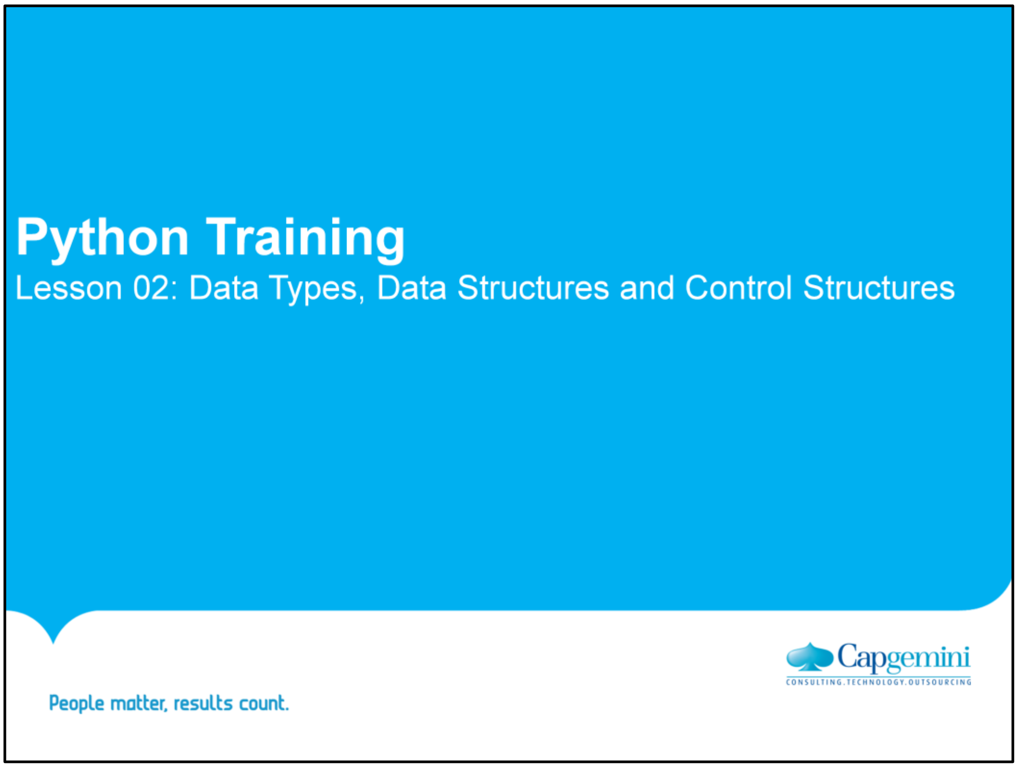
Instructor Notes:

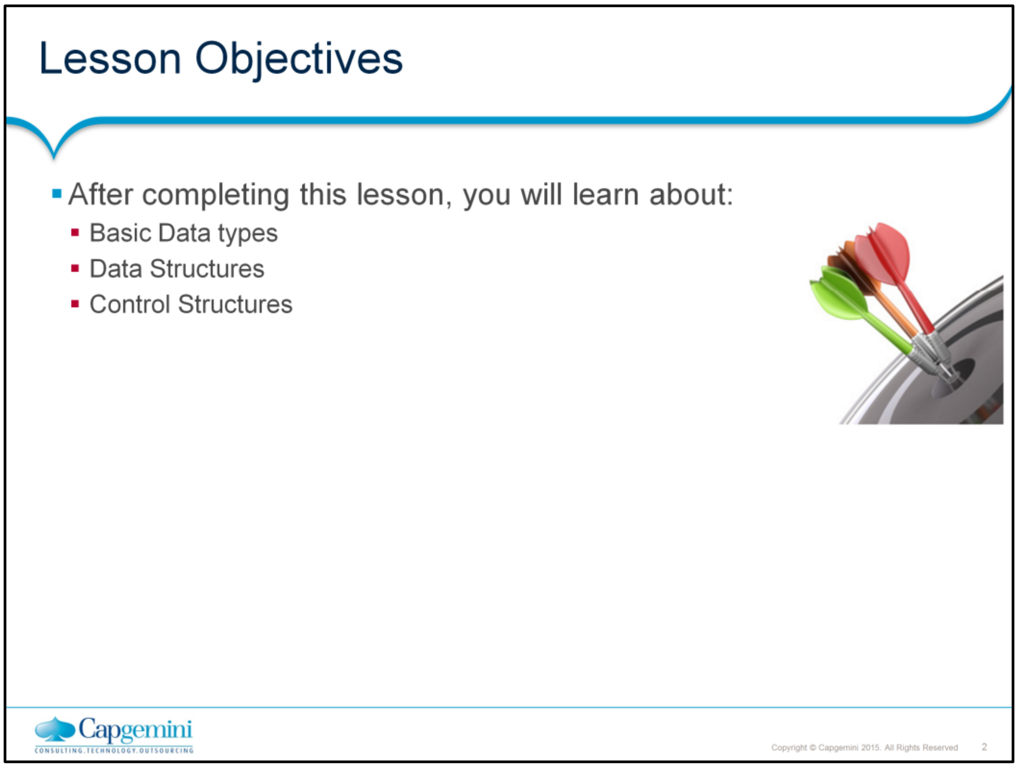


Python Training

Lesson 02: Data Types, Data Structures and Control Structures

People matter, results count.

Instructor Notes:

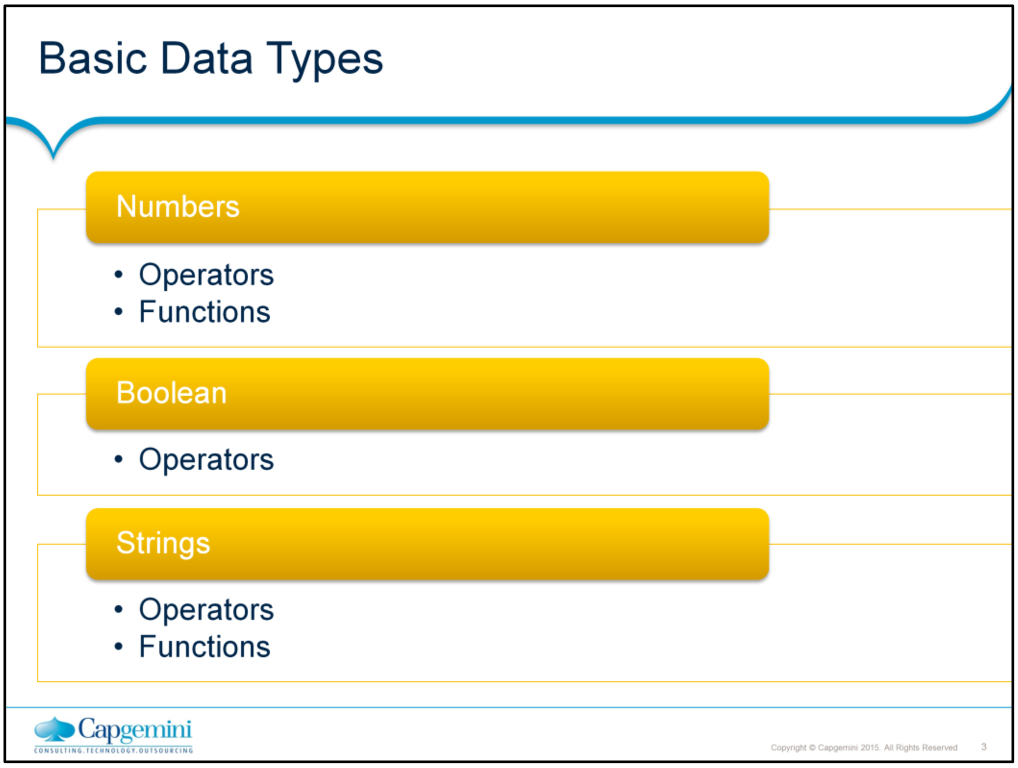


Lesson Objectives

- After completing this lesson, you will learn about:
  - Basic Data types
  - Data Structures
  - Control Structures

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Instructor Notes:



