OOP

special methods

https://docs.python.org/3/reference/datamodel.html#special-method-names

- __new__
- __init__ - called after the instance has been created by **new**
- __del__ - finalizer: called when the instance is about to be destroyed.
  - It is not guaranteed that del() methods are called for objects that still exist when the interpreter exits.
  - **Note**: `del x` doesn't directly call `x.__del__()` — the former decrements the reference count for x by one, and the latter is only called when x's reference count reaches zero.
- __repr__ - Called by the `repr()` built-in function to compute the "official" string representation of an object. This is typically used for debugging, so it is important that the representation is information-rich and unambiguous
- __str__ - Called by `str(object)` and the built-in functions `format()` and `print()` to compute the "informal" or nicely printable string representation of an object.
- bytes
- format
- lt / le / eq / ne / gt / ge
- hash
- bool

call super constructor

```
class A(object):
    def __init__(self):
        print("world")

class B(A):
    def __init__(self):
        print("hello")
        super().__init__()
```

singletons

```
class MySingleton:
    instance = None

    def __new__(cls, *args, **kwargs):
        if not isinstance(cls.instance, cls):
            cls.instance = object.__new__(cls)
```

```
        return cls.instance
```

```
class Singleton(type):
    _instances = {}
    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            cls._instances[cls] = super (Singleton,cls).__call__(*args,
**kwargs)
        return cls._instances[cls]

class SerialNumber(metaclass=Singleton):
```

attributes / properties

- **attribute** - direct access to data member of object
- **property** - properties are methods accessed like attributes. It gives full control on its getter, setter and deleter access.

delattr(object, name) This is a relative of `setattr()`. The arguments are an object and a string. The string must be the name of one of the object's attributes. The function deletes the named attribute, provided the object allows it. For example, `delattr(x, 'foobar')` is equivalent to `del x.foobar`. name need not be a Python identifier (see setattr()).

Object-like attribute access for nested dictionary

- https://stackoverflow.com/questions/38034377/object-like-attribute-access-for-nested-dictionary
- https://bobbyhadz.com/blog/python-use-dot-to-access-dictionary
- https://github.com/frmdstryr/magicattr