



# Student Cheatsheet v1

## Creating a Vector

```
(vector 1 2 3)
;=> [1 2 3]
```

```
[1 2 3 4]
;=> [1 2 3 4]
```

## Vector Examples

```
(conj [5 10] 15)
;=> [5 10 15]
```

```
(count [5 10 15])
;=> 3
```

```
(nth [5 10 15] 1)
;=> 10
```

```
(first [5 10 15])
;=> 5
```

## Defining a value

```
(def name "Sally")
```

## Defining a function

```
(defn function-name
  "description of function, optional"
  [param1 param2]
  (function-body))
```

## Flow Control

```
(if conditional-expression
  expression-to-evaluate-when-true
  expression-to-evaluate-when-false)
```

## Logic Functions

```
(= x 4)
(> x 4) (>= x 4)
(< x 4) (<= x 4)
(and x y)
(or x y)
(not x)
```

## Creating a Map

```
(hash-map :a 1 :b 2)
;=> {:a 1, :b 2}
```

```
{:a 1 :b "two"}
;=> {:a 1, b "two"}
```

## Map examples

```
(get {:first "Sally" :last "Brown"} :first)
;=> "Sally"
```

```
(get {:first "Sally"} :last :MISS)
;=> :MISS
```

```
(assoc {:first "Sally"} :last "Brown")
;=> {:first "Sally", :last "Brown"}
```

```
(dissoc {:first "Sally" :last "Brown"} :last)
;=> {:first "Sally"}
```

```
(merge {:first "Sally"} {:last "Brown"})
;=> {:first "Sally", :last "Brown"}
```

```
(count {:first "Sally" :last "Brown"})
;=> 2
```

```
(keys {:first "Sally" :last "Brown"})
;=> (:first :last)
```

```
(vals {:first "Sally" :last "Brown"})
;=> ("Sally" "Brown")
```

## Let

```
(let [first-name (:first-name user)
      message (str "Hello, " first-name "!")]
  (println message))
```