Basic Data Types

Numbers

- Operators
- Functions

Boolean

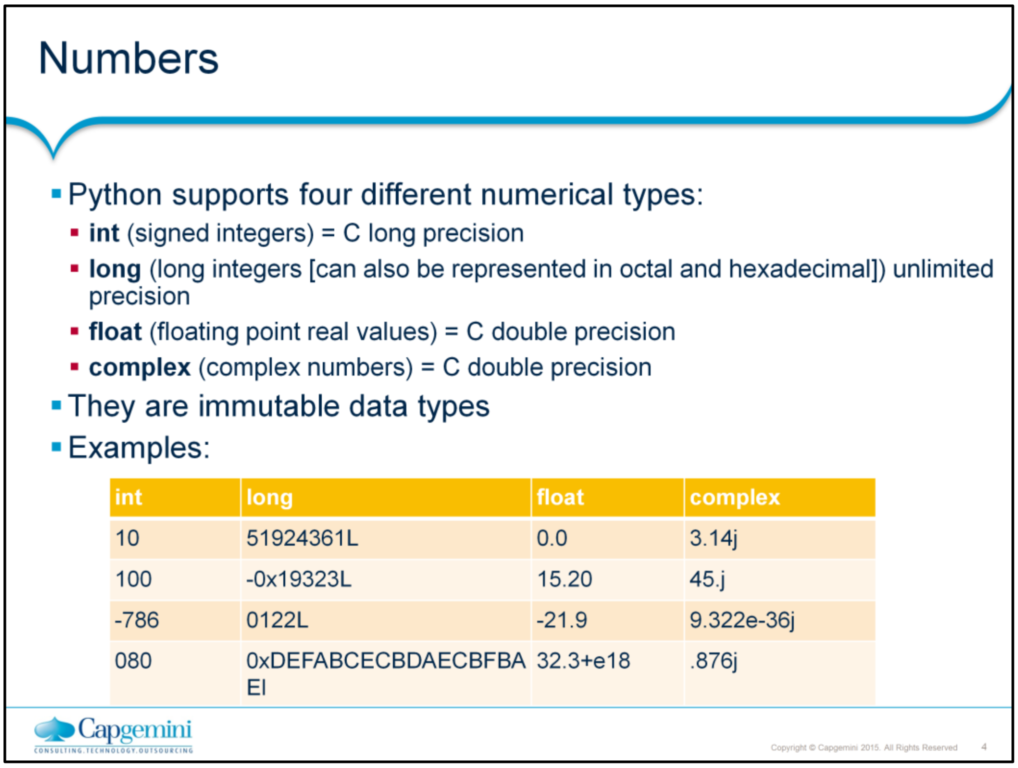
- Operators

Strings

- Operators
- Functions

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Instructor Notes:



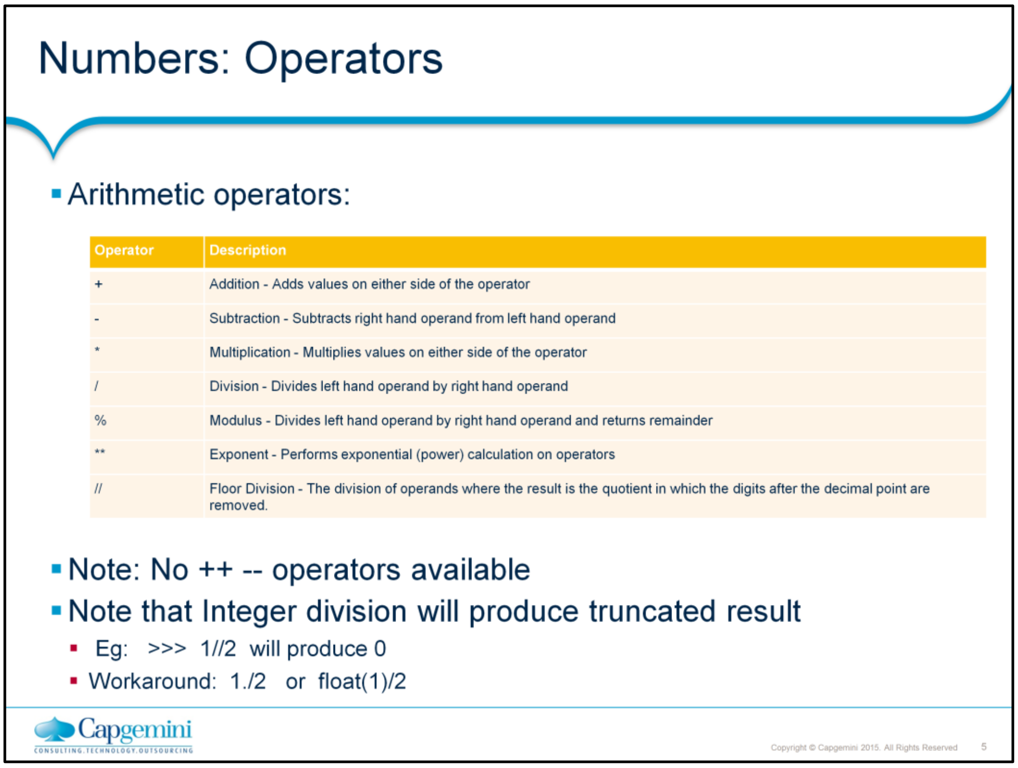
Numbers

- Python supports four different numerical types:
  - **int** (signed integers) = C long precision
  - **long** (long integers [can also be represented in octal and hexadecimal]) unlimited precision
  - **float** (floating point real values) = C double precision
  - **complex** (complex numbers) = C double precision
- They are immutable data types
- Examples:

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEI	32.3+e18	.876j

Immutable means that changing the value of a number data type results in a newly allocated object

Instructor Notes:



Numbers: Operators

- Arithmetic operators:

Operator	Description
+	Addition - Adds values on either side of the operator
-	Subtraction - Subtracts right hand operand from left hand operand
*	Multiplication - Multiplies values on either side of the operator
/	Division - Divides left hand operand by right hand operand
%	Modulus - Divides left hand operand by right hand operand and returns remainder
**	Exponent - Performs exponential (power) calculation on operators
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.

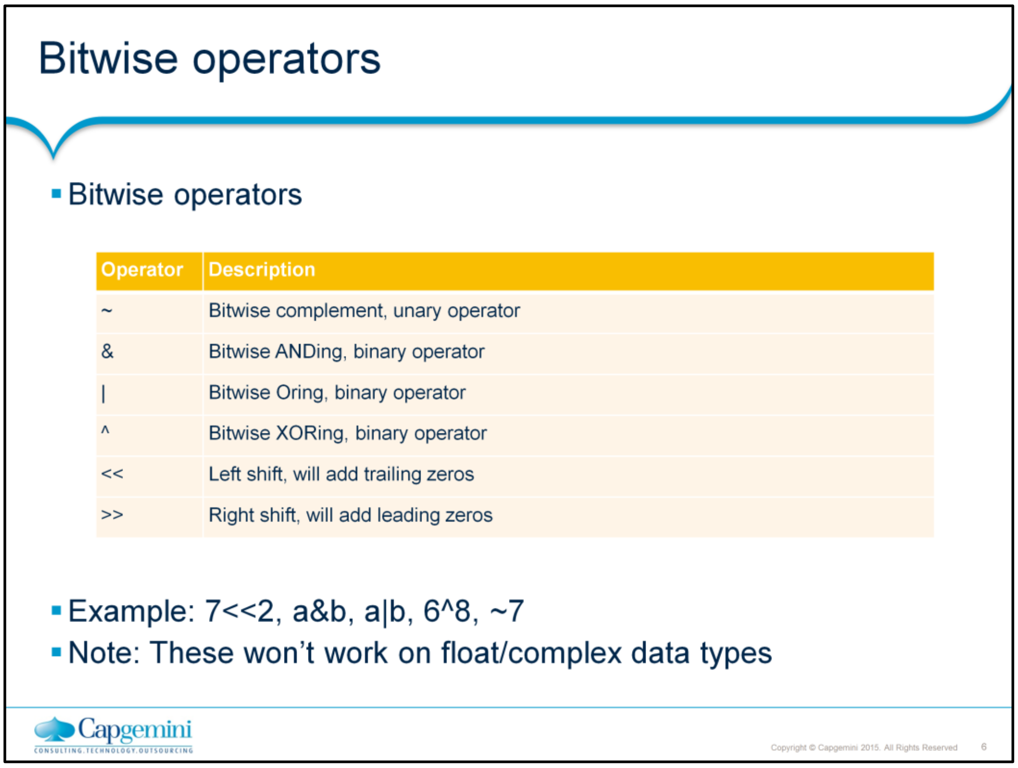
- Note: No ++ -- operators available
- Note that Integer division will produce truncated result
  - Eg: >>> 1//2 will produce 0
  - Workaround: 1./2 or float(1)/2

