functional programming and object oriented diagram summary- 1

Functional programming: for large, and or nested calculations and data management
Ie: excel, react, javascript, python, c#, other

Object oriented programming: large systems with lots of dependent functions and code reuse..
Ie: Javascript, python, c++/c, c#, java, other

Functional programming diagram example:

non-functional programming
---
func1(data1,date2,data3,...)
func2(data1,date2,data3,...)
func3(data1,date2,...)
func4(data1,date2,...)
func4(data1,date2,...)

functional programming -- group of functions by name
---
function_group_name_func1(data1,date2,data3,...)
function_group_name_func2(data1,date2,data3,...)
function_group_name_func3(data1,date2,...)
function_group_name_func4(data1,date2,...)
function_group_name_func4(data1,date2,...)

Nested functional programming example

- Output from 1 function is input to another function ie:function chaining
- Functions does not modify input data and returns new/modified output data

avg(sum))
array.map.reduce

Object Oriented diagram example

non-oop
---
func1(data1,date2,data3,...)
func2(data1,date2,data3,...)
func3(data1,date2,...)
func4(data1,date2,...)
func4(data1,date2,...)

oop -- group of functions
---

class/object1
 data1,date2,data3,...
 func1()
 func2(data1,date2,data3,...)

class/object2
 data1,date2,...
 func3(data1,date2,...)
 func4(data1,date2,...)

class/object3
 data1,date2,...
 func4(data1,date2,...)

class/object4 extend object1 - re-use code
 data...
 funca()
 fucca(data...)

--- creating objects and data .....

#object in own memory space
var1 = object1(data1,data2,...)
var1.func1

#object in own memory space
var2 = object1(data1,data2,...)
var2.func1

#object in own memory space

```
var3 = object2(data1,data2,...)
var3.func1

#object in own memory space
var3 = object4(data1,data2,...)
var3.funca

#function does not remain in memory
#functions run and exit from memory
```