		string	**bytes**	**bytearray**	**tuple**	**list**	**set**	**frozenset**	**dictionary**
	Values	Single characters	Bytes (binary data)	Bytes (binary data)	Any	Any	Hashable	Hashable	Keys: Hashable; Values: Any
	Accessed	By index	By index	By index	By index	By index	Arbitrary	Arbitrary	By key
	Ordered	Yes	Yes	Yes	Yes	Yes	No	No	No
	Mutable	No	No	Yes	No	Yes	Yes	No	Yes
	Typical Uses	Text	Binary data	Buffer for data transfer	Immutable ordered data	Mutable ordered data	Unique unsorted data	Immutable set	Map from unique value to data
			image or video data	Transforming data in place	Parameter lists		Set operations		Lookup tables
					Positional data structures				Data structures
Getting		*Get a single value without modifying x*							
	x[index]	raises IndexError if out of bounds	🟩	🟩	🟩	🟩			
	x[key]								Raises KeyError if not found
	x.get(key)								Returns None if not found
	x.get(key, default)								Returns default if not found
Slicing		*Return a "slice" of x, truncates if out of range. List and ByteArray slices are assignable. "start" and "stop" start at 0, and can be negative, with -1 being the last item. Mutable types can assign to a slice.*							
	x[start:stop:stride]	🟩	🟩	🟩	🟩	🟩			
Testing		*Test for the presence (or not) of a value. Returns a boolean: "Is val in x"*							
	val in x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	Tests keys only
	val not in x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	Tests keys only
Finding		*Return the index of the (start of, for strings and bytes/bytearrays) first occurance of value: "Where is val in x?"*							
	x.find(val)	Returns -1 if not found	🟩	🟩					
	x.index(val)	raises ValueError if not found	🟩	🟩	🟩	🟩			
Inserting		*Add a new value to x*							
	x[key] = value								Same as updating
	x.insert(index, value)			🟩		🟩			
	x.append(val)			Inserts at the end		Inserts at the end			
	x.add(val)						🟩		
Updating		*Update a value in x*							
	x[key] = value								Same as inserting
	x[index] = value			🟩		🟩			
Extending		*Iterate over val, and append each item to x*							
	x.extend(val)			🟩		🟩			
Removing		*Delete (and sometimes return) val from x*							
	x.pop()			Remove last value		Removes last value	Removes arbitrary value		
	x.pop(key)								🟩
	x.remove(value)			🟩		ValueError if not found	KeyError if not found		
	del x[index]					IndexError if out of bounds			
	del x[key]								KeyError if not found
	x.discard(value)						🟩		
Listing		*Convert x to a list (or iterator)*							
	list(x)	🟩	🟩	🟩	🟩	🟩	🟩	🟩	Same as x.keys()
	x.keys()								Keys only
	x.values()								Values only
	x.items()								Acts like a list of (k, v) tuples
Iterating		Iterate over each value in x							
	for val in x	🟩	🟩	🟩	🟩	🟩	🟩	🟩	Iterates over keys