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

5

Immutable means that changing the value of a number data type results in a newly allocated object

Instructor Notes:



Bitwise operators

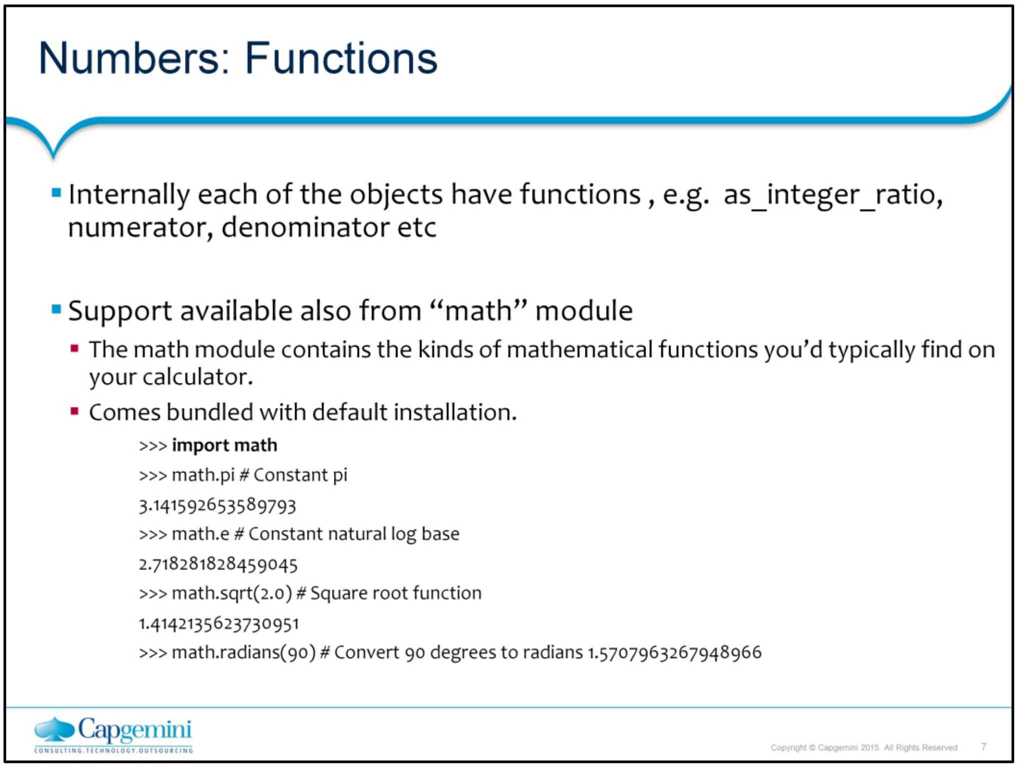
- Bitwise operators

Operator	Description
~	Bitwise complement, unary operator
&	Bitwise ANDing, binary operator
|	Bitwise Oring, binary operator
^	Bitwise XORing, binary operator
<<	Left shift, will add trailing zeros
>>	Right shift, will add leading zeros

- Example: 7<<2, a&b, a|b, 6^8, ~7
- Note: These won't work on float/complex data types

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Instructor Notes:

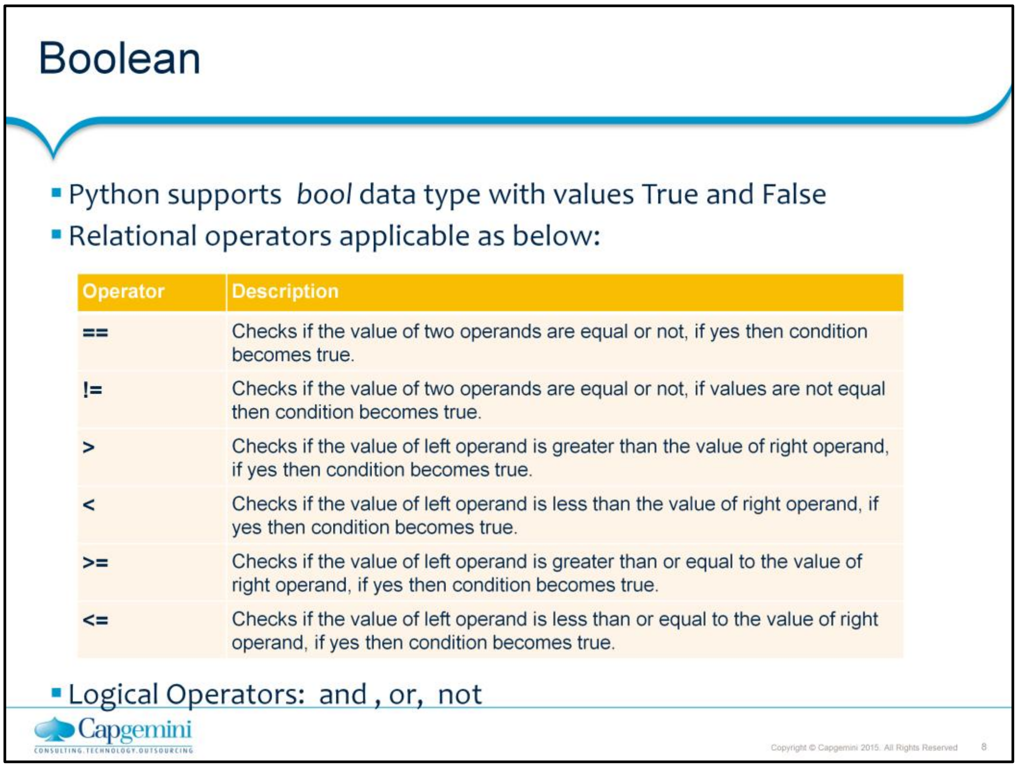


Numbers: Functions

- Internally each of the objects have functions , e.g. as_integer_ratio, numerator, denominator etc
- Support available also from "math" module
  - The math module contains the kinds of mathematical functions you'd typically find on your calculator.
  - Comes bundled with default installation.

```
>>> import math
>>> math.pi # Constant pi
3.141592653589793
>>> math.e # Constant natural log base
2.718281828459045
>>> math.sqrt(2.0) # Square root function
1.4142135623730951
>>> math.radians(90) # Convert 90 degrees to radians 1.5707963267948966
```

Instructor Notes:



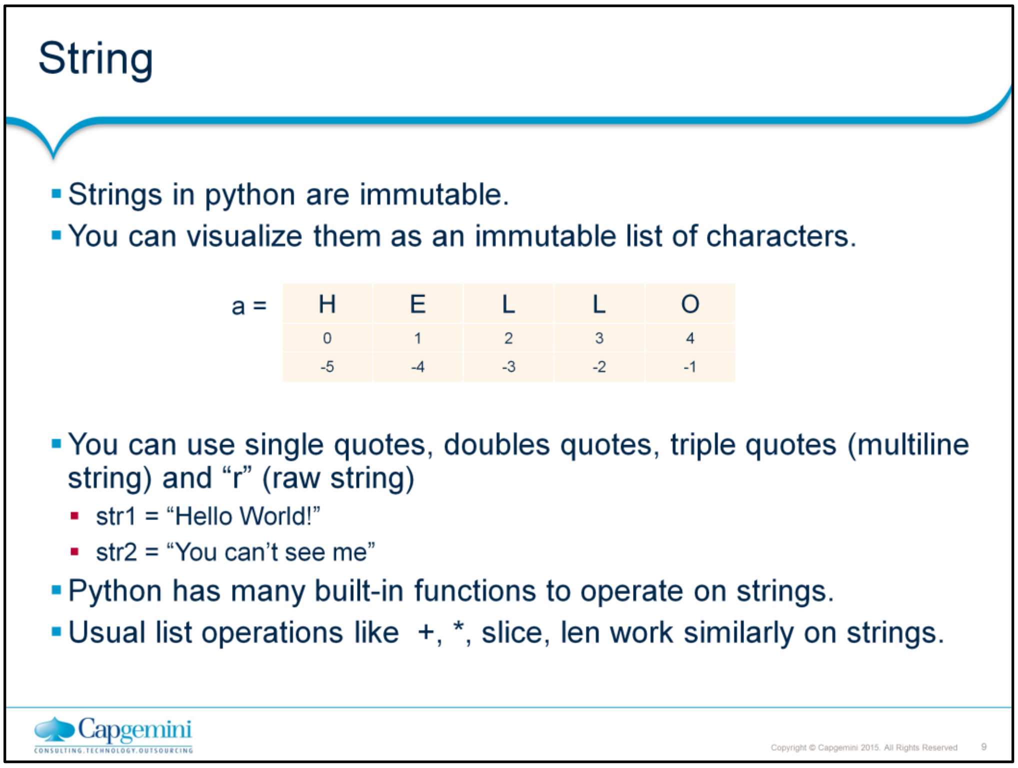
Boolean

- Python supports *bool* data type with values True and False
- Relational operators applicable as below:

