

Fundamentals	Looping	List Methods (cont)	Dictionary Methods
<pre>my_int = 5 my_string = " Hello World" my_othe_r_string = 'Hello e' Print to console: print(" Hello d! ") Collect input (string): my_input = inpu t(" Write: ") Addition: 5 + 3 Subtraction: 5 - 3 Multiplication: 5 * 3 Float division: 5 \ 3 Integer division (floors): 5 \ \ 3 Modulo: 5 % 3 Exponentiation: 5 ** 3 True: True False: False Type conversion: str(), int(), float(), bool() Detect the type: type() Equality of values: a == b Equality of references: a is b</pre>	<pre>for i in range(5): pri nt(i) # Prints 0 to 4 for i in ["Al pha ", " Bet a"]: Worl d pri nt(i) # Prints " Alp ha" and " - Beta" j = 0 while (j < 3): pri nt(j) # Prints 0 to 3 j = j+1</pre> <p>You can loop through an iterable, which can be a range, a list, a tuple, a dictionary using for or through a condition, using while.</p>	<p>Removes an item at an index: <code>lst.de l(i ndex)</code></p> <p>Gives the index of a given value: <code>lst.in dex (val, start, end)</code></p> <p>Gives the count of a given value: <code>lst.co unt (val)</code></p> <p>In-place reverses the list: <code>lst.re -verse()</code></p> <p>In-place sorts the list: <code>lst.sort()</code></p> <p>String aggregates: <code>sep.join(lst)</code></p> <p>Negative indices start from the end of the list, with <code>my_lis t[-1]</code>. Using <code>list()</code> converts an iterable to a list.</p>	
	<h3>List Comprehension</h3> <pre># Squares of even numbers less than 10 [i * i for i in range(10) if i % 2 == 0] # Multiple conditions (odds become negative) [i if i % 2 == 0 else -i for i in range(10)]</pre>	<h3>Slicing</h3> <pre>lst = [0, 5, 10, 15, 20] # Generic collec tion[s tar t:e - nd: step] collec tion[3] # 15 collec tion[1:] # [5, 10, 15, 20] collec tion[:2] # [0, 5] collec tion[::2] # [0, 10, 20] collec tion[:: -1] # [20, 15, 10, 5, 0]</pre> <p>Slicing can be applied to ordered collections. The start is inclusive and the end is exclusive.</p>	
<h3>Conditional Logic</h3> <pre>if (boolean_condition_1): res ult_1 elif (boole an_ con - dit ion_2): res ult_2 else: res ult_de - fault</pre> <p>If one block has been executed, the others cannot be executed, even if their relative condition is True.</p>	<h3>List Methods</h3> <p>Add an item to the end: <code>lst.ap -pend(item)</code></p> <p>Add a list of items to the end: <code>lst.ex ten d(l ist_of _item s)</code></p> <p>Insert an item at an index: <code>lst.i nser t(i ndex, item)</code></p> <p>Remove all items: <code>lst.cl ear()</code></p> <p>Remove & return an item at an index: <code>lst.po p(i ndex)</code></p> <p>Remove the first item with given value: <code>lst.re mov e(val)</code></p>	<h3>Zip</h3>	

