# Week 11 Tutorial: Classes and Object-oriented Programming

## POP77001 Computer Programming for Social Scientists

Module website: tinyurl.com/POP77001

# Recap: Class definition

- Let us recap how a class can be defined in Python.
- Look at the mock class definition below.
- Check that you understand what every part of it is.
- Create an instance of the class.
- Make sure that you can access docstrings for both class and individual methods.

In [1]:

```
class MyClass(object):
    """Docstring describing the class."""
    # Use docstrings rather than comments to document
    # both the class and its individual methods
    # docstrings can be accessed interactively using help()
    # function. While comments would require going back to
    # source code.

    # Special Method for object instantiation - always first
    def __init__(self, var):
        """Create a new instance of class."""

        # Methods with two underscores ("__") serve special purposes.
        # E.g. __init__() is a constructor method that creates
        # an instance of the class.
        #
        # `self` refers to a specific instance of the class.
        # Then data fields can be described using `self.` prefix
        self.var = var

    def class_method(self, arg):
        """Docstring describing the method."""

        # Some method that does something.
        pass

    # Special Method for object string representation - always last
    def __str__(self):
        """Return a string representation of object."""
```

```
        # Define here how you want an instance of the classto be printe
        return str(self.var)
```

# Class definition example

- Let us come back to the definition of class `Survey` from the lecture.
- Check that you understand what each of the lines of code does.
- Try creating and manipulating the objects of this class.

In [2]:

```
class Survey(object):
    """Create a new Survey"""
    def __init__(self):
        """Initialize a new Survey with an empty questionnaire"""
        self.questionnaire = []
    def add_question(self, question):
        """Add question to the questionnaire"""
        self.questionnaire.append(question)
    def __add__(self, other):
        """Combine Surveys together"""
        return self.questionnaire + other.questionnaire
    def __len__(self):
        """Returns the length of Survey questionnaire"""
        return len(self.questionnaire)
```

# Exercise 1: Creating methods

- Notice that there only one method designed to be explicitly called by a user.
- It is the method for adding question to survey questionnaire.
- Create a corresponding question for removing questions.
- Try out a new class design:
    - First, create a Survey object;
    - Second, add 2 questions to the questionnaire;
    - Lastly, remove one of those questions from it.

# Exercise 2: Inheritance Hierarchy

- Create a subclass of `Survey` class called OnlineSurvey.
- Implement operator overloading for `<` (less than) operator.
- It will return True if the number of questions in one survey is smaller than the other.
- Create 2 Survey with different number of questions.
- Combine them in a list.
- Sort this list using `sorted()` function.

# Week 11: Assignment 4

- Data Wrangling in Python and Classes
- Due by 12:00 on Monday, 28th November