Operator	Description
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

- Logical Operators: and , or, not

Instructor Notes:



String

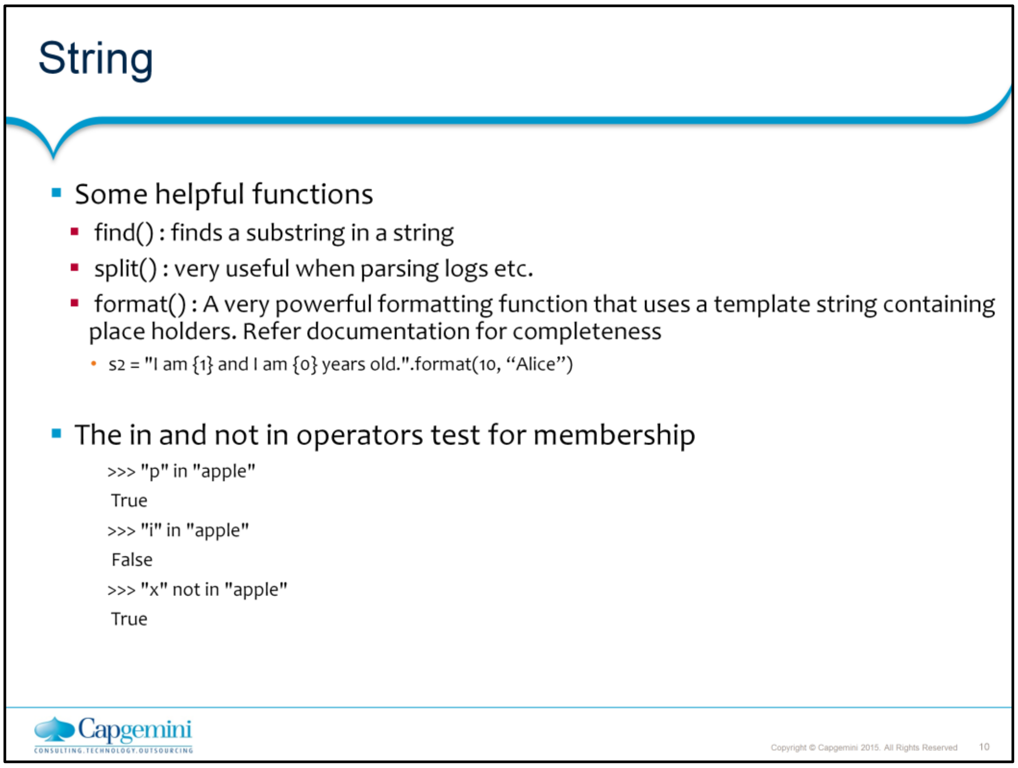
- Strings in python are immutable.
- You can visualize them as an immutable list of characters.

a =	H	E	L	L	O
	0	1	2	3	4
	-5	-4	-3	-2	-1

- You can use single quotes, doubles quotes, triple quotes (multiline string) and “r” (raw string)
  - str1 = “Hello World!”
  - str2 = “You can’t see me”
- Python has many built-in functions to operate on strings.
- Usual list operations like +, *, slice, len work similarly on strings.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Instructor Notes:



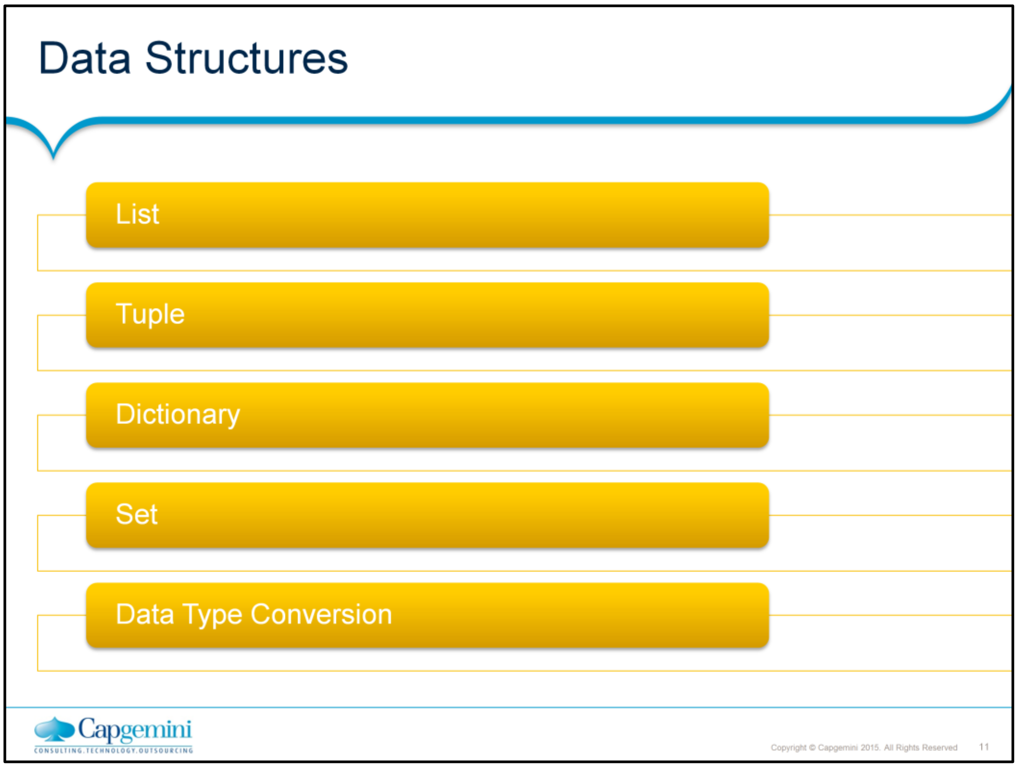
String

- Some helpful functions
  - find() : finds a substring in a string
  - split() : very useful when parsing logs etc.
  - format() : A very powerful formatting function that uses a template string containing place holders. Refer documentation for completeness
    - s2 = "I am {1} and I am {0} years old.".format(10, “Alice”)
- The in and not in operators test for membership

```
>>> "p" in "apple"
True
>>> "i" in "apple"
False
>>> "x" not in "apple"
True
```

10

Instructor Notes:

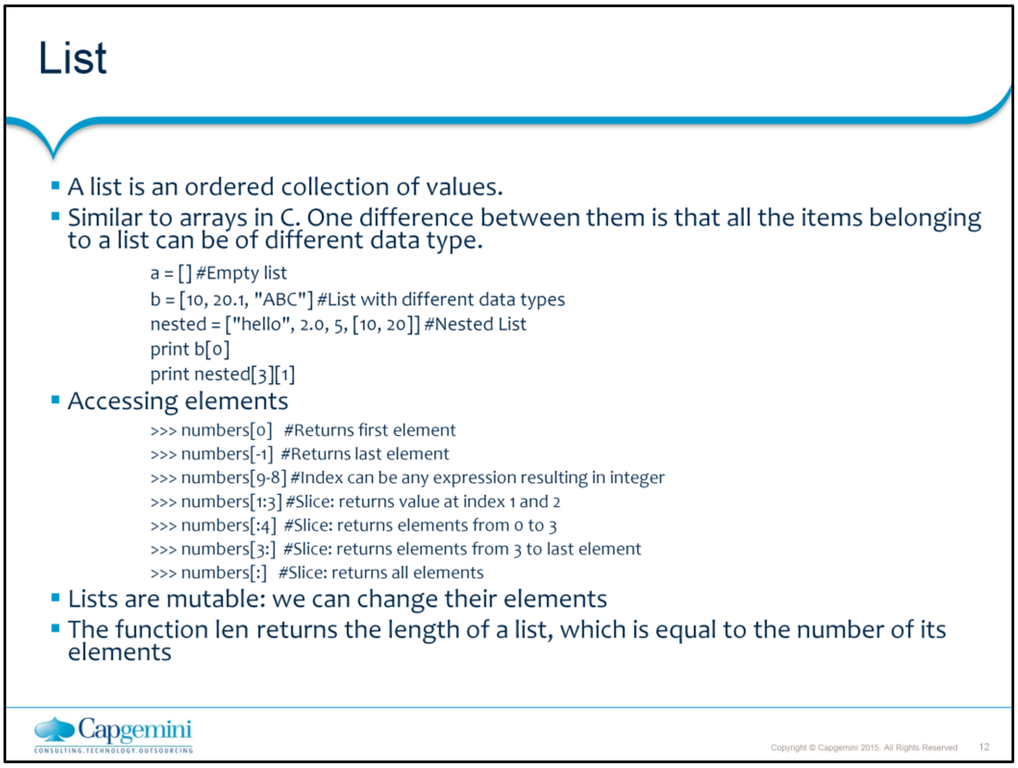


Data Structures

- List
- Tuple
- Dictionary
- Set
- Data Type Conversion

11

Instructor Notes:



