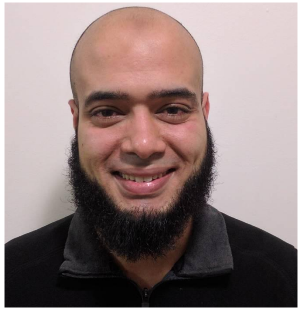
# Python Programming
# Inheritance with Super Function

**Mostafa S. Ibrahim**
*Teaching, Training and Coaching since more than a decade!*

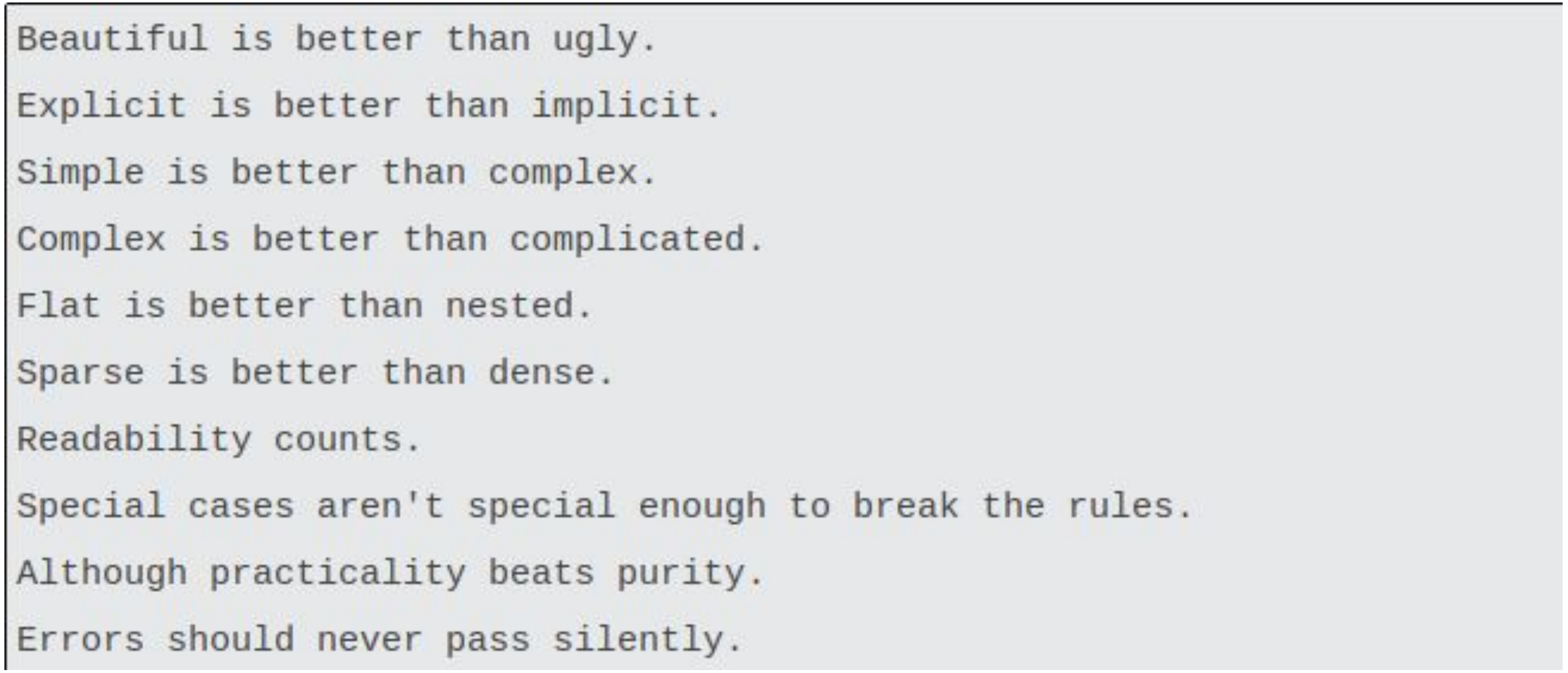
*Artificial Intelligence & Computer Vision Researcher*
*PhD* from Simon Fraser University - Canada
*Bachelor / Msc* from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



