ##
sum_result = add_numbers(5, 3)
print("The sum is:", sum_result)
```

Parameters

Function call

Arguments

Example Function Call Without Arguments

function space

```
def greet():
    print('Hello World!')
```

⟹ function header

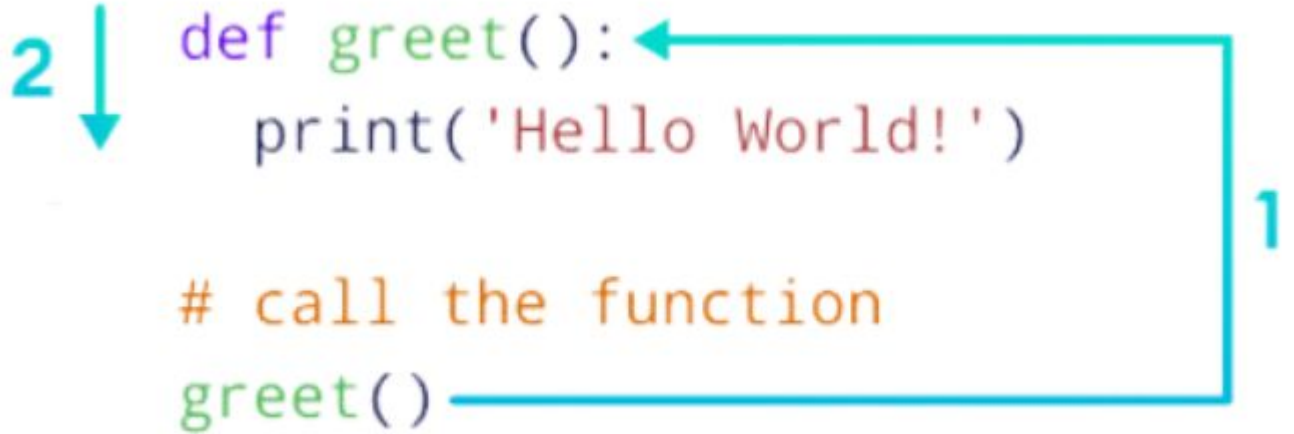
⟹ function body

main program space

```
# call the function
greet()
```

⟹ **function call (name must be the same as function name)**

iii) B. How the program execute when the function is called



1. When the function `greet()` is called from the main space, the program's control transfers to the function definition in function space.
2. All the code inside the function is executed.

iii) C. Function Call without argument(s)

1. Function call without argument(s)
    - 1.1 Without return value
    - 1.2 With return value

Return value is the **result** that a function sends back to the main space after it finishes running.

1. Function Call without argument(s)

1.1 Without return value

```
## create function ##
def greet():
 print ('Hi there')

## main ##
greet()
```

function call

Output:

```
Hi there
```

1. Function Call without argument(s)

1.2 With return value

```
## create function ##
def get_greeting():
    return "Hello, welcome back!"

## main ##
print(get_greeting())
```

*Direct print from the returned value

```
## create function ##
def get_greeting():
    return "Hello, welcome back!"

## main ##
message = get_greeting()
print(message)
```

*Store returned value in a variable to be used later

Output:

```
Hello, welcome back!
```

iii) D. Function Call with argument(s)

1. Function call with argument(s)
    1.1 Without return value
    1.2 With return value

iii) D. Function Call with argument(s)

2.1 Without return value

```
## create funtion to print a grade ##
def print_grade(grade):
    print("The grade is:", grade)

## main ##
print_grade("A")
```

function call with argument

Output:

```
The grade is: A
```

2.2 With return value

```
## create function to add two numbers ##
def add_numbers(a, b):
    result = a + b
    return result

## main ##
sum_result = add_numbers(5, 3)
print("The sum is:", sum_result)
```

function call with arguments

Output:

```
The sum is: 8
```

Parameter vs Argument

- Parameter: in function header (function space)
- Argument: in function call (main program space)

Recap

LECTURE NOTES


```
## create function calcArea ##
def calcArea(length, width):
    area_rectangle = length * width
    return area_rectangle

## main ##
print(calcArea(2,3))
print(calcArea(4,20))
```

parameters (variable) — length, width

arguments (values) — 2, 3 / 4, 20

Output:

```
6
80
```

LECTURE NOTES

```
## create function calcArea ##
def calcArea(length, width):
    area_rectangle = length * width
    return area_rectangle