List

- A list is an ordered collection of values.
- Similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

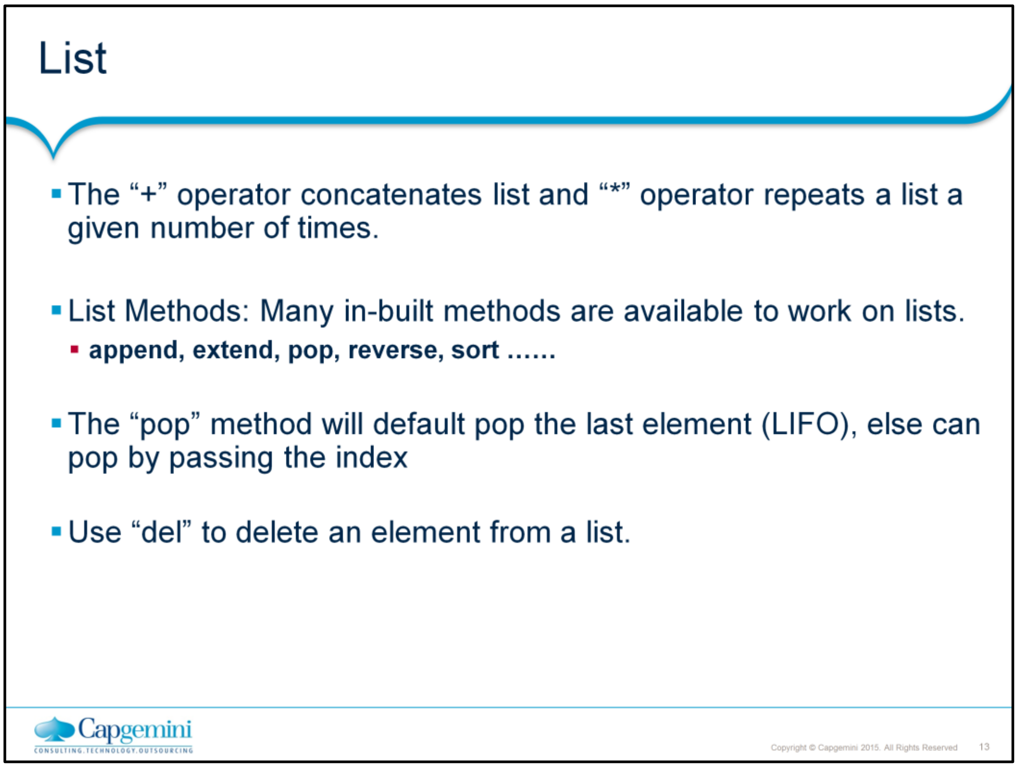
```
a = [] #Empty list
b = [10, 20.1, "ABC"] #List with different data types
nested = ["hello", 2.0, 5, [10, 20]] #Nested List
print b[0]
print nested[3][1]
```

- Accessing elements

```
>>> numbers[0]   #Returns first element
>>> numbers[-1]  #Returns last element
>>> numbers[9-8] #Index can be any expression resulting in integer
>>> numbers[1:3] #Slice: returns value at index 1 and 2
>>> numbers[:4]  #Slice: returns elements from 0 to 3
>>> numbers[3:]  #Slice: returns elements from 3 to last element
>>> numbers[:]   #Slice: returns all elements
```

- Lists are mutable: we can change their elements
- The function len returns the length of a list, which is equal to the number of its elements

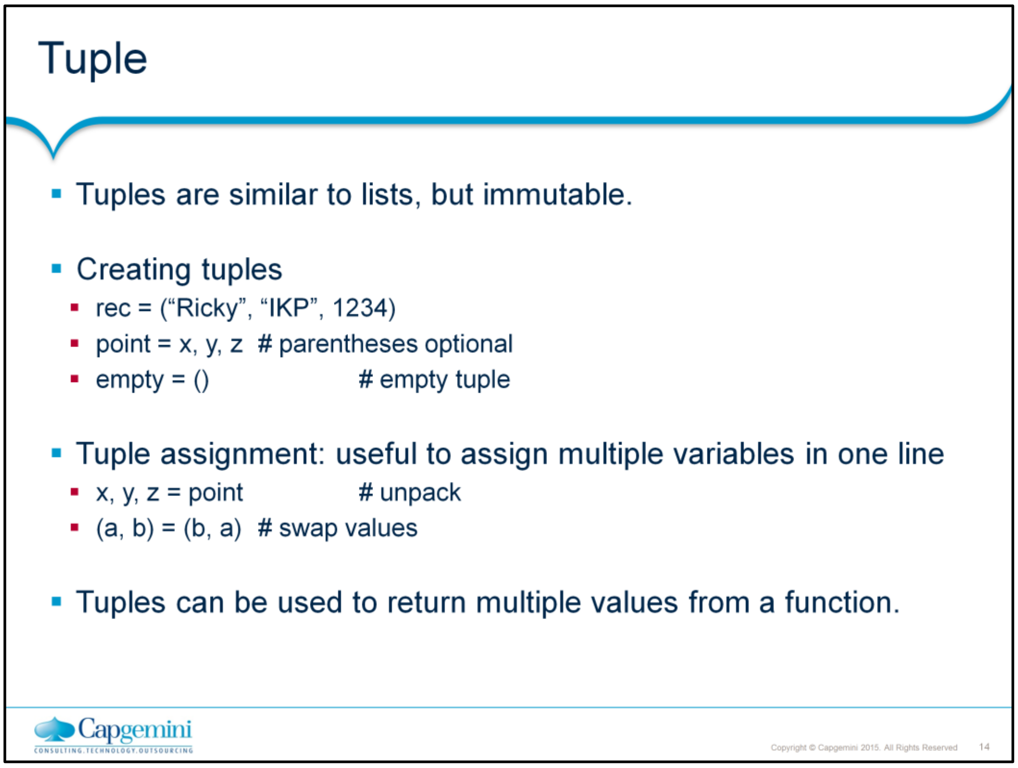
Instructor Notes:



List

- The “+” operator concatenates list and “*” operator repeats a list a given number of times.
- List Methods: Many in-built methods are available to work on lists.
  - **append, extend, pop, reverse, sort ……**
- The “pop” method will default pop the last element (LIFO), else can pop by passing the index
- Use “del” to delete an element from a list.

