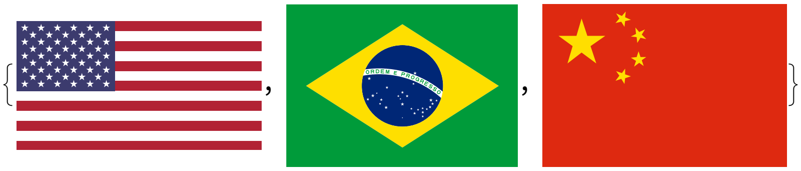
Harper’s Wolfram Language Cheat Sheet

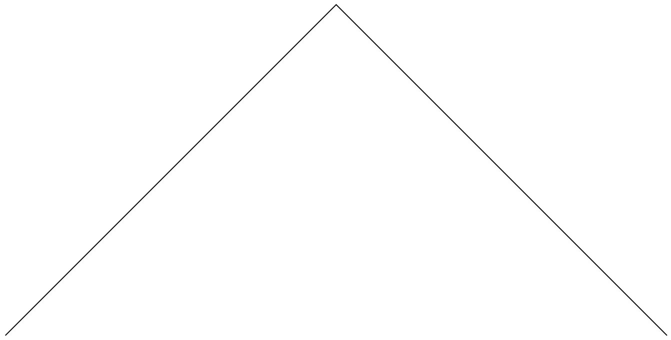
```
         (*What's the deal with all the @s?*)

         ToUpperCase@{"a", "b", "c"} (*@ is just like regular brackets*)
Out[∘]=
         {A, B, C}

In[∘]:=  ToUpperCase @@ {"a"}
Out[∘]=
         A

         Plus @@ {1, 2, 3} (*replaces curly brackets with normal brackets*)
Out[∘]=
         6

In[∘]:=  Plus @@@ {{1, 2}, {3, 4}}
Out[∘]=
         {3, 7}

In[∘]:=  Plus @@@ {1, 2, 3}
Out[∘]=
         {1, 2, 3}

In[∘]:=  Plus@{{1, 2}, {3, 4}}
Out[∘]=
         {{1, 2}, {3, 4}}

         ToUpperCase /@ {"a", "b", "c"} (*/@ applies to every element in a list*)
Out[∘]=
         {A, B, C}

         (*Lists and such*)

In[∘]:=  {1, 1, 2} * {1, 2, 3}
Out[∘]=
         {1, 2, 6}

In[∘]:=  Count[{a, b, a, a, c, b, a}, a]
Out[∘]=
         4

In[∘]:=  Transpose[{{1, 2}, {3, 4}}]
Out[∘]=
         {{1, 3}, {2, 4}}

In[∘]:=  Transpose[{{4, 5, 6, 7, 8}, {9, 10, 11, 12, 13}}]
Out[∘]=
         {{4, 9}, {5, 10}, {6, 11}, {7, 12}, {8, 13}}
```

```
In[•]:= Part[{1, 2, 3, 4, 5}, 5]
Out[•]=
        5

In[•]:= {1, 2, 3, 4, 5}⟦5⟧
Out[•]=
        5

In[•]:= {1, 2, 3, 4, 5}⟦3 ;; 5⟧
Out[•]=
        {3, 4, 5}

        (*associations*)

        <|1 → a, 2 → b, 3 → c|>

In[•]:= <|1 → a, 2 → b, 3 → c|>⟦2⟧
Out[•]=
        b

In[•]:= Sort[<|1 → a, 2 → b, 4 → d, 3 → c|>]
Out[•]=
        <|1 → a, 2 → b, 3 → c, 4 → d|>

In[•]:= KeySort[<|1 → a, 2 → b, 4 → d, 3 → c|>]
Out[•]=
        <|1 → a, 2 → b, 3 → c, 4 → d|>

        (*association with a pure function*)

In[•]:= f[#apples, #oranges] &[<|"apples" → 10, "oranges" → 12, "pears" → 4|>]
Out[•]=
        f[10, 12]

        (*arrays*)

        (*an array is a table with two axes*)

In[•]:= Grid[Table[i, {i, 4}, {j, 5}]]
Out[•]=
        1 1 1 1 1
        2 2 2 2 2
        3 3 3 3 3
        4 4 4 4 4

        (*dealing with real-world data*)
```

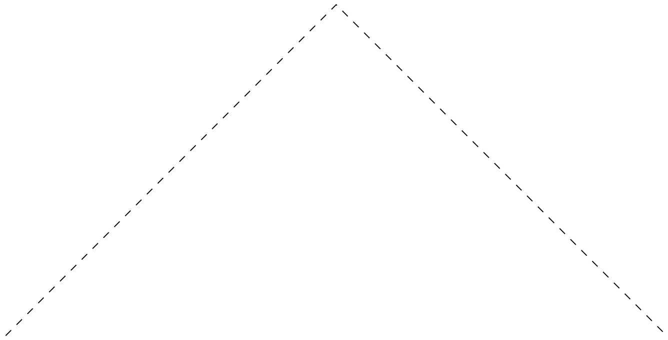
```
In[∘]:= EntityValue[{Entity["Country", "UnitedStates"],
          Entity["Country", "Brazil"], Entity["Country", "China"]}, "Flag"]
Out[∘]=
```



```
EntityValue[ United States COUNTRY ••• ✓ ""]

(*the above can tell any number of things*)

(*graphics tools*)

In[∘]:= Graphics[Line[{{1, 2}, {3, 4}, {5, 2}}]]
Out[∘]=
```



```
In[∘]:= Graphics[{Dashed, Line[{{1, 2}, {3, 4}, {5, 2}}]}]
Out[∘]=
```



```
(*Modules*)

In[∘]:= Module[{x = 3}, x^2]
Out[∘]=
        9

In[∘]:= Module[{x = Range[10], y = 2}, x y]
Out[∘]=
        {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
        (*multiplication by justaposition*)

        (*this works:*)

In[∘]:= Module[{x = Range[10], y = 2}, x y]
Out[∘]=
        {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}

        (*but this does not*)

In[∘]:= Module[{x = Range[10], y = 2}, xy]
Out[∘]=
        xy

        (*patterns*)

In[∘]:= MatchQ[{a, x, b}, {_, x, _}]
Out[∘]=
        True

In[∘]:= Cases[{{a, a}, {b, a}, {a, b, c}, {b, b}, {c, a}, {b, b, b}}, {_, _}]
Out[∘]=
        {{a, a}, {b, a}, {b, b}, {c, a}}

In[∘]:= EvenQ[3]
Out[∘]=
        False

In[∘]:= MatchQ[3, 3]
Out[∘]=
        True

In[∘]:= MatchQ[{3, 3}, {_, _}]
Out[∘]=
        True

        (*If statements*)

In[∘]:= Clear[x]

In[∘]:= Module[{x = RandomInteger[]}, If[OddQ[x], 3, 4]]
Out[∘]=
        3
```