Lists are a mutable, data structure which can contain any type of data at indexed locations within the list.

The index locations start from zero and are referred to using integer numbers.

red	green	blue	white	orange
0	1	2	3	4

Items in a list can be referenced: **[start index: end index]**

The end index is the index **after** the one you want the range to finish on It is **up to but not including** the end index.

[1:3] refers to items at index 1 and 2 which is “green” and “blue”

List Syntax	**Description**	**Example**
listName = [value,value,value]	Assigns a set of values to a list.	myList= [“apple”,”oranges”,8]
listName[index]	Identifies an element of a list by reference to its position in the list, where index is an integer value starting at 0.	myList[2]
print(listName)	Displays a list on the screen	print(listName)
listName[start_index:end_index]	Reference an item in a list	print(listName[2:6] will display list items at locations 2 to 5
del listName[index] del listName[start_index:end_index]	Delete an item or range of items in a list	del listName[3] del listName[5:9]

Useful List methods	
list.append(x)	Appends the item x at the end of the list
list.sort()	Sorts the list (need to be a sortable data type)
list.reverse()	Reverses the elements in the list
list.count(x)	Counts the number of times x appears in a list
list.index(x)	Returns the index at which x appears
list.insert(index,x)	Inserts an item x at position before index

Range(start, stop, step)

Numbering starts at 0 and ends the item before “stop”

range(40)	Numbers from 0 to 39
range(5,15)	Numbers from 5 to 15
Range(10,100.5)	Numbers from 10 to 100 in steps of 5

Repetition		
Syntax	**Description**	**Example**

for variable in <expression>:
    <commands>
	Executes <commands> for a fixed number of times, given by <expression>.	myList=["cat","dog","cow","donkey","rabbit","canary"]
		  for next in myList:
		    print(next)

while <condition>:
    <commands>
	Executes <commands whilst <condition> is true. This is a pre-condition loop.	answer="N"
		counter=0
		while answer != "Y":
		    print("Are you hungry? You have been asked {0} times.".format(counter))
		    answer = input("Please respond Y or N:")
		    counter = counter + 1
		print("Please get something to eat!")

Symbol	Description
AND	Returns true if both conditions are true.
OR	Returns true if one of the conditions is true.
NOT	Reverses the outcome of the expression; true becomes false, false becomes true.

Subprogram	A small computer program that runs within another computer program. Subprograms are used to split up a program into a number of smaller programs, with each subprogram performing a specific function. Subprograms can be called in any order any number of times.
Function	A subprogram that returns a value.
Procedure	A subprogram that does not return a value.
Return value	The value returned by a subprogram.
Parameter	The names that appear in a function definition when passing data to a function.
Argument	A piece of information/value that is required by a function to perform a task, e.g. function(argument1, argument2)
Built-in subprograms	Pre-existing libraries of subprograms that are built into the programming language.
Library subprograms	Pre-existing libraries of sub-programs that can be imported and used in the programming language.