Instructor Notes:

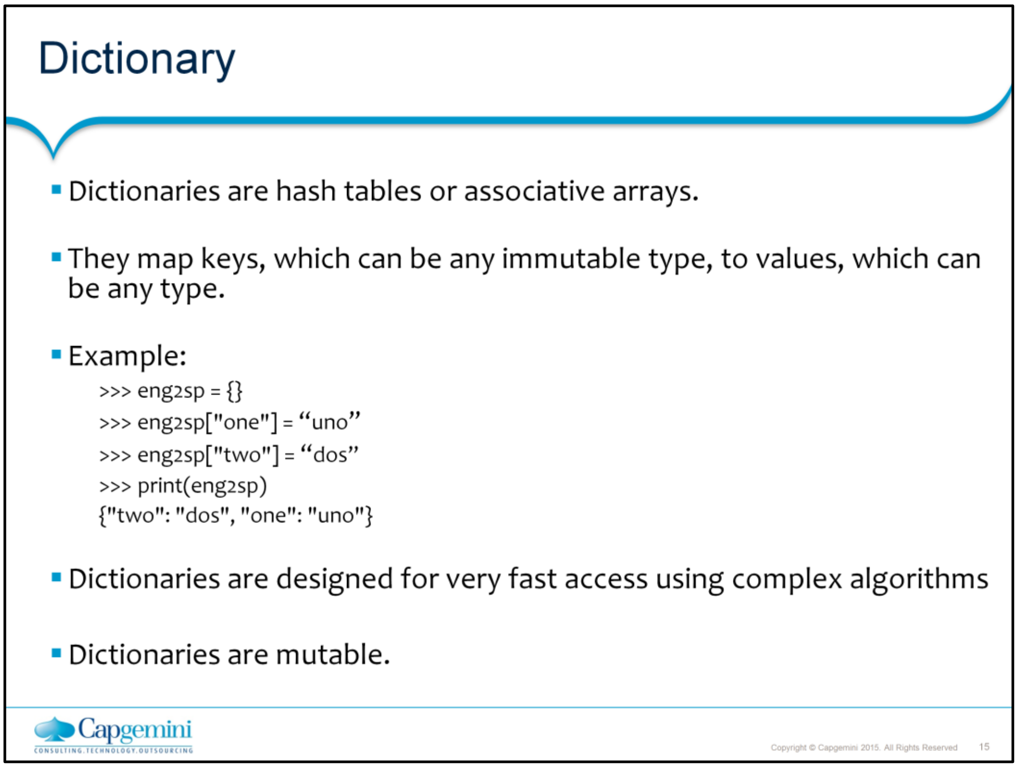


Tuple

- Tuples are similar to lists, but immutable.
- Creating tuples
  - rec = (“Ricky”, “IKP”, 1234)
  - point = x, y, z # parentheses optional
  - empty = () # empty tuple
- Tuple assignment: useful to assign multiple variables in one line
  - x, y, z = point # unpack
  - (a, b) = (b, a) # swap values
- Tuples can be used to return multiple values from a function.

14

Instructor Notes:



Dictionary

- Dictionaries are hash tables or associative arrays.
- They map keys, which can be any immutable type, to values, which can be any type.
- Example:

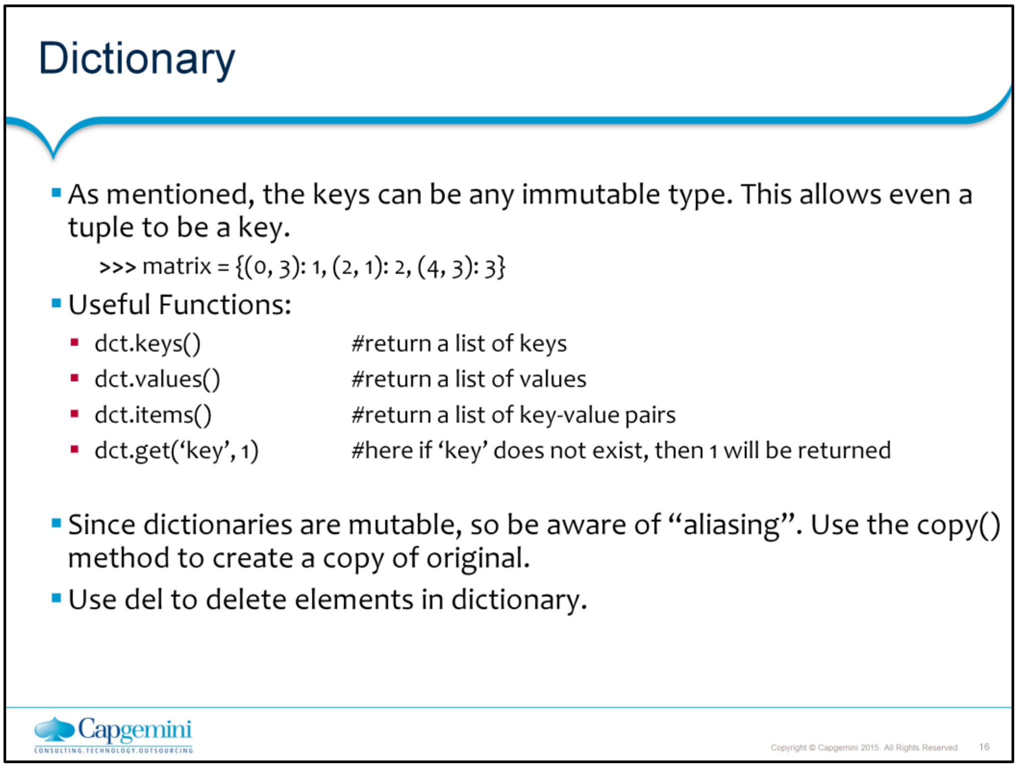
```
>>> eng2sp = {}
>>> eng2sp["one"] = “uno”
>>> eng2sp["two"] = “dos”
>>> print(eng2sp)
{"two": "dos", "one": "uno"}
```

- Dictionaries are designed for very fast access using complex algorithms
- Dictionaries are mutable.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

15

Instructor Notes:



Dictionary

- As mentioned, the keys can be any immutable type. This allows even a tuple to be a key.

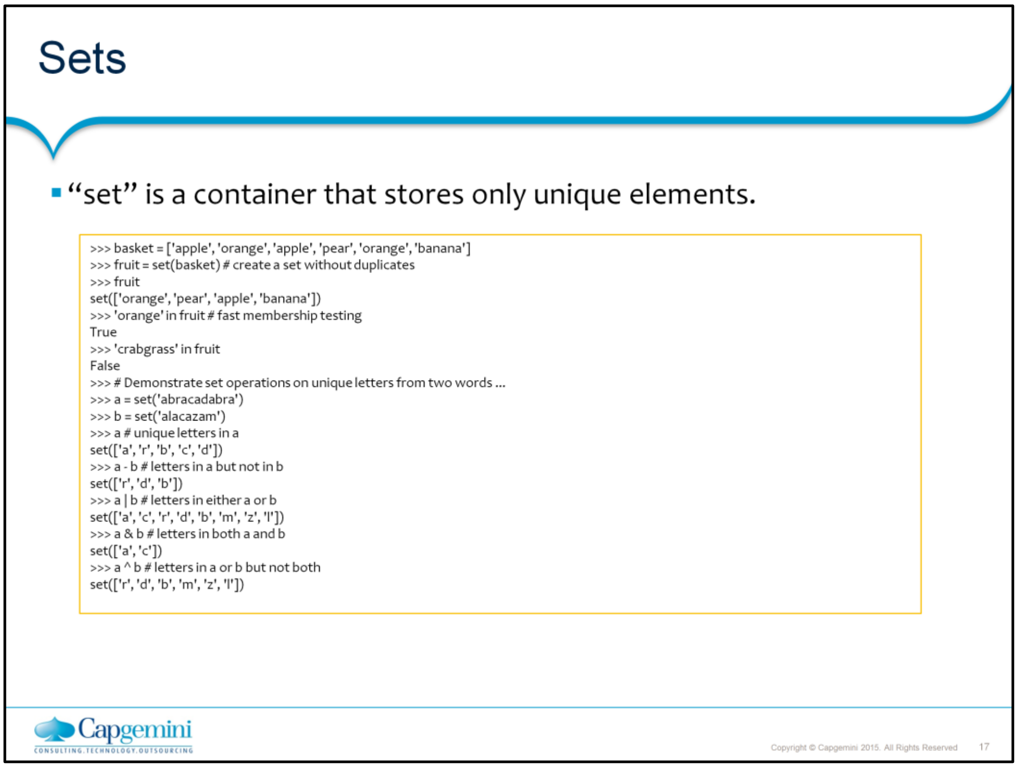
```
>>> matrix = {(0, 3): 1, (2, 1): 2, (4, 3): 3}
```

- Useful Functions:

dct.keys()	#return a list of keys
dct.values()	#return a list of values
dct.items()	#return a list of key-value pairs
dct.get('key', 1)	#here if 'key' does not exist, then 1 will be returned

- Since dictionaries are mutable, so be aware of "aliasing". Use the copy() method to create a copy of original.
- Use del to delete elements in dictionary.

