

# Ayudantia Guía 6 PNG 2021

## Funciones en Matlab

Alex Villarroel Carrasco

Universidad de Concepción

*avillarroe2019@udec.cl*

4 de Mayo de 2021



# Tabla de Contenidos

- 1 Funciones
- 2 Funciones de Ayuda
- 3 Ejercicios



# ¿Cómo funcionan?

A menudo resulta útil generar funciones, debido a que pueden ser llamadas y utilizarlas varias veces, ahorrando tiempo y líneas de código.

Cuándo ocupan comandos tales como `mean`, `sum`, etc, lo que hacen es llamar a una función determinada que realiza esos cálculos, por lo tanto, una función consta generalmente de una entrada y de una salida, siendo la entrada lo que uno ingresa y la salida el resultado de la función. **Una función puede tener más de una variable en la entrada y en la salida.**

Ex:

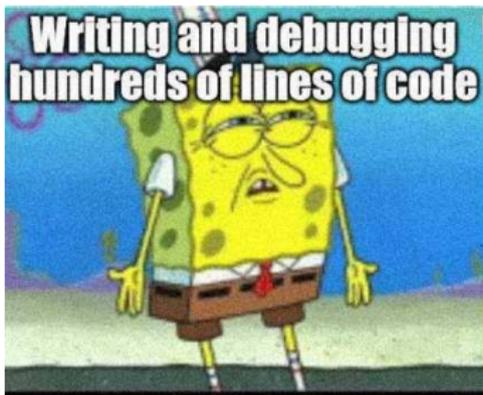
```
function [m,s] = stat(x)
n = length(x);
m = sum(x)/n;
s = sqrt(sum((x-m).^2/n));
end
```

# A considerar

- El nombre del archivo que contiene la función debe llamarse tal cual nombraste la función.
- Deben pensar en los errores que puede cometer la persona al ingresar una entrada, y anticiparlos.
- Pueden ocupar otras funciones dentro de su propia función 😊



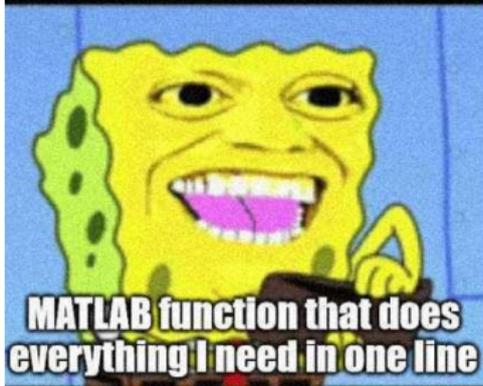
## F



```

21 % ...
22 dA_curr = dA_prev
23
24 activation = nn_architecture{1}('activation')
25 W_curr = parameter{'W' = str(1)}
26 Z_curr = forward_cache{'Z' = str(1)}
27 A_prev = forward_cache{'A' = str(1)}
28
29 dA_prev, dZ_curr, dW_curr = linear_activation_backward
30
31 grads{dW} = str(1)} = dZ_curr
32 grads{dZ} = str(1)} = dW_curr
33
34 return grads
35
36 def linear_activation_backward(dA, E, A_prev, W, activation)
37 if activation == 'relu':
38     dZ = relu_backward(dA, Z)
39     dA_prev, dW, dZ = linear_backward(dZ, A_prev, W)
40 elif activation == 'sigmoid':
41     dZ = sigmoid_backward(dA, Z)
42     dA_prev, dW, dZ = linear_backward(dZ, A_prev, W)
43
44 return dA_prev, dW, dZ
45
46 ...
47 ...
48 ...
49 ...
50 ...
51 ...
52 ...
53 ...
54 ...
55 ...
56 ...
57 ...
58 ...
59 ...
60 ...
61 ...
62 ...
63 ...
64 ...
65 ...
66 ...
67 ...
68 ...
69 ...
70 ...
71 ...
72 ...
73 ...
74 ...
75 ...
76 ...
77 ...
78 ...
79 ...
80 ...
81 ...
82 ...
83 ...
84 ...
85 ...
86 ...
87 ...
88 ...
89 ...
90 ...
91 ...
92 ...
93 ...
94 ...
95 ...
96 ...
97 ...
98 ...
99 ...
100 ...

```



- `isa(A,dataType)` , devuelve 1 si es verdadero(la variable A es de tipo 'dataType') y 0 si es falso.
- `nargin` sirve para saber cuantos elementos se han ingresado a la función
- `try catch` , intenta hacer algo que se le asigna en `try` y encuentra los errores en `catch`(Veámoslo mejor en Matlab...)



# Vámonos a Matlab!

Los archivos con los ejercicios resueltos en las prácticas se encuentran en los archivos del canal de Teams.

