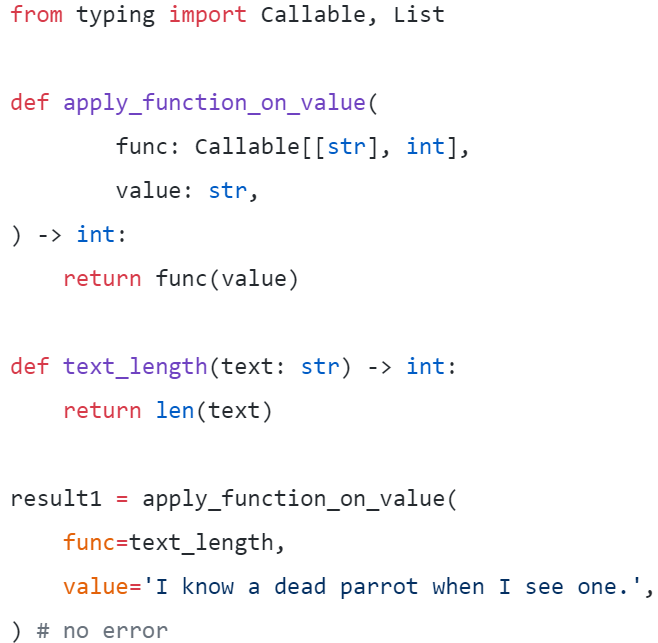
Python understanding

**Typing in Python**: There are two things basically, classes and types, classes are more of runtime concepts, and types depend on type checker (mypy in python) which checks whether your code has proper types and minimizes possible bugs

|  |
| --- |
|  |
| AnnealFunc = Callable[[Number,Number,float], Number] |
|  |

In python’s typing system functions have types just like any other values. In fact, function’s type is, in some way, very similar to Dict type, which “maps” one value to another; like Dict[str, int] maps strs to ints. From a typing point of view, a function is more complex — there can be none, one or multiple arguments and dict has only one key. However, the very mapping idea is the same. In Python to describe types of functions (and other callables) Callable type is used.



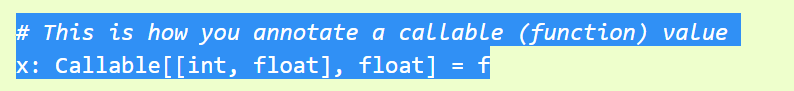
apply\_function\_on\_value takes a function func as its first parameter and applies it on its second parameter, value. This function’s type is “str to int”, or Callable[[str], int]. So passing text\_length function (which is just Python’s len defined on strings) to it is correct because func is defined as taking a str and returning an int.

The function arguments is the first list which contains types of each parameter and the other type is return type of the function

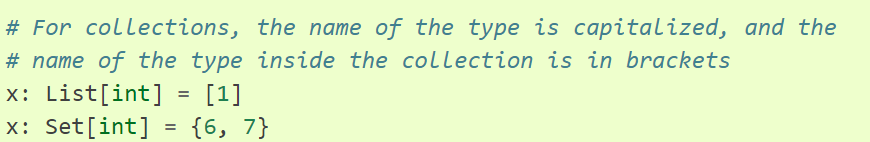
Here [**Number,Number,float**] are the types of parameter and **Number** is the return type

**Functions used from typing system**

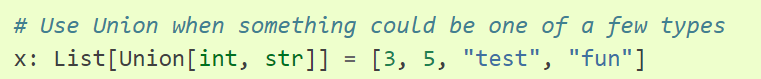
1. **Callable**: Assigns the type of a function to a variable



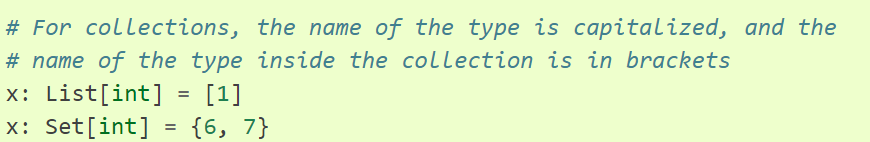
1. **Collection**:



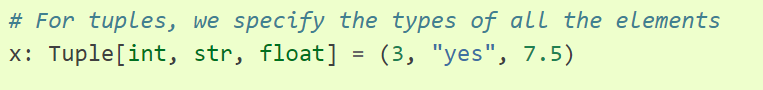
1. **Union**



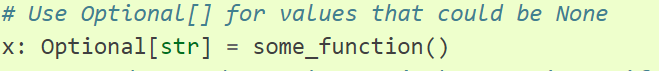
1. **List**



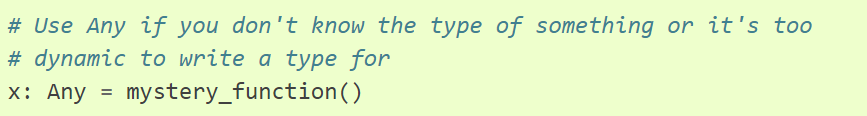
1. **Tuple**



1. **Optional**



1. **Any**

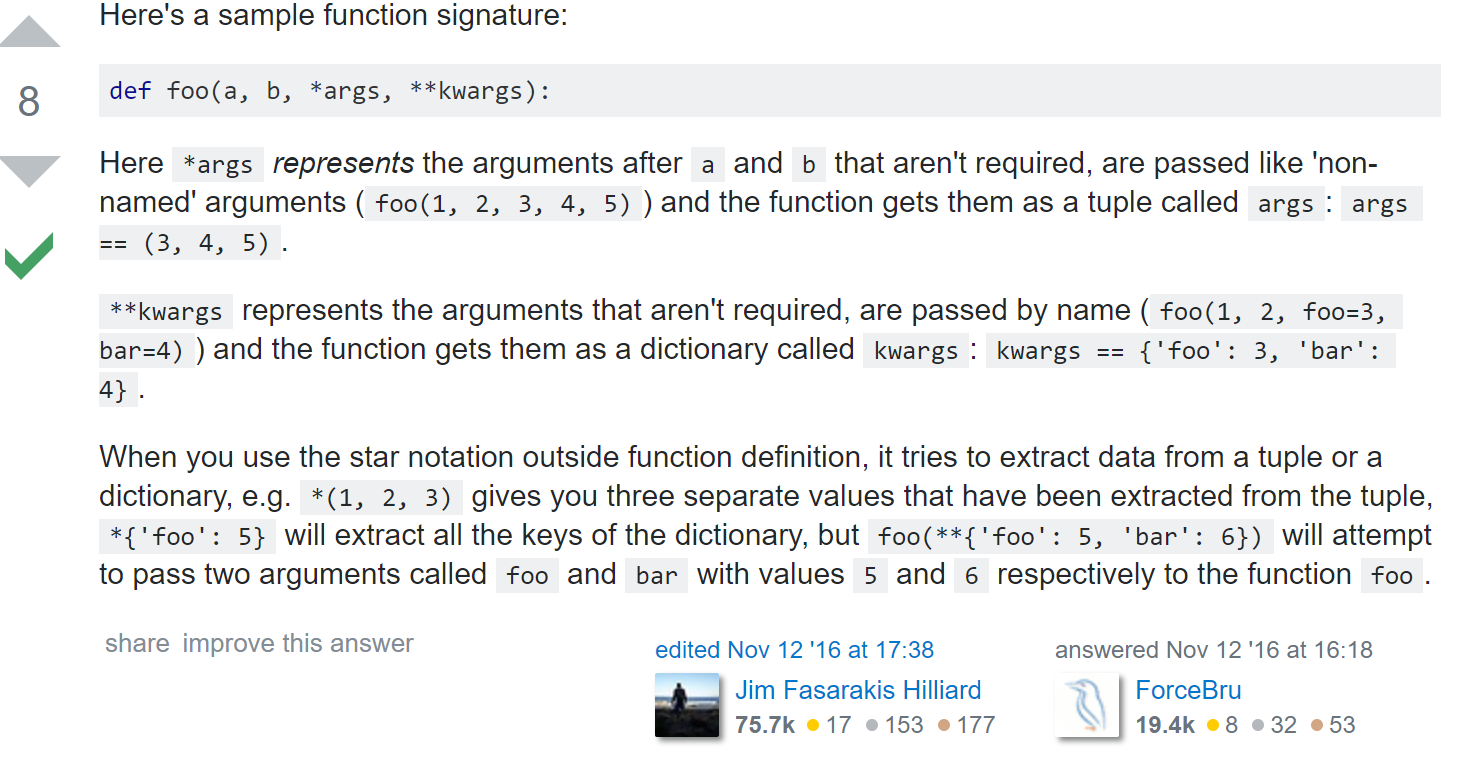


1. **SimpleNamespace**

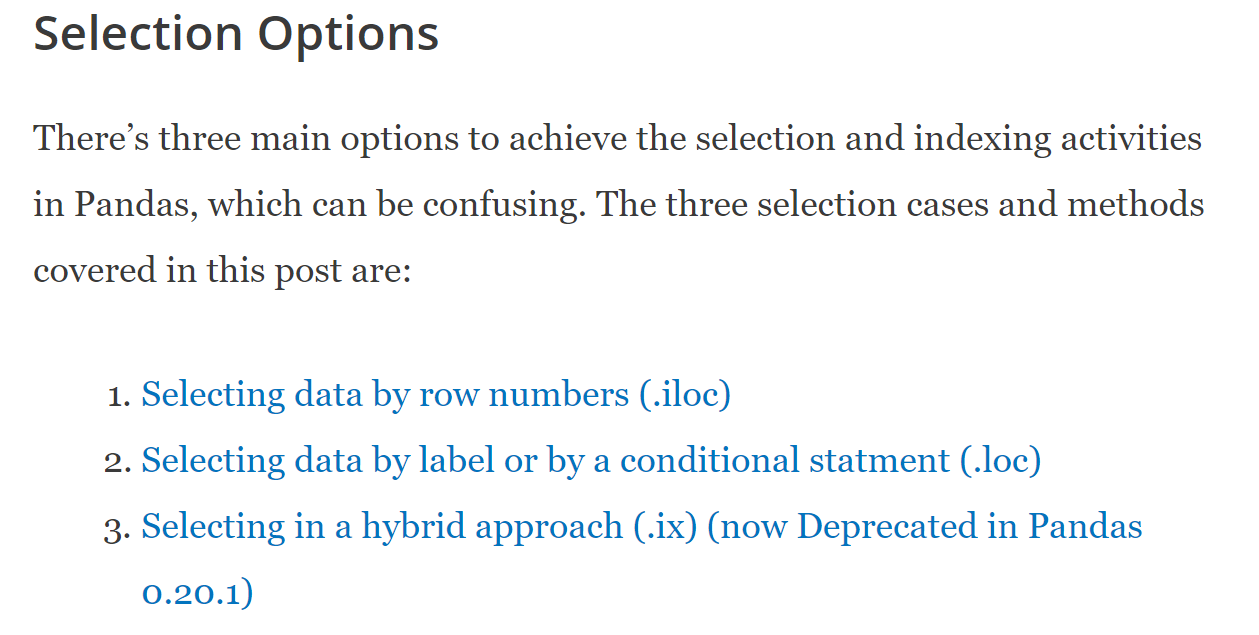
It allows you to initialize attributes while constructing the object: sn = SimpleNamespace(a=1, b=2)