## Super Function

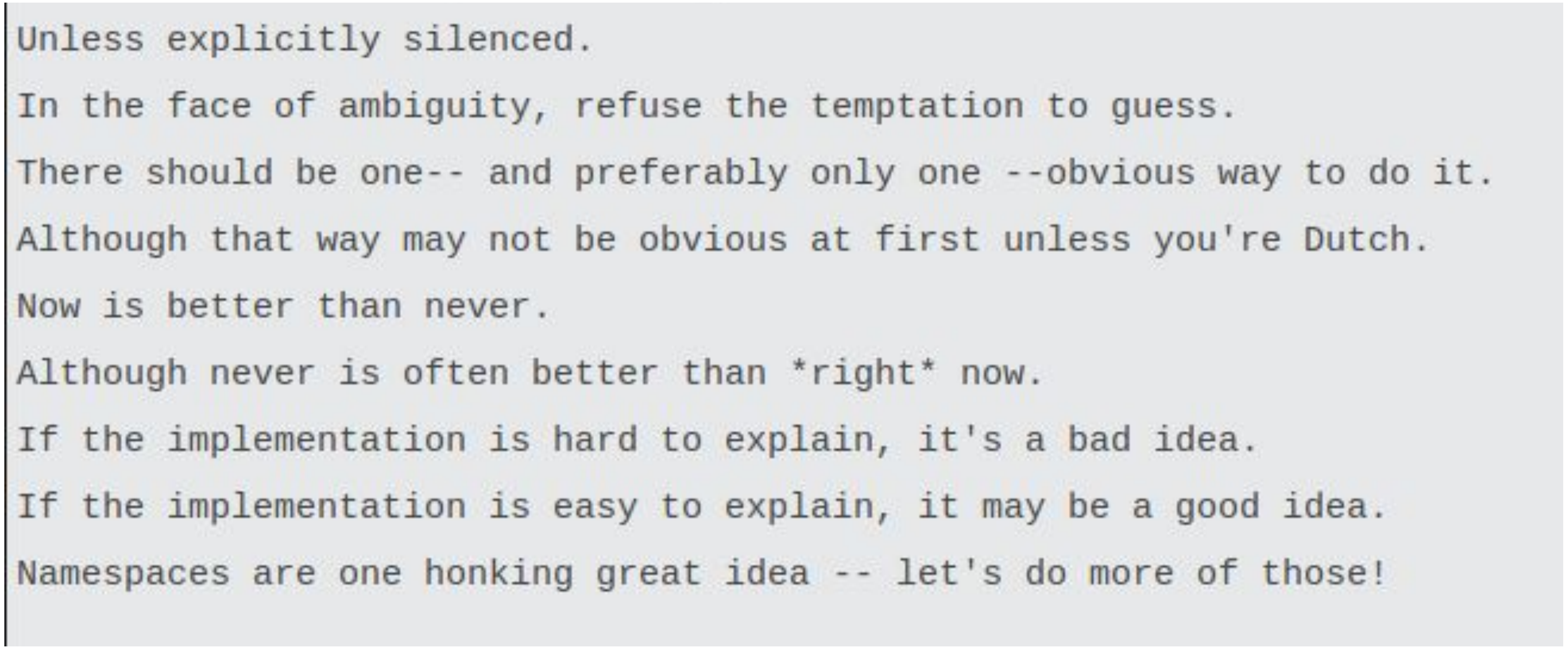
- An issue in previous calling for the parent is using the class name explicitly
    - Person.__init__(self)
    - With every class name change, you have to change it!
    - If you changed your inheritance hierarchy, you have to change it!
    - But Cons: Making code **less explicit** violates The **Zen of Python**
- Can we make things more dynamic? Yes super function
- super() returns an object of the superclass
    - Now we can just force call to its __init__
    - Later, I will explain more details
- *Note: Python 2 is a bit different*

