# Object-Oriented Programming

```
#-------------------------------------------------------------------------------
#   This is the file 'bankaccount.py'
#-------------------------------------------------------------------------------

class BankAccount( object ) :

    # A simple class for bank account objects

    #---------------------------------------------------------------------------

    def __init__( self, owner ) :

        # Create and return a 'BankAccount' object for 'owner',
        # initialising its '_balance' and '_transaction' fields to 0

        self._owner        = owner
        self._balance      = 0
        self._transactions = 0

    #---------------------------------------------------------------------------

    def Deposit( self, amount ) :

        # Increase the '_balance' field of the current object by 'amount'

        self._balance      += amount
        self._transactions += 1

    #---------------------------------------------------------------------------

    def Withdraw( self, amount ) :

        # Decrease the '_balance' field of the current object by 'amount',
        # but only if the result would be non-negative

        if self._balance < amount :
            print( "*** withdrawal ( %i ) exceeds balance ( %i )" % \
                   ( amount, self._balance ) )
        else :
            self._balance      -= amount
            self._transactions += 1

    #---------------------------------------------------------------------------

    def PrintStatement( self ) :

        # Output the owner, balance, and number of transactions
        # for the current object

        print( "Owner        =", self._owner        )
        print( "Balance      =", self._balance      )
        print( "Transactions =", self._transactions )

#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------

>>> from bankaccount import BankAccount

>>> a = BankAccount( "Anne" )

>>> a.PrintStatement( )
Owner        = Anne
Balance      = 0
Transactions = 0

>>> a.Deposit( 200 )

>>> a.Withdraw( 50 )

>>> a.Deposit( 100 )

>>> a.PrintStatement( )
Owner        = Anne
Balance      = 250
Transactions = 3

>>> t = BankAccount( "Tim" )

>>> t.Deposit( 100 )

>>> t.Withdraw( 150 )
*** withdrawal ( 150 ) exceeds balance ( 100 )

>>> t.Withdraw( 30 )

>>> t.PrintStatement( )
Owner        = Tim
Balance      = 70
Transactions = 2

#-------------------------------------------------------------------------------
```

Procedural Programming

- data items are considered passive
- functions are considered active and they process data:
  `len( lst )`
  `EqualFiles( f1, f2 )`
  `PrintDictionary( d )`

Object-Oriented Programming

- data items, known as OBJECTS, are considered active
- objects can STORE stuff and DO stuff
- a FIELD is a variable inside an object in which it can store stuff
- a METHOD is a function inside an object which enables it to do stuff
- fields are generally accessed and/or updated by calling methods
- to tell an object 'o' to call its method 'm' : o.m( arguments )
- the first parameter 'self' of every method is automatically set to the object which called the method
- a CLASS gives the blueprint for creating objects of a particular form
- to create an object from a class 'C' and store it in a variable 'o' :
  `o = C( arguments )`
  where 'arguments' are passed to a special method '__init__' within 'C'