**General Standard Library Functions**

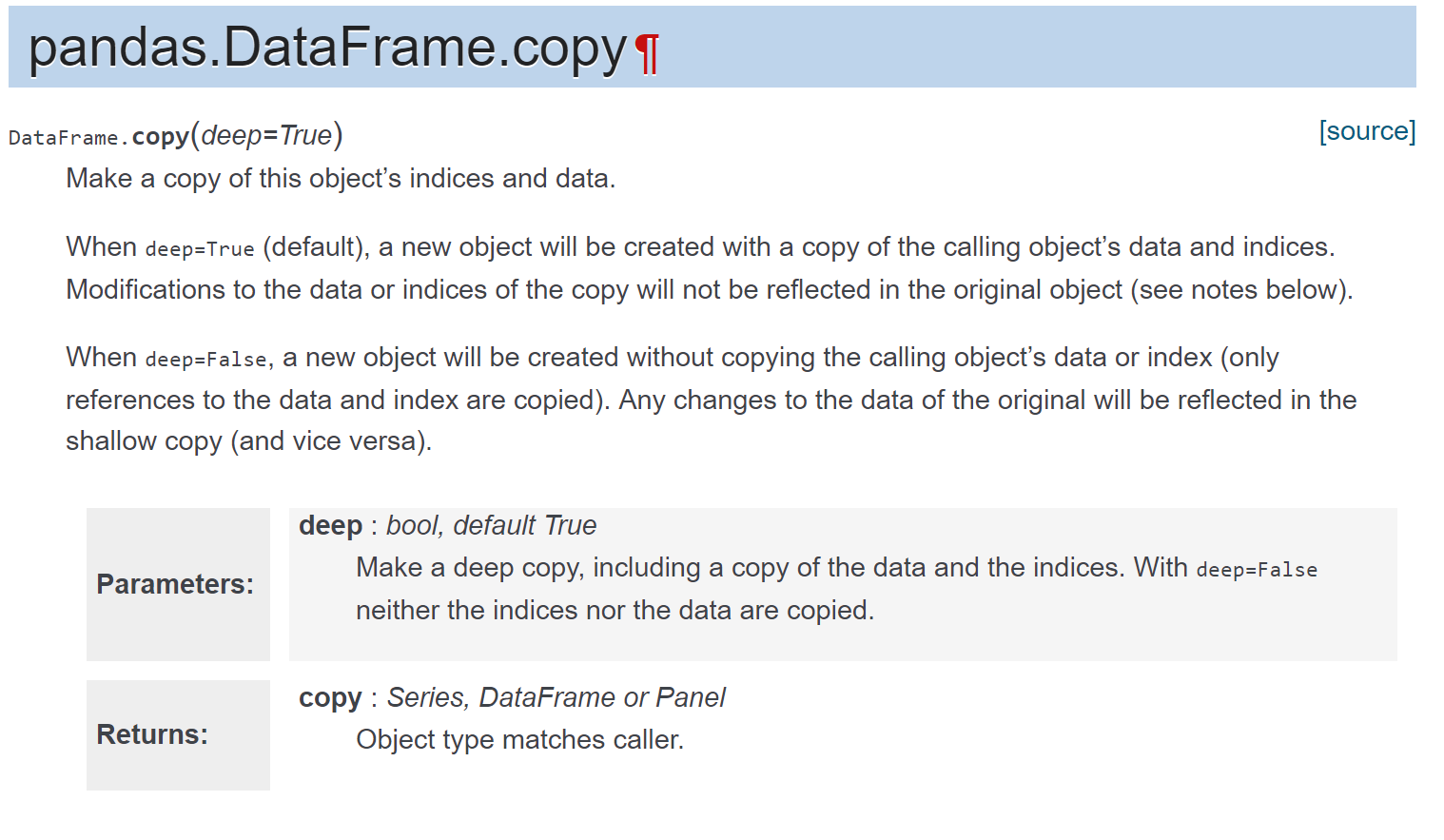
1. **isinstance()** : The isinstance() function checks if the object (first argument) is an instance or subclass of classinfo class (second argument).
2. **\*args and \*\*kwargs**



