# The Flix Cheetsheet

## Opt

An option Opt is either None or Some(v). *Options cannot be nested[[1]](#footnote-1).*

|  |
| --- |
| null(o: Opt[A]): Bool  Returns true iff o is None. |
| get(o: Opt[A]): A  Unsafely returns the value of the option o.  Note: If o can be None the semantics of the Flix program is undefined. |
| getWithDefault(o: Opt[A], a: A): A  Returns the value of the option o if it exists. Otherwise returns a. |
| map(f: A => B, o: Opt[A]): Opt[B]  Returns the result of applying f to the value of the given option o if it is exists and wraps the result in Some. Otherwise returns None. |
| map2(f: (A, B) => C, o1: Opt[A], o2: Opt[B]): Opt[C]  Returns the result of applying f to the value of the given options o1 and o1 if they exists and wraps the result in Some. Otherwise returns None. |
| flatMap(f: A => Opt[B], o: Opt[A]): Opt[B]  Returns the result of applying f to the value of given option o if it exists. Otherwise returns None. |
| flatMap2(f: (A, B) => Opt[C],  o1: Opt[A], o2: Opt[A]): Opt[C]  Returns the result of applying f to the value of given options o1 and o2 if they it exists. Otherwise returns None. |
| withDefault(o1: Opt[A], o2: Opt[A]): Opt[A]  Returns o1 if it is Some otherwise returns o2. |
| toList(o: Opt[A]): List[A]  Returns a one-element list of the value in the given option o if it exists. Otherwise returns the empty list. |
| toSet(o: Opt[A]): Set[A]  Return a one-element set of the value in the given option o if it exists.  Otherwise returns the empty set. |

## List

### Basic Operations

|  |
| --- |
| range(Int, Int)  Returns a list |
| repeat(A, Int) |
| permutations(List[A]): List  fdsfds |
| null(xs: List[A])  Returns true if xs is the empty list. |
| map(f: A => B, xs: List[A]): List[B]  Foo |
| flatMap(f: A => List[B], xs: List[A]): List[B] |

## Set

## Map

1. This ensures efficient representation of options. [↑](#footnote-ref-1)