## Side note: Zen of Python (*guiding principles*)



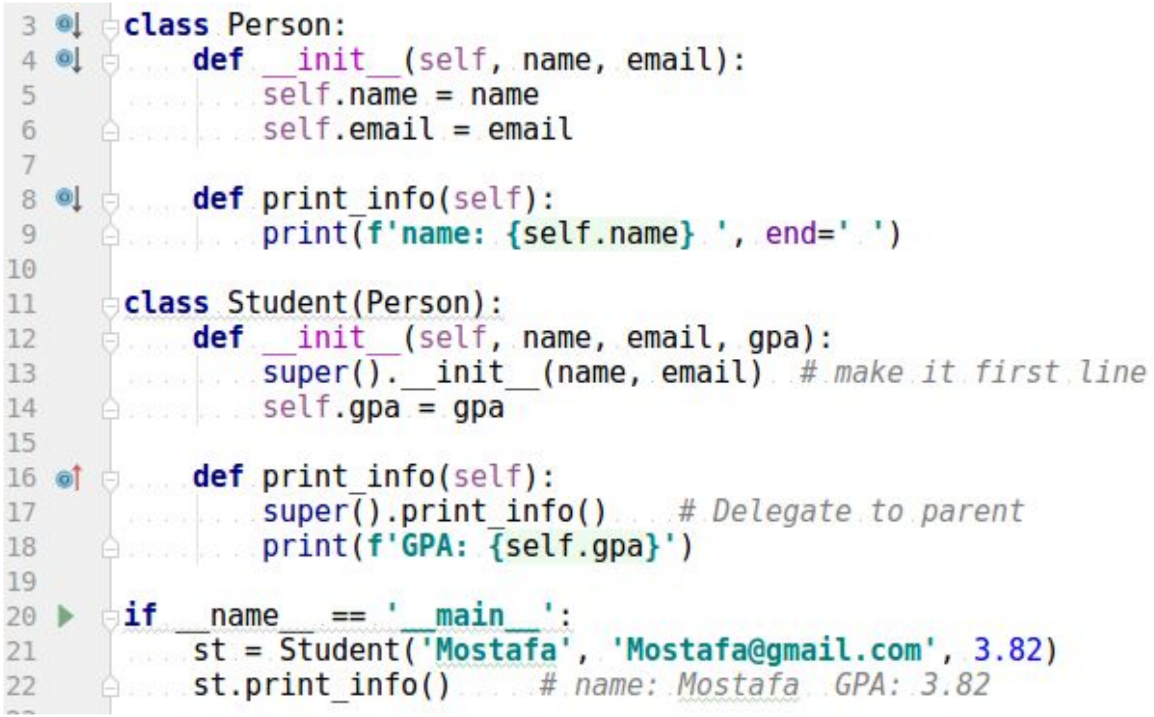
```
Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.
```