## main ##
print(calcArea(2,3))
print(calcArea(4,20))
```

parameters used in function header

arguments used in function call

LECTURE NOTES

```
## create function calcArea ##
def calcArea(length, width):
    area_rectangle = length * width
    return area_rectangle

## main ##
areaRect = calcArea(2,3)
print (areaRect)
areaRect = calcArea(4,20)
print (areaRect)
```

we could assign function call to a variable

Output:

```
6
80
```

Exercise 1

Write a function to calculate area of a rectangle (with and without return value)

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))
area = length * width
print("The area of the rectangle is:", area)
```

Exercise 2

Exercise 2.1

(i) Create a function header named `bmi()` that accepts two parameters named `height` and `weight`.

(ii) Call the previous function by using the values 1.53 and 55 for height and weight respectively. (Assume there is a return values)

Exercise 2.2

(i) Create a function header named `sum()` that accepts three integers value named `num1`, `num2` and `num3`.

(ii) Call the previous function by using the values 10, 20 and 30 for num1, num2, and num3 respectively. (Assume there is a return values)

(iii) Identify the arguments and parameters from (i) and (ii)

Summary

- Introduction to Namespace
- Three components of a function
  i. Function Header
  ii. Function Body
  iii. Function Call
    A. Example Function Call
    B. How the program execute when the function is called
    C. Function Call without argument(s)
    D. Function Call with argument(s)
- Parameter vs argument

Topic 1: Pre-defined and User-defined Functions

- **Learning outcomes:**

(f) Identify how to call functions and write simple user-defined functions to perform calculation.

(1 hour)

Table of Contents (TOC)

a. Function without parameter without return value - Calling function once

function name →
function body →

```
### Function Definition ###

def acknowledge():
    print("Hi, Welcome!")


### Main Program ###
acknowledge()
```

calling function →

```
Hi, Welcome!
```

Output

Function Definition and Main Program must be separated because they exist in different spaces. We can create many functions but only has one main program.

a. Function without parameter without return value - Calling function three time

Function can be invoked/ called multiple times →

```
### Function Definition ###
def acknowledge():
  print("Hi, Welcome!")

### Main Program ###
acknowledge()
acknowledge()
acknowledge()
```

```
Hi, Welcome!
Hi, Welcome!
Hi, Welcome!
```

Output

a. Function without parameter without return value - Calling function location

```
### Main Program ###
acknowledge()

### Function Definition ###
def acknowledge():
  print("Hi, Welcome!")
```

```
Traceback (most recent call last):
  File "./prog.py", line 2, in <module>
NameError: name 'acknowledge' is not defined
```

Output

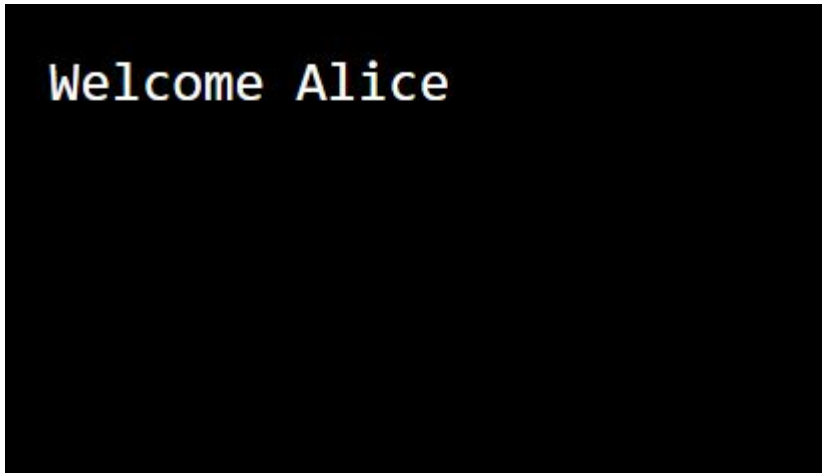
Error message?

