A brute force approach

```
def dToB(n, numDigits):
    """requires: n is a natural number less than 2**numDigits
      returns a binary string of length numDigits representing
      the decimal number n."""
    assert type(n)==int and type(numDigits)==int
           and n >=0 and n < 2**numDigits
    bStr = ''
    while n > 0:
        bStr = str(n % 2) + bStr
        n = n//2
    while numDigits - len(bStr) > 0:
        bStr = '0' + bStr
    return bStr
```

A brute force approach

```
def genPset(Items):
    """Generate a list of lists representing
       the power set of Items"""
    numSubsets = 2**len(Items)
    templates = []
    for i in range(numSubsets):
        templates.append(dToB(i, len(Items)))
    pset = []
    for t in templates:
        elem = []
        for j in range(len(t)):
            if t[j] == '1':
                elem.append(Items[j])
        pset.append(elem)
    return pset
```

A brute force approach

```
def chooseBest(pset, constraint, getVal, getWeight):
    bestVal = 0.0
    bestSet = None
    for Items in pset:
        ItemsVal = 0.0
        ItemsWeight = 0.0
        for item in Items:
            ItemsVal += getVal(item)
            ItemsWeight += getWeight(item)
        if ItemsWeight <= constraint and ItemsVal > bestVal:
            bestVal = ItemsVal
            bestSet = Items
    return (bestSet, bestVal)
```

And the best is…

```
def testBest():
    Items = buildItems()
    pset = genPset(Items)
    taken, val = chooseBest(pset, 20, Item.getValue,
                                Item.getWeight)
    print ('Total value of items taken = ' + str(val))
    for item in taken:
        print '  ', item

>>> testBest()
Total value of items taken = 275.0
   <clock, 175.0, 10.0>
   <painting, 90.0, 9.0>
   <book, 10.0, 1.0>
```

Complexity?

- Note that this will in principle be slow since we generate the entire power set, then check each possible subset of elements to determine whether that subset meets the constraints
- This will find the best solution, since we check all options, but this is exponential in the number of items because we create the entier power set