Instructor Notes:



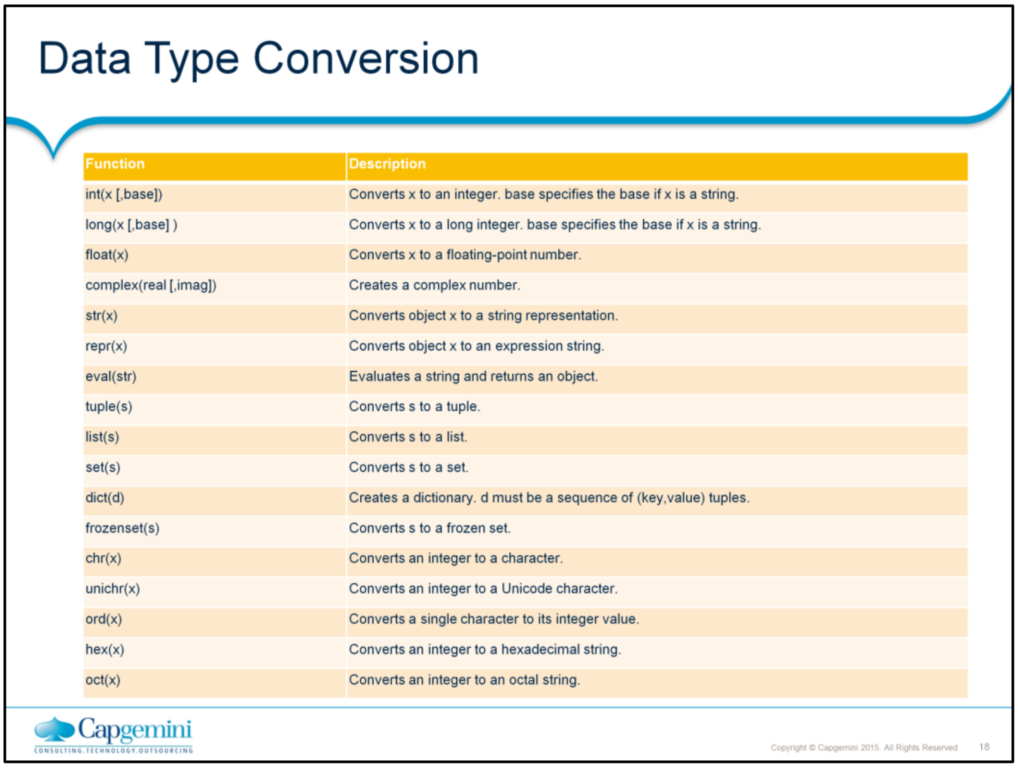
Sets

- “set” is a container that stores only unique elements.

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket) # create a set without duplicates
>>> fruit
set(['orange', 'pear', 'apple', 'banana'])
>>> 'orange' in fruit # fast membership testing
True
>>> 'crabgrass' in fruit
False
>>> # Demonstrate set operations on unique letters from two words ...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b # letters in both a and b
set(['a', 'c'])
>>> a ^ b # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

17

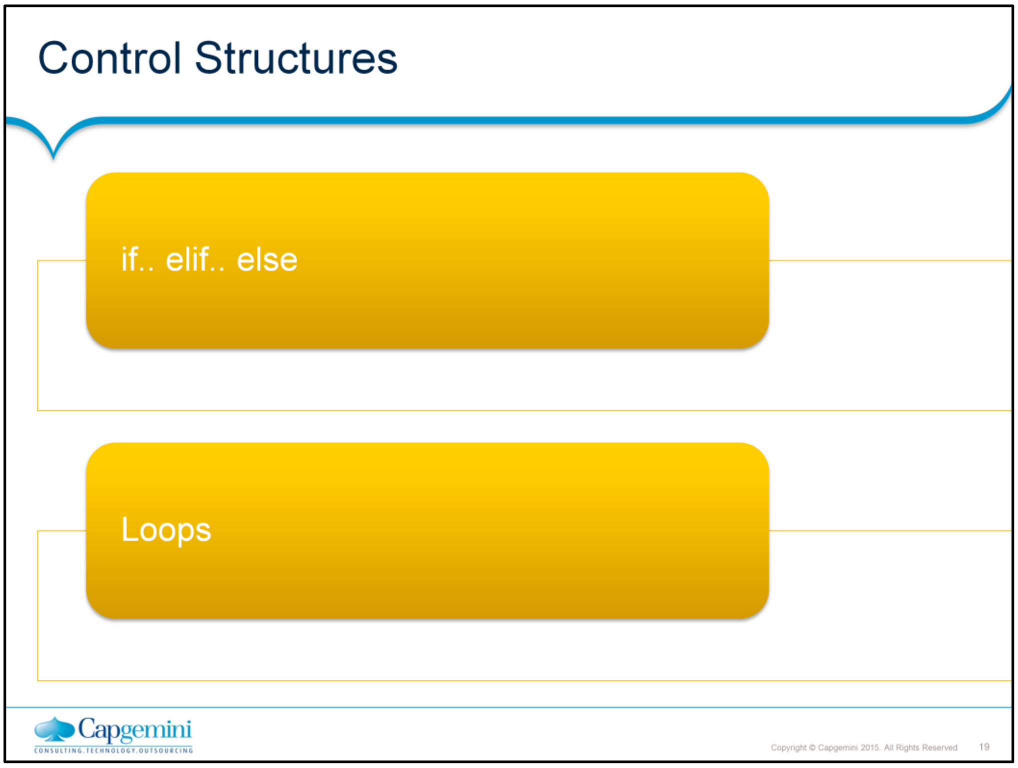
Instructor Notes:



Data Type Conversion

Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
long(x [,base])	Converts x to a long integer. base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

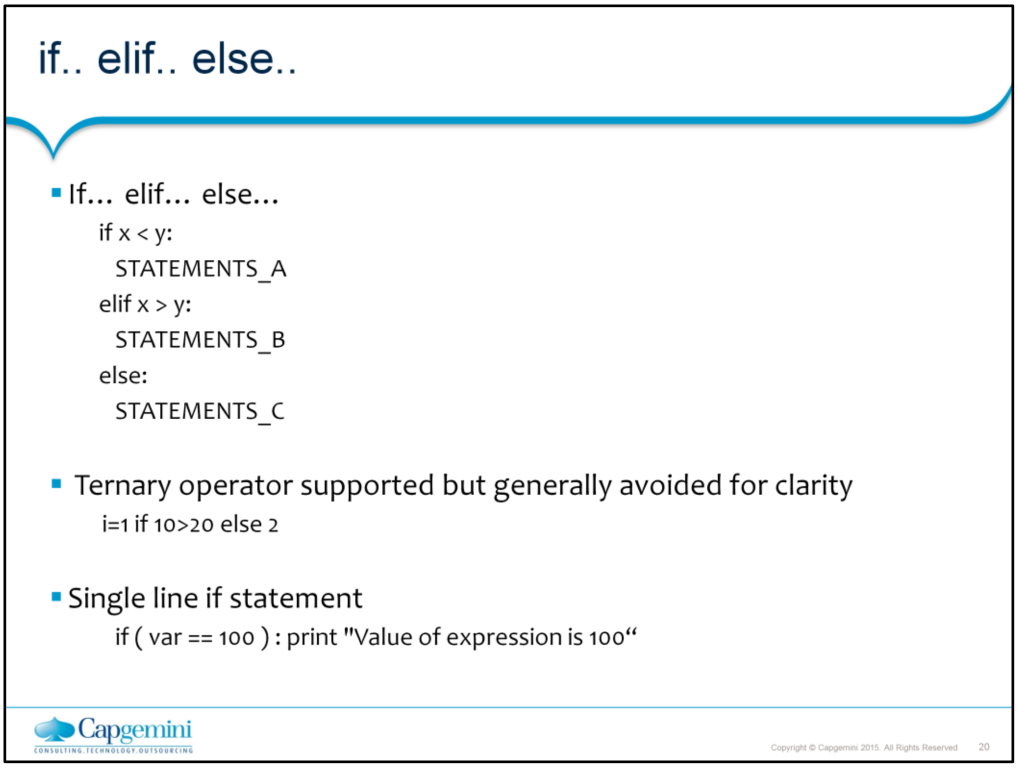
Instructor Notes:



Control Structures

- if.. elif.. else
- Loops

Instructor Notes:



if.. elif.. else..