## Side note: Zen of Python (*guiding principles*)



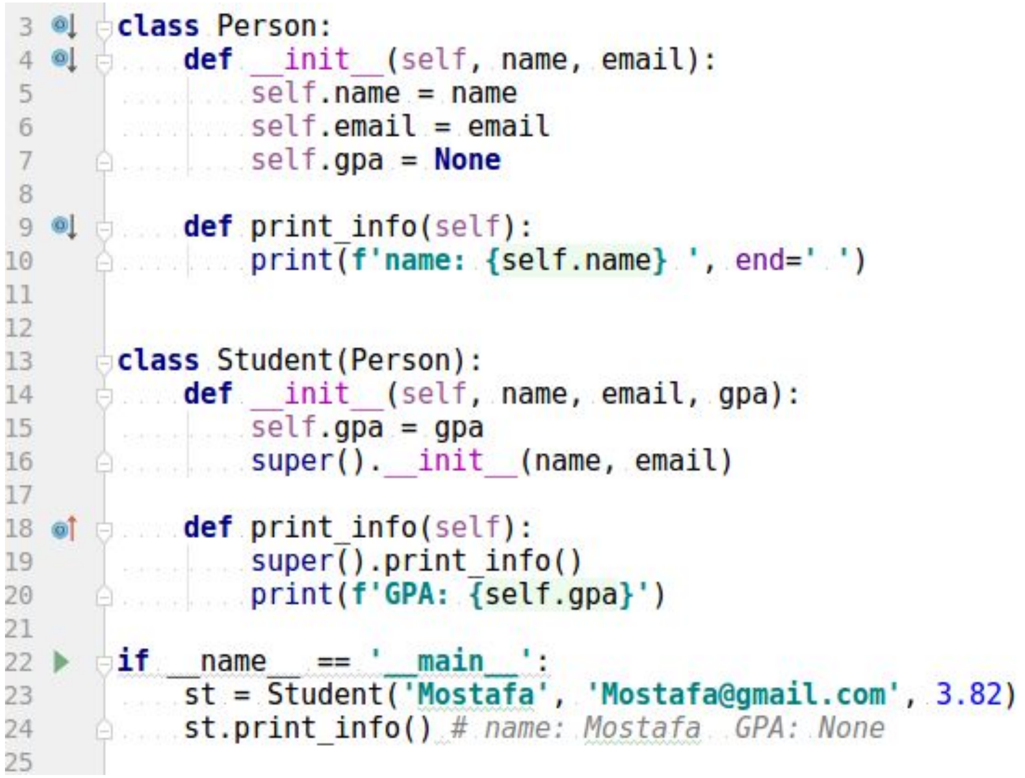
```
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

# super() function



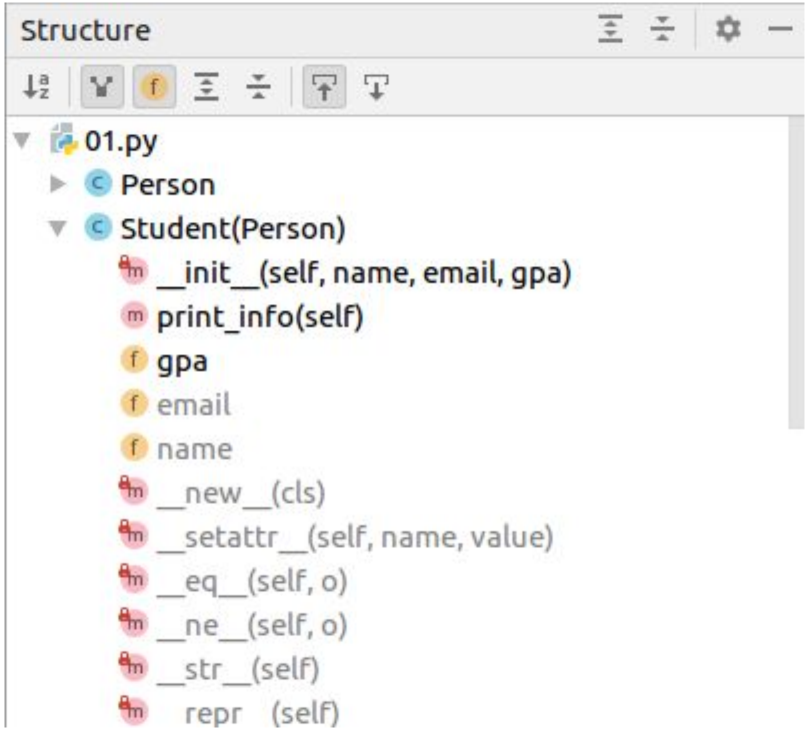
```python
class Person:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    def print_info(self):
        print(f'name: {self.name} ', end=' ')

class Student(Person):
    def __init__(self, name, email, gpa):
        super().__init__(name, email)  # make it first line
        self.gpa = gpa

    def print_info(self):
        super().print_info()    # Delegate to parent
        print(f'GPA: {self.gpa}')