1. iloc , loc and .ix



1. **Deep Copy vs Shallow Copy**

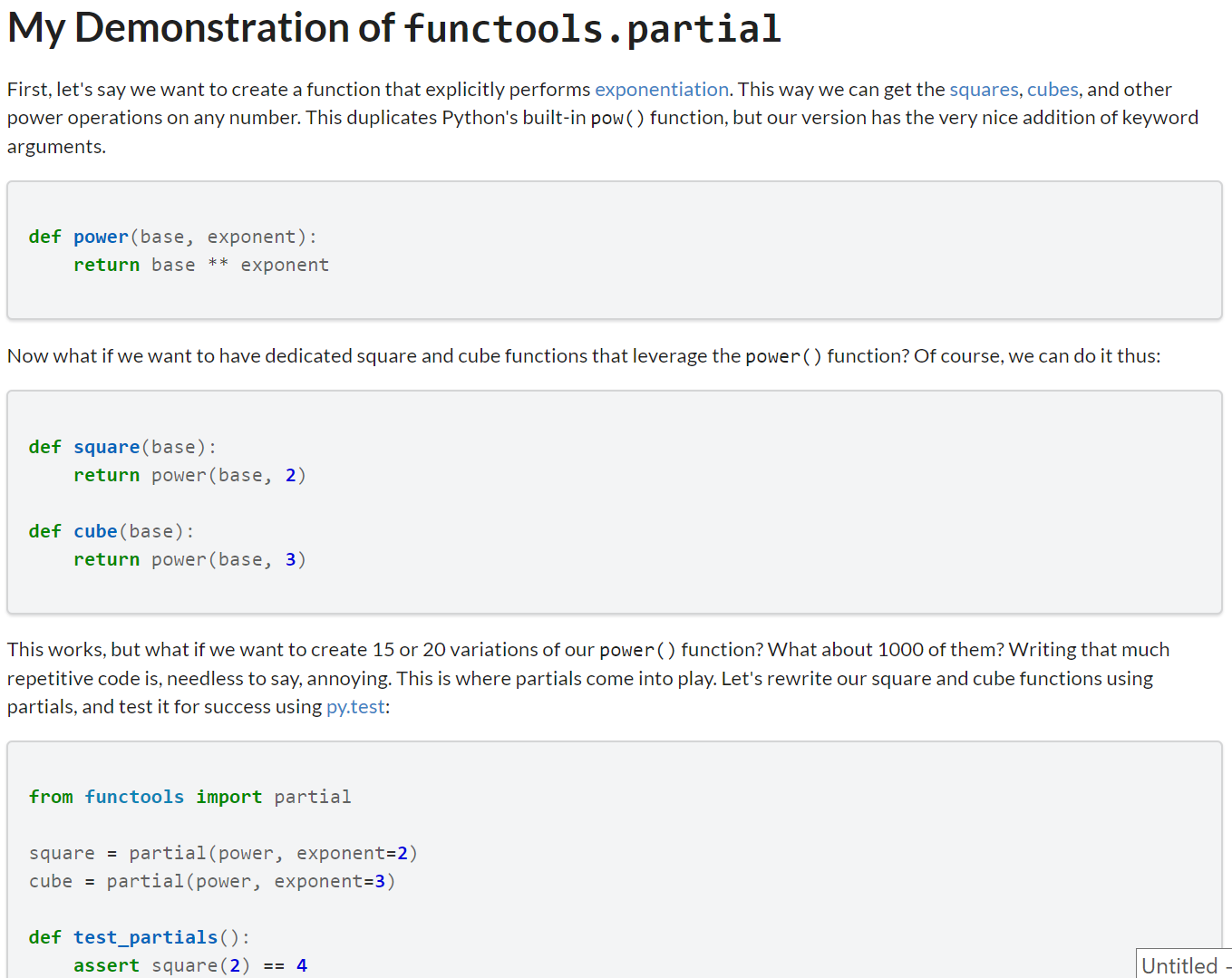


**General Programming Concepts**

1. **Synchronous Vs Asynchronous**



1. **Python Partials**



1. **Static Vs class methods Vs instance methods**

* **Instance methods** need a class instance and can access the instance through self.
* **Class methods** don’t need a class instance. They can’t access the instance (self) but they have access to the class itself via cls.
* **Static methods** don’t have access to cls or self. They work like regular functions but belong to the class’s namespace.

1. **Bound Vs Unbound**

The key difference between unbound and bound methods is that an unbound method is not associated with a particular instance, while a bound method is.

**Styling and Syntax**

* camelCase
* PascalCase
* snake\_case
* kebab-case

**References**

<https://blog.daftcode.pl/next-steps-with-python-type-system-efc4df5251c9>