- If... elif... else...

```
if x < y:
  STATEMENTS_A
elif x > y:
  STATEMENTS_B
else:
  STATEMENTS_C
```

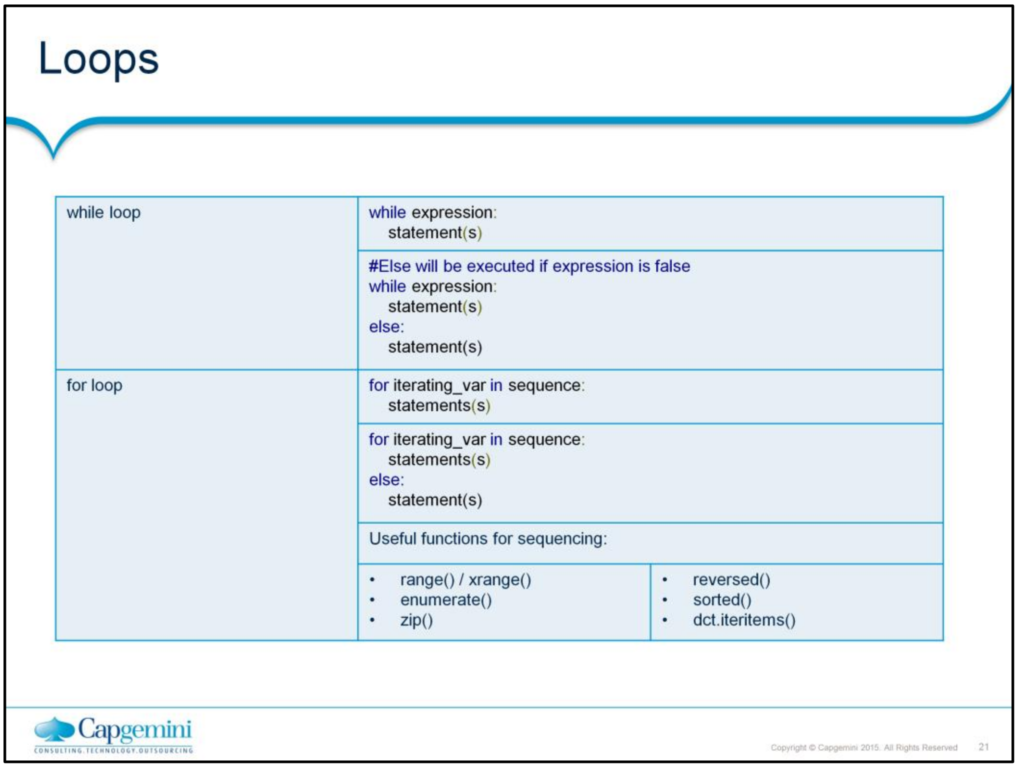
- Ternary operator supported but generally avoided for clarity

```
i=1 if 10>20 else 2
```

- Single line if statement

```
if ( var == 100 ) : print "Value of expression is 100“
```

Instructor Notes:



Loops

while loop	while expression:
	   statement(s)	

	#Else will be executed if expression is false
	while expression:
	   statement(s)
	else:
	   statement(s)	

for loop	for iterating_var in sequence:
	   statements(s)	

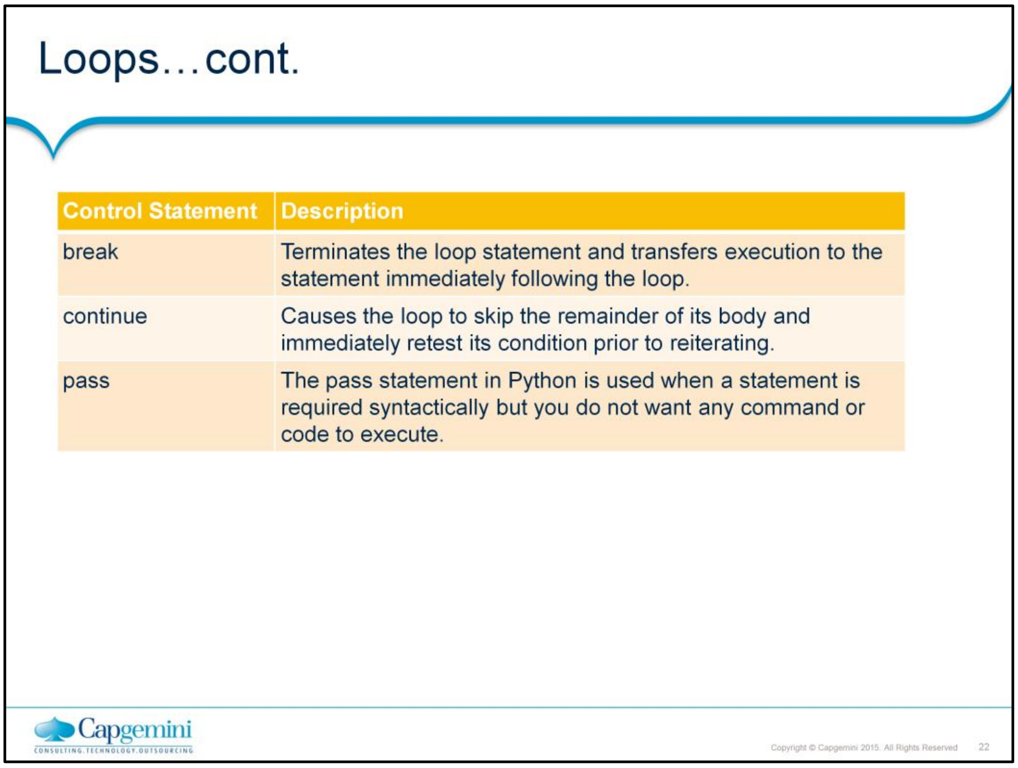
	for iterating_var in sequence:
	   statements(s)
	else:
	   statement(s)	

	Useful functions for sequencing:	
	• range() / xrange() • enumerate() • zip()	• reversed() • sorted() • dct.iteritems()

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

21

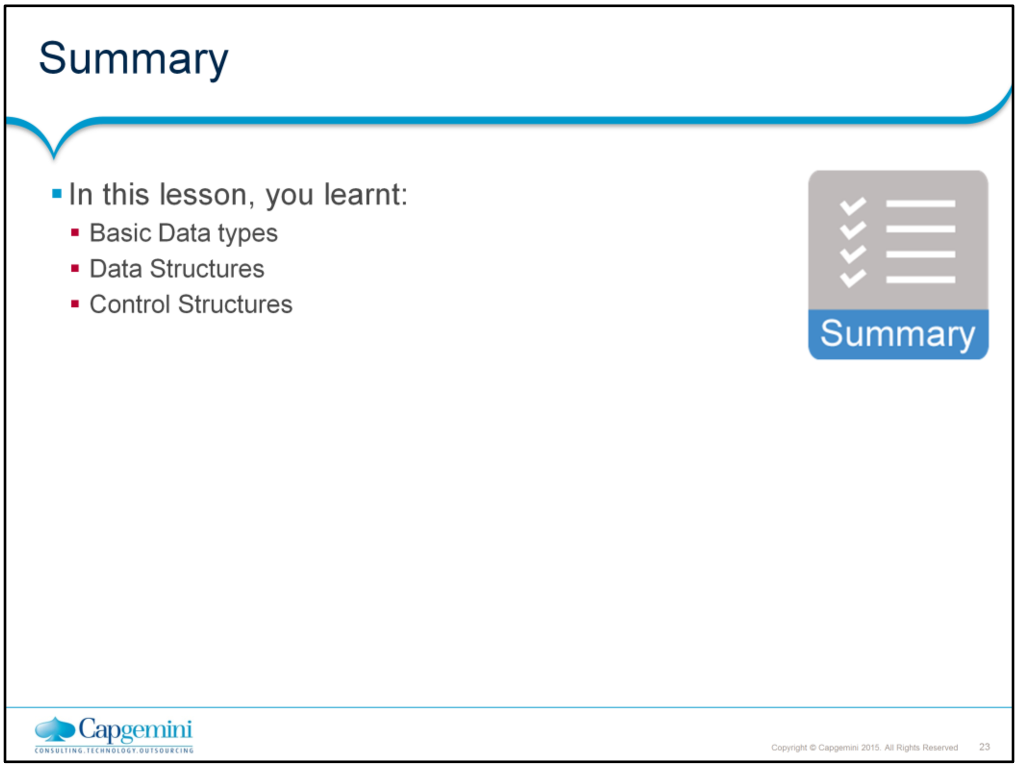
Instructor Notes:



Loops…cont.

Control Statement	Description
break	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
pass	The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Instructor Notes:



Summary

- In this lesson, you learnt:
  - Basic Data types
  - Data Structures
  - Control Structures