if __name__ == '__main__':
    st = Student('Mostafa', 'Mostafa@gmail.com', 3.82)
    st.print_info()    # name: Mostafa  GPA: 3.82
```

# super() call order

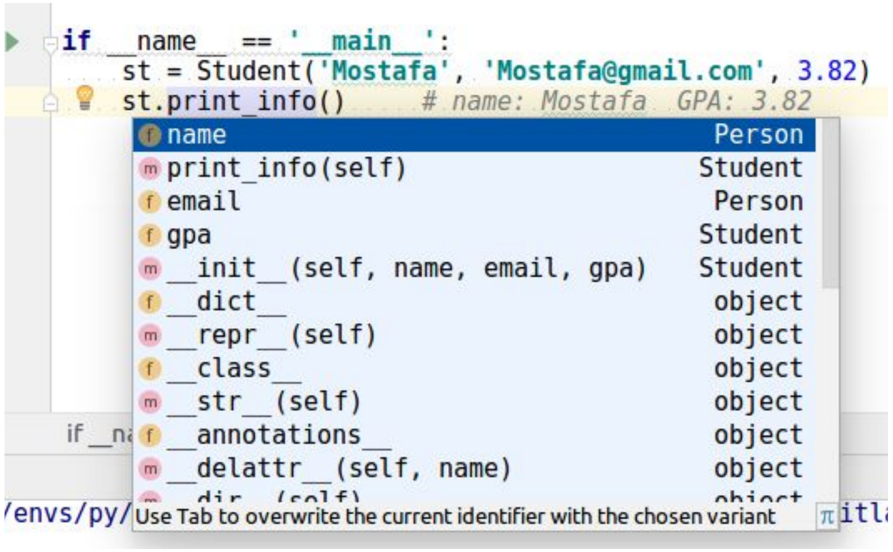


```python
class Person:
    def __init__(self, name, email):
        self.name = name
        self.email = email
        self.gpa = None

    def print_info(self):
        print(f'name: {self.name} ', end=' ')


class Student(Person):
    def __init__(self, name, email, gpa):
        self.gpa = gpa
        super().__init__(name, email)

    def print_info(self):
        super().print_info()
        print(f'GPA: {self.gpa}')

if __name__ == '__main__':
    st = Student('Mostafa', 'Mostafa@gmail.com', 3.82)
    st.print_info() # name: Mostafa  GPA: None

```

# Inheritance in PyCharm



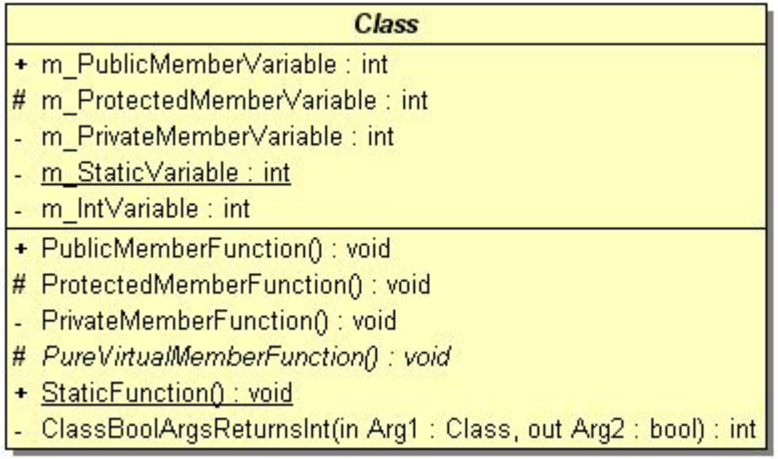
```
Structure
01.py
  Person
  Student(Person)
    __init__(self, name, email, gpa)
    print_info(self)
    gpa
    email
    name
    __new__(cls)
    __setattr__(self, name, value)
    __eq__(self, o)
    __ne__(self, o)
    __str__(self)
    __repr__(self)
```



```
if __name__ == '__main__':
    st = Student('Mostafa', 'Mostafa@gmail.com', 3.82)
    st.print_info()        # name: Mostafa  GPA: 3.82
```

```
name                                  Person
print_info(self)                      Student
email                                 Person
gpa                                   Student
__init__(self, name, email, gpa)      Student
__dict__                              object
__repr__(self)                        object
__class__                             object
__str__(self)                         object
__annotations__                       object
__delattr__(self, name)               object
__dir__(self)                         object
Use Tab to overwrite the current identifier with the chosen variant
```

## Protected in C++ UML: #   (Optional)


Class
+ m_PublicMemberVariable : int
# m_ProtectedMemberVariable : int
- m_PrivateMemberVariable : int
- m_StaticVariable : int
- m_IntVariable : int
+ PublicMemberFunction() : void
# ProtectedMemberFunction() : void
- PrivateMemberFunction() : void
# PureVirtualMemberFunction() : void
+ StaticFunction() : void
- ClassBoolArgsReturnsInt(in Arg1 : Class, out Arg2 : bool) : int

- + private
- - public
- # protected

Img Src

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."