|  |  |
| --- | --- |
| **make**(T, len int[, cap int]) T | Works on slices, maps, and chan. |
| **new**(T) \*T | Value returned is a pointer to a newly allocated zero value of that type. |
|  |  |
| **append**(slice []T, elems… T) []T | Appends to slice, will return new or same slice back. |
| **copy**(dst, src []T) int | Copy from source to destination, return number of elements copied. |
|  |  |
| **len**(T) int | Length of array, slice, map or chan. |
| **cap**(T) int | Capacity of array, slice, map or chan. |
|  |  |
| **delete**(m map[T]T1, key T) | Delete element from the map. Check for presence idiom: x, ok := m[key] |
| **close**(c chan) | Close channel. Only senders should close. Check for closed channel: x, ok := <- c |
|  |  |
| **panic**(interface{}) **recover**() interface{} | For recover use idiom: defer func() {  if err := recover(); err != nil { /\* do something recovering \*/ }  } |

Slice tricks:

|  |  |
| --- | --- |
| a = **append**(a, b…) or a = **append**(a, x) | Append vector or element |
| b = **make**([]T, len(a));  sz := **copy**(b, a) | Copy sz is count of copied items |
| a = **append**(a[:i], a[j:]…) | Delete element or Cut vector out |
| a = **append**(a, **make**([]T, j)…) | Extend |
| a = **append**(a[:i], **append**([]T{x}, a[i:]…)…) | Insert element |
| a = **append**(a[:i], **append**(b, a[i:]…)…) | Insert vector |
|  |  |
| x, a = a[**len**(a)-1], a[:**len**(a)-1] | Pop last element |
| x, a := a[0], a[1:] | Pop first element |
| a = **append**([]T{x}, a…) | Prepend element |
|  |  |
| b := a[:0]  for \_, x := **range** a {  if f(x) { b = **append**(b, x) }  } | Filtering without allocation. Slice shares the same backing array and capacity as the original, so storage is reused for the filtered slice. After b := a[:0]; b points to the same array. |
| b := a[:0:2] b = **append**(b, x) b = **append**(b, x1, x2) | a and b shares same backing array. b has capacity of 2. x is being set into the same backing array so a and b both get x. now b detaches from a backing array since its capacity of 2 exceeded. |
| b := a[:4:5] b = **append**(b, x) b = **append**(b, x) | a and b shares same backing array. b has len=4, cap=5.  a and b both get x set at position [4]. now b detaches from a and no longer share same array. |
|  |  |
| for i,j := 0, **len**(a)-1; i < j; i, j = i+1, j-1 {  a[i], a[j] = a[j], a[i]  } | Reversing |
|  |  |

Channels:

|  |  |
| --- | --- |
| Closed channel | Never blocks. Always receives in “select”. Check if channel is closed: x, ok := <- c |
| nil channel | Always blocks. Always gets ignored in “select”. |
|  |  |

MIT license. Find this cheatsheet at <https://github.com/tadvi/go-cheatsheet>