Because Python executes code from top to bottom, Python won't know what you're referring to, *acknowledge()*, and will raise an error. Functions should be defined before they are invoked/ called.

b. Function without parameter with return value

```
### Function Definition ###
def acknowledge () :
  name = "Alice"
  return name

### Main Program ###
print ("Welcome", acknowledge())
```



Output

c. Function with parameter without return value
- Using static value as argument

```
### Function Definition ###
def acknowledge(name):
    print("Hi, "+name +", Welcome!")

### Main Program ###
acknowledge("Ahmad")
acknowledge("Mutu")
acknowledge("Chong")
```

```
Hi, Ahmad, Welcome!
Hi, Mutu, Welcome!
Hi, Chong, Welcome!
```

Output

The values of "Ahmad", "Mutu", and "Chong" were passed on to function acknowledge() one at a time and received via variable ***name*** (as a parameter).

c. Function with parameter without return value
- Using variable as argument

```
### Function Definition ###
def acknowledge(name):
    print("Hi, "+name +", Welcome!")

### Main Program ###
student = "Ahmad"
acknowledge(student)
student = "Mutu"
acknowledge(student)
student = "Chong"
acknowledge(student)
```

```
Hi, Ahmad, Welcome!
Hi, Mutu, Welcome!
Hi, Chong, Welcome!
```

Output

The variables in the main program and the function do not have to be the same.

d. Function with parameter with return value
- Example 1 of calling function

```
### Function Definition ###
def sum(x, y):
    return x + y;

### Main Program ###
result = sum(5,9)
print(result)
```

Calling function →

```
14
```

Output

Calling function is sum(5,9), then the return value is saved/stored in a variable “result” in main program space

d. Function with parameter with return value
- Example 2 of calling function

```
### Function Definition ###
def sum(x, y):
    return x + y;

### Main Program ###
result = sum(5,9)
print(result)
print(sum(7,8))
```

2 different ways of calling functions with return values

```
14
15
```

Output

2. Construct simple functions

General steps

1. Create the Header
2. Complete the Header with or without a parameter
3. Create the function body
4. Calling the function in main program

2. a. Functions with Parameter(s) and without Return Value

Create a function to calculate average of three numbers and print the average.

Steps to solve the problem

1. **Create the Header** for function name using meaningful identifier;
   def calc_average( ) :

2. **Complete the Header with the required parameter(s)**.
   Identify whether the function header needs to have a parameter or not. For this function three numbers should be passed into the function to calculate the average.
   def calc_average(num1, num2, num3) :

*variable num1, num2, num3 can be any meaningful identifier

Steps to solve the problem

3. **Create the function body** : to calculate the average of the three numbers and print the average (without return statement).

```
### Function Definition ###
def calc_average(num1, num2, num3):
    average = (num1 + num2 + num3) / 3
    print(average)
```

Explanation of the function body

(the body statements depend on the problems given)

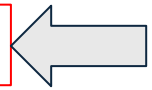
- We start by calculating the sum of the three numbers (num1, num2, and num3) and divide by 3 and store the average result in the variable calc_average.
- We **must use the variables (or input) that have been declared in the parameters**. The input **originates from the main program.**
- Finally, we print out the value of average stored in calc_average

Steps to solve the problem

4. Calling the function in main program

```
### Function Definition ###
def calc_average(num1, num2, num3):
    average = (num1 + num2 + num3) / 3
    print(average)

### Main Program ###
calc_average(3, 4, 5)
```



Calling the function

2. b. Functions with Parameter(s) and Return Value

Create a function to calculate area of a rectangle and return the result to main program

Steps to solve the problem

1. **Create the Header** for function name using meaningful identifier;

```
def area_rect(  ) :
```

2. **Complete the Header with parameter(s)**.
   Identify whether the function header needs to have a parameter or not. For this function variable length and width should be passed into the function to calculate the area.

```
def area_rect(length, width) :
```

Steps to solve the problem

3. **Create the function body** : to calculate area of rectangle and return the result to main program (with return statement).

```
### Function Definition ###
def area_rect(length, width):
    area = length * width
    return area
```

return statement

Steps to solve the problem

4. Calling the function in main program

```
### Function Definition ###
def area_rect(length, width):
    area = length * width
    return area

### Main Program ###
result = area_rect(5, 6)
print(result)
```

Calling the function

Activity 1

Create a function to receive two integers and returns the larger integer value.

