Instructor Notes:

Instructor Notes:

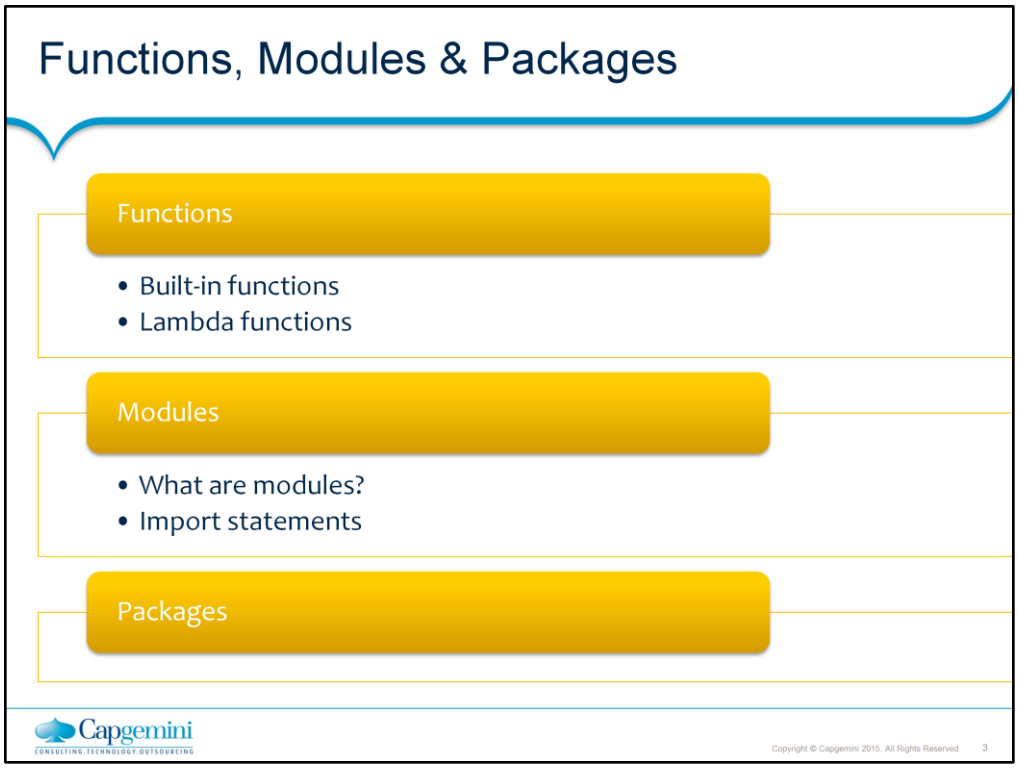
Lesson Objectives

- After completing this lesson, you will learn about:
  - Functions
  - Modules
  - Packages

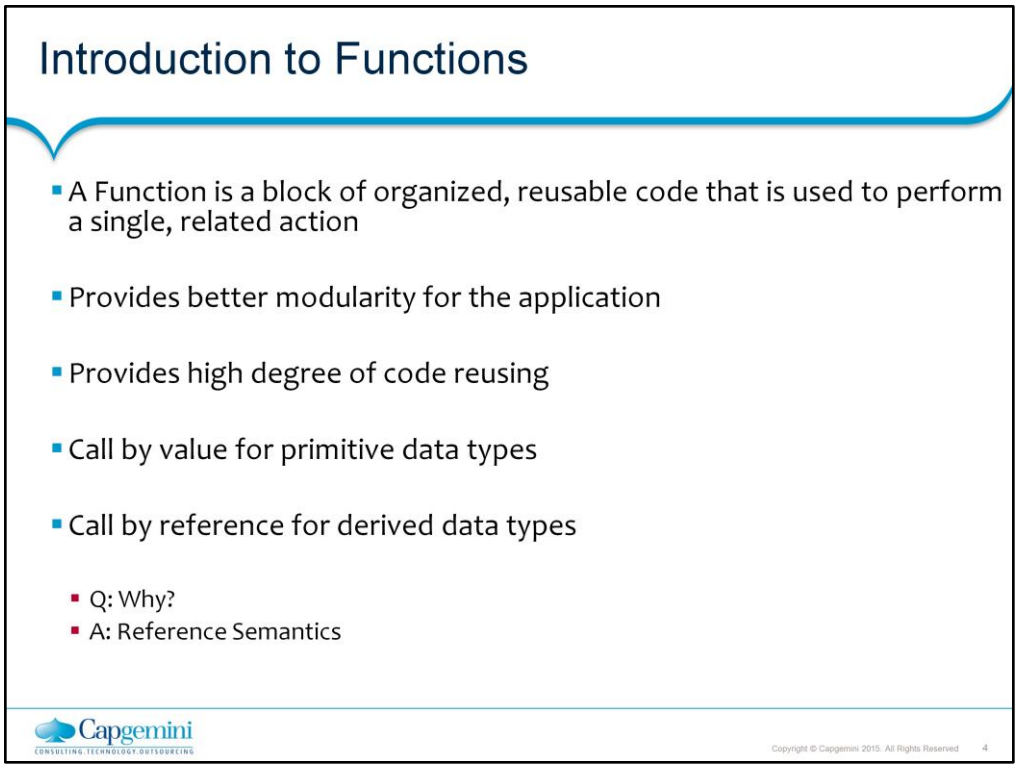
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Instructor Notes:



Instructor Notes:



- A function is one that returns a value. The opposite of a fruitful function is void function — one that is not executed for its resulting value, but is executed because it does something useful. (Languages like Java, C#, C and C++ use the term “void function”, other languages like Pascal call it a procedure.) Even though void functions are not executed for their resulting value, Python always wants to return something. So if the programmer doesn’t arrange to return a value, Python will automatically return the value None.

Instructor Notes:

Functions..contd

- Function blocks begins with **def** keyword, followed by the name of the function and parentheses ()
- Input parameter is placed within these parentheses and parameter can be defined inside these parentheses
- The first statement of a function can be an optional statement - the documentation string of the function or docstring
- Code block within every function starts with a colon (:) and is indented
- The statement return [expression] exits a function, optionally passing back an expression to the caller
- A return statement with no arguments is the same as return None

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Instructor Notes:

Functions..contd

- The syntax for function definition is

```
def NAME( PARAMETERS ):
    """Docstring"""
    STATEMENTS
  [return]
```

- A function must be defined before its first use
- The return statement is used to return a value from function.
- If a function does not have return statement , it is considered as a Procedure
- If a function has to return multiple values , tuples are preferred
- Sample function call

```
name = my_func(arg1, arg2, arg='Default')
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Instructor Notes:

Functions: Parameter passing

`def hello(greeting='Hello', name='world'):`
`  print ('%s, %s!' % (greeting, name))`
 	
`hello('Greetings')`
	Adding default values to parameters

`def hello_1(greeting, name):`
`  print ('%s, %s!' % (greeting, name))`
`# The order here doesn't matter at all: hello_1(name='world', greeting='Hello')`
	Using named parameters. In this case the order of the arguments does not matter.

`def print_params(*params):`
`  print (params)`
 	
`print_params('Testing')`
`print_params(1, 2, 3)`
	The variable length function parameters allow us to create a function which can accept any number of parameters.

`def print_params_3(**params):`
`  print (params)`
 	
`print_params_3(x=1, y=2, z=3)`
	Variable named parameters

`def print_params_4(x, y, z=3, *pospar, **keypar):`
`  print (x, y, z)`
`  print (pospar)`
`  print (keypar)`
 	
`print_params_4(1, 2, 3, 5, 6, 7, foo=1, bar=2)`
`print_params_4(1, 2)`
	A combination of all of above cases

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Instructor Notes:

Built-in functions

abs()	divmod()	***input()***	open()	staticmethod()
all()	***enumerate()***	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	***filter()***	len()	***range()***	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	***reduce()***	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	***map()***	repr()	xrange()
cmp()	globals()	max()	reversed()	***zip()***
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Instructor Notes:

Lambda functions

- Unnamed functions
- Mechanism to handle function objects
- To write inline simple functions
- Generally used along with maps, filters on lists, sets etc.
- Not as powerful as in C++11, Haskell etc. e.g. no looping etc.
- Example: lambda x,y : x+y to add two values

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Instructor Notes:

Modules

- A module is a file containing Python definitions and statements intended for use in other Python programs.
- It is just like any other python program file with extension .py
- Use the “import <module>” statement to make the definitions in <module> available for use in current program.
- A new file appears in this case \path\<module>.pyc. The file with the .pyc extension is a compiled Python file for fast loading.
- Python will look for modules in its system path. So either put the modules in the right place or tell python where to look!

```
import sys
sys.path.append('c:/python')
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Instructor Notes:

Modules

- Three import statement variants

import math x = math.sqrt(10) import math as m print m.pi	Here just the single identifier math is added to the current namespace. If you want to access one of the functions in the module, you need to use the dot notation to get to it.
from math import cos, sin, sqrt x = sqrt(10)	The names are added directly to the current namespace, and can be used without qualification.
from math import * x = sqrt(10)	This will import all the identifiers from module into the current namespace, and can be used without qualification.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Instructor Notes:

Packages

- Packages are used to organize modules. While a module is stored in a file with the file name extension .py, a package is a directory.
- To make Python treat it as a package, the folder must contain a file (module) named __init__.py

File/Directory	Description
~/python/	Directory in PYTHONPATH
~/python/drawing/	Package directory (drawing package)
~/python/drawing/__init__.py	Package code ("drawing module")
~/python/drawing/colors.py	colors module
~/python/drawing/shapes.py	shapes module
~/python/drawing/gradient.py	gradient module
~/python/drawing/text.py	text module
~/python/drawing/image.py	image module

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Instructor Notes: