```
from sympy import symbols, diff
```

Amazing module for handling math.

Use this with Python functions to turn them into ~~fake~~ formulas

Together with "symbols", this can differentiate from within your code.

```
a, b = symbols('x, y')
f(a,b)  # Returns the ~~fake~~ function
diff(f(a,b), a) # Returns the derivative with respect to "x".
```

Here "a" will represent "x" and "b" will represent "y".

```
# To evaluate a derivative at a given point:
diff(f(a,b), a).evalf(subs={a:1.8, b:1.0})
```

you pass in the key-value pairs to substitute

To avoid writing this awfully long code every time, we can save this into a function to evaluate it in a easier way:

```
dfx = lambdify([a, b], diff(f(a,b), a))
```

Iterable

Expression

Now you can easily call it directly

```
dfx(1.8, 1.0)
```

I initially thought lambdify wouldn't make much of a difference.
I was wrong!

10K iterations with lambdify is way faster than 1000 differentiating every time (in the gradient descent algorithm).