Mathematica Cool Tips

Prefix (@), Postfix (//)

```
In[ ]:= f[x]
Out[ ]=
f[x]

In[ ]:= f@x
Out[ ]=
f[x]

In[ ]:= x // f
Out[ ]=
f[x]

In[ ]:= MatrixForm@{1, 2, 3, 4, 5, 6, 7, 8, 9}
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}$$

```
In[ ]:= {1, 2, 3, 4, 5, 6, 7, 8, 9} // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}$$

One-Time Function (#&)

```
In[ ]:= g[x_] := x^3

In[ ]:= g[x]
Out[ ]=
```

$$x^3$$

```
In[•]:= #^3 &[x]
```

Out[•]=

$$x^3$$

```
In[•]:= g[x_, y_] := x^2 * (y + 2)

In[•]:= g[x, y]
```

Out[•]=

$$x^2 \, (2 + y)$$

```
In[•]:= (#1^2 * (#2 + 2)) &[x, y]
```

Out[•]=

$$x^2 \, (2 + y)$$

Map, MapThread: Avoid using “For” or “Do”

Suppose you want to make list `{f[1],f[2],f[3],f[4],f[5],f[6]}`.

```
In[•]:= Table[f[k], {k, 6}]
```

Out[•]=

```
{f[1], f[2], f[3], f[4], f[5], f[6]}
```

`Map` may be more useful.

```
In[•]:= Map[f, Range[6]]
```

Out[•]=

```
{f[1], f[2], f[3], f[4], f[5], f[6]}
```

`Map` is far more general and handy.

```
In[•]:= kk = {a, b, c, d, e}
```

Out[•]=

```
{a, b, c, d, e}
```

```
In[•]:= Map[f, kk]
```

Out[•]=

```
{f[a], f[b], f[c], f[d], f[e]}
```

```
In[•]:= f /@ kk
```

Out[•]=

```
{f[a], f[b], f[c], f[d], f[e]}
```

Pure function (i.e., one-time function) is also handy in this respect.

```
In[•]:= kk = {a, b, c, d, e}
```

Out[•]=

```
{a, b, c, d, e}
```

```
In[•]:= Map[(2 + #^2) &, kk]
```

Out[•]=

$$\{2 + a^2,\ 2 + b^2,\ 2 + c^2,\ 2 + d^2,\ 2 + e^2\}$$

In fact, the above particular example can be achieved in a simpler way.

```
In[•]:= 2 + kk^2
```

Out[•]=

$$\{2 + a^2,\ 2 + b^2,\ 2 + c^2,\ 2 + d^2,\ 2 + e^2\}$$

```
In[•]:= Attributes[Power]
```

Out[•]=

```
{Listable, NumericFunction, OneIdentity, Protected, ReadProtected}
```

```
In[•]:= Attributes[Plus]
```

Out[•]=

```
{Flat, Listable, NumericFunction, OneIdentity, Orderless, Protected}
```

```
In[•]:= Sin[kk]
```

Out[•]=

```
{Sin[a], Sin[b], Sin[c], Sin[d], Sin[e]}
```

```
In[•]:= Attributes[Sin]
```

Out[•]=

```
{Listable, NumericFunction, Protected}
```

Suppose you have two lists.

```
In[•]:= Let[Complex, a, b]
aa = Array[a, 5]
bb = Array[b, 5]
```

Out[•]=

$$\{a_1,\ a_2,\ a_3,\ a_4,\ a_5\}$$

Out[•]=

$$\{b_1,\ b_2,\ b_3,\ b_4,\ b_5\}$$

You want to construct a new list such that
$\{F[a_1, b_1], F[a_2, b_2], F[a_3, b_3], F[a_4, b_4], F[a_5, b_5]\}$.

```
In[•]:= MapThread[F, {aa, bb}]
```

Out[•]=

$$\{F[a_1, b_1],\ F[a_2, b_2],\ F[a_3, b_3],\ F[a_4, b_4],\ F[a_5, b_5]\}$$

Apply (@@), MapApply (@@@)

```
In[•]:= Clear[f, g]
```

Suppose you have a function with head **g**.

```
In[•]:= g[x, y]
Out[•]=
g[x, y]
```

You can replace the head of the function with any other head, say, **F**.

```
In[•]:= F @@ g[x, y]
Out[•]=
F[x, y]
```

Suppose you have a list of functions with different heads. You want to replaced all those heads to, say, **F**.

```
In[•]:= {f[x], g[x], h[x, y]}
Out[•]=
{f[x], g[x], h[x, y]}
```

MapApply does it for you.

```
In[•]:= F @@@ {f[x], g[x], h[x, y]}
Out[•]=
{F[x], F[x], F[x, y]}
```

```
In[•]:= Let[Qubit, S]
```

```
In[•]:= bs = Basis[S@{1, 2}]
Out[•]=
{|␣⟩, |1_S2⟩, |1_S1⟩, |1_S1 1_S2⟩}
```

```
In[•]:= InputForm[bs]
Out[•]//InputForm=
{Ket[<||>], Ket[<|S[2, $] -> 1|>], Ket[<|S[1, $] -> 1|>], Ket[<|S[1, $] -> 1, S[2
```

```
In[•]:= Bra @@@ bs
Out[•]=
{⟨␣|, ⟨1_S2|, ⟨1_S1|, ⟨1_S1 1_S2|}
```

```
In[•]:= InputForm[%]
Out[•]//InputForm=
{Bra[<||>], Bra[<|S[2, $] -> 1|>], Bra[<|S[1, $] -> 1|>], Bra[<|S[1, $] -> 1, S[2
```

Summary

Functions

- Table
- **Apply**
- **Map**

- `Thread`
- `MapThread`
- `Through`
- `@, @@, @@@`
- `f@x, f[x], x//f`

Related Links

- S. Wolfram (2017), “An Elementary Introduction to Wolfram Language,” 2nd edition (2017).
- The Wolfram Language: Fast Introduction for Math Students
- The Wolfram Language: Fast Introduction for Programmers