concat _____

syntax: `concat(first, second)`

purpose: Joins two vectors or arrays to make a single vector or array.

```
≫    first = [1 2 3 4];
≫    second = [5 6 7 8 9 10];
≫    together = concat(first, second)
    together:   [1 2 3 4 5 6 7 8 9 10]
```

examples: Suppose that you have data from two groups, say a control group and the experimental group:

```
≫    control = [4.4 6.7 8.6 3.4 7.4];
≫    expt = [9.3 8.4 4.7 5.5 5.2 6.2];
```
We want to find out whether the mean of experimental group is statistically different from the mean of the control group. First, compute the difference in means:

```
≫    diffmean = mean(expt) - mean(control)
    ans:      -4.5
```
As a Null Hypothesis, we will take that the two groups are not different. We will simulate a process where we combine the groups and then randomly pick mock control and experimental groups from the controlled data. We'll see how often these random groups differ by more than the amount we found for the original data.

```
z = starttally;
Ntrials = 1000;
 % how big are the groups?
exptlen = length(expt);
controllen = length(control);

newdata = concat(expt, control);

for trials = 1:Ntrials
  mockexpt    = sample(exptlen, newdata);
  mockcontrol = sample(controllen, newdata);
  a = mean(mockexpt) - mean(mockcontrol);
  score a z;
end
```
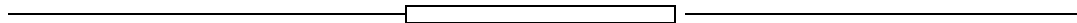
```
disp('Fraction trials exceeding sample difference.');
count(z>diffmean)/Ntrials
 % Note: this is a one-sided test.
```

For MATLAB experts: `concat` will work for both vectors and matrices. If both arguments are vectors, it concatenates them to produce a longer vector. This will be a row vector or a column vector, depending on the shape of `first`.

If one or both of the arguments are a matrix, `concat` will try to find a compatible direction for the concatenation. This will perhaps involve taking the transpose of the second argument.