Ask students to follow these 4 steps to solve the problem (4Cs)
1. Create the Header
2. Complete the Header with or without a parameter.
3. Create the function body
4. Calling the function in main program

Steps to solve the problem

1. Create the header

```
def max_of_two( ):
```

2. Complete the Header with or without a parameter

```
def max_of_two(a, b):          #a and b is variables for the two integers
```

3. Create the function body

```
if a > b:
        return a
else:
        return b
```

4. Calling the function in main program

result = max_of_two(5, 10)

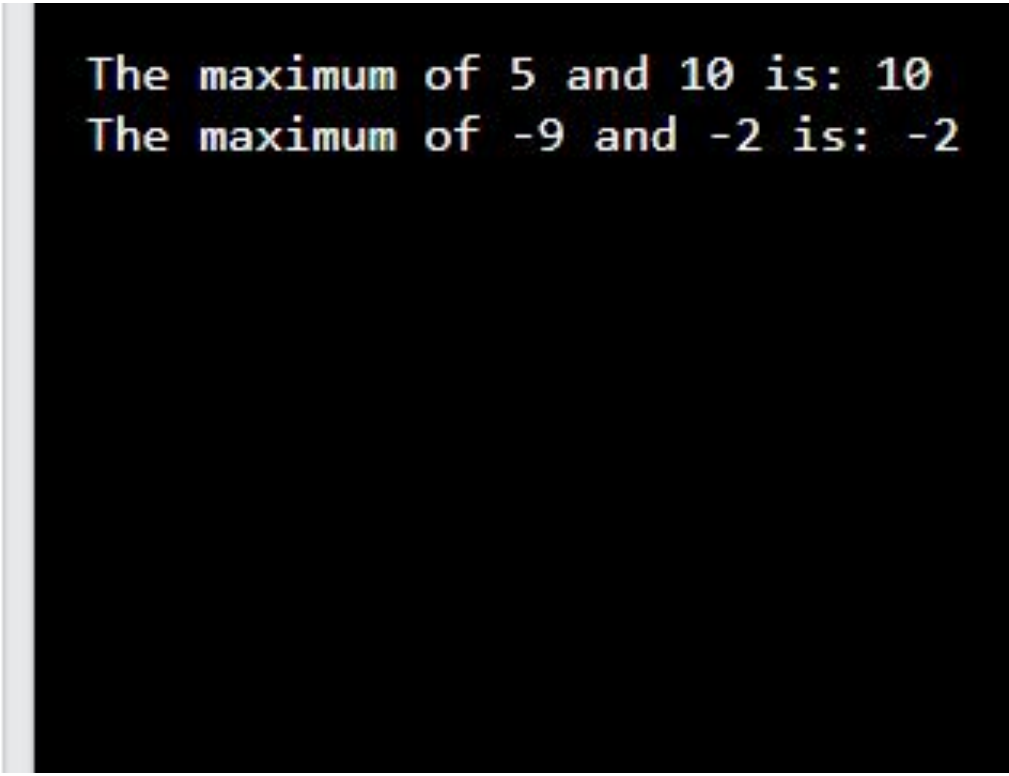
```
### Function Definition ###
def max_of_two(a, b):
    if a > b:
        return a
    else:
        return b

### Main Program ###
result = max_of_two(5, 10)
```

Sample complete answer (with return statement using Selection Control Structure)

```
### Function Definition ###
def max_of_two(a, b):
    if a > b:
            return a
    else:
            return b

### Main Program ###
result = max_of_two(5, 10)
print("The maximum of 5 and 10 is:", result)
print("The maximum of -9 and -2 is:",
max_of_two(-9,-2))
```



Output

Activity 2 (without return)

Create a function to determine whether a number given by user is odd or even.

Ask students to follow these 4 steps to solve the problem (4Cs)
1. Create the Header
2. Complete the Header with or without a parameter.
3. Create the function body
4. Calling the function in main program

Steps to solve the problem

1. Create the Header

```
def odd_or_even( ):
```

2. Complete the Header with or without a parameter.

```
def odd_or_even(num ):          #num is variables for the number given by
                                #the user
```

3. Create the function body

```
if num % 2 == 0:
    print("The number is even.")
else:
    print("The number is odd.")
```

4. Calling the function in main program

```
number = int(input("Enter any number: "))
odd_or_even(number)
```

Sample Answer

```
1  ## Function Definition ##
2  def odd_or_even(num):
3      if num % 2 == 0:
4          print("The number is even.")
5      else:
6          print("The number is odd.")
7
8  ## Main Program ##
9  number = int(input("Enter any number: "))
10 odd_or_even(number)
```

Summary

1. How to call functions
    a. Functions without Parameter without Return Value
    b. Functions without Parameter with Return Value
    c. Functions with Parameter without Return Value
    d. Functions with Parameter with Return Value
2. Construct simple functions
    a. Functions with Parameter without Return Value
    b. Functions with Parameter with Return Value