```
def main():
    blurb()
    # Set up a random stardate
    stardate=float(random.randrange(1000,1500,1))
    # Enterprise starts with 3,000 units of energy, 15 torpedoes and 1,000
    # units of energy in its shields
    energy=3000
    torpedoes=15
    shields=0
    # No klingons around ... yet!
    klingons = 0
    # The galaxy is divided into 64 sectors. Each sectoris represented by one
    # element in the galaxy list. The galaxy list contains a three digit number
    # Hundreds = number of klingons in the sector
    # Tens = number of starbases
    # Units = number of stars
    galaxy=[]
    # Initialise the galaxy list
    for i in range (0,64):
        x=y=0
        z=random.randint(1,5)
        if random.randint(1,10)<8:
            x=random.randint(1,3)
        if random.randint(1,100)>88:
            y=1
        galaxy.append(x*100+y*10+z)
        # Keep a record of how many klingons are left to be destroyed
        klingons=klingons+x
    # Choose the starting sector and position for the Enterprise
    sector = random.randint(0,63)
    ent_position = random.randint(0,63)
    # Set up current sector and decode it
    # x = klingons; y = starbases; z = stars
    x,y,z=decode(galaxy[sector])
    # Set up the current sector map
    # Each sector has 64 positions in which a klingon, starbase, star
    # or the Enterprise may be located in
    current_sector=init(x,y,z,ent_position)
    # Perform a short range scan
    condition=srs(current_sector,ent_position)
    status(sector,stardate,condition,energy,torpedoes,shields,klingons)
    # Keep going until we have destroyed all the klingons or we run out of
    # energy or we quit
    while klingons > 0 and energy > 0:
        # Command
        # 1 = Helm
        # 2 = Long range scan
        # 3 = Phasers
        # 4 = Photon torpedoes
        # 5 = Shields
        # 6 = Resign
        command=int(raw_input('Command (1-6, 0 for help)? '))
        if command == 0:
            showhelp()
        elif command == 1:
            new_sector,energy,ent_position,stardate=helm(galaxy,sector,
            energy,current_sector,ent_position,stardate)
            # If we're still in the same sector as before, draw the Enterprise
            if sector == new_sector:
                current_sector[ent_position]=4
            else:
            # Else set up the Enterprise in the new sector
                sector = new_sector
                ent_position = random.randint(0,63)
                x,y,z=decode(galaxy[sector])
                current_sector=init(x,y,z,ent_position)
            # Perform a short range scan after every movement
            condition=srs(current_sector,ent_position)
```

```
        if condition == "Docked":
            # Reset energy, torpedoes and shields
            energy=3000
            torpedoes=15
            shields=0
        status(sector,stardate,condition,energy,torpedoes,shields,klingons)
    elif command == 2:
        lrs(galaxy,sector)
    elif command == 3:
        shields,energy,current_sector,ks=phasers(condition,shields,energy,
        current_sector,ent_position,x)
        if ks < x:
            # (x-ks) Klingons have been destroyed-update galaxy map
            galaxy[sector]=galaxy[sector]-(100*(x-ks))
            # update total klingons
            klingons=klingons-(x-ks)
            # update sector klingons
            x=ks
        # Do we still have shields left?
        if shields < 0:
            print "Enterprise dead in space"
            energy = 0
        else:
            condition=srs(current_sector,ent_position)
            status(sector,stardate,condition,energy,torpedoes,
            shields,klingons)
    elif command == 4:
        torpedoes,current_sector,ks=photontorpedoes(torpedoes,
        current_sector,ent_position,x)
        # A Klingon has been destroyed-update galaxy map
        if ks < x:
            galaxy[sector]=galaxy[sector]-100
            # update total klingons
            klingons=klingons-(x-ks)
            # update sector klingons
            x=ks
        condition=srs(current_sector,ent_position)
        status(sector,stardate,condition,energy,torpedoes,shields,klingons)
    elif command == 5:
        energy,shields=addshields(energy,shields)
        condition=srs(current_sector,ent_position)
        status(sector,stardate,condition,energy,torpedoes,shields,klingons)
    elif command == 6:
        # Set quit condition by making energy = 0
        energy = 0
    else:
        print "Command not recognised captain"
    # After a command has been issued and condition is Red, a klingon may
    # fire randomly on the enterprise!
    if condition == "Red" and command != 0:
        if random.randint(1,9)<6:
            print "Red alert - Klingons attacking!"
            time.sleep(0.5)
            damage=x*random.randint(1,50)
            shields=shields-damage
            print "Hit on shields: ",damage," energy units"
            # Do we still have shields left?
            if shields < 0:
                print "Enterprise dead in space"
                energy = 0
            else:
                condition=srs(current_sector,ent_position)
                status(sector,stardate,condition,energy,torpedoes,
                shields,klingons)
# If we get here we've won if no klingons are left, but lost otherwise
if klingons == 0:
    promotion()
else:
```

```
        lose()

def status(sector,stardate,condition,energy,torpedoes,shields,klingons):
    time.sleep(0.2)
    print "\nStardate:           ",stardate
    time.sleep(0.2)
    print "Condition:          ",condition
    time.sleep(0.2)
    print "Energy:             ",energy
    time.sleep(0.2)
    print "Photon torpedoes:   ",torpedoes
    time.sleep(0.2)
    print "Shields:            ",shields
    time.sleep(0.2)
    print "Klingons in galaxy: ",klingons, "\n"
    time.sleep(0.2)

def blurb():
    print "\nSpace ... the final frontier."
    time.sleep(1.5)
    print "These are the voyages of the starship Enterprise"
    print "Its five year mission ..."
    time.sleep(2.5)
    print "... to boldly go where no-one has gone before"
    time.sleep(1.5)
    print "You are Captain Kirk."
    print "Your mission is to destroy all of the Klingons in the galaxy."
    time.sleep(2.5)

def promotion():
    print "\nYou have successfully completed your mission!"
    print "The federation has been saved."
    print "You have been promoted to Admiral Kirk."

def lose():
    print "\nYou are relieved of duty."

def decode(sector):
    # Hundreds = klingons, tens = starbases, units = stars
    klingons=sector/100
    starbases=(sector-klingons*100)/10
    stars=sector-klingons*100-starbases*10
    return(klingons,starbases,stars)

def init(klingons,bases,stars,eposition):
    current_sector=[]
    for j in range (0,64):
        current_sector.append(0)
    # A value of 4 in the sector map indicates the Enterprise's position
    current_sector[eposition]=4
    # Add in the stars (value = 3)
    while stars > 0:
        position = random.randint(0,63)
        if current_sector[position]==0:
            current_sector[position]=3
            stars=stars-1
    # Add in the starbases (value = 2)
    while bases > 0:
        position=random.randint(0,63)
        if current_sector[position]==0:
            current_sector[position]=2
            bases=bases-1
    # Add in the klingons (value = -200)
    while klingons > 0:
        position=random.randint(0,63)
        if current_sector[position]==0:
            current_sector[position]=-200
            klingons=klingons-1
```

```
    return(current_sector)

def srs(current_sector,ent_pos):
    # Print out sector map
    # Key: >!< = Klingon
    #      <O> = Starbase
    #       *  = Star
    #      -O- = Enterprise
    klingons=False
    for i in range (0,64):
        if i%8 == 0:
            print
            time.sleep(0.2)
        if current_sector[i]<0:
            klingons=True
            print ">!<",
        elif current_sector[i]==0:
            print " . ",
        elif current_sector[i]==2:
            print "<O>",
        elif current_sector[i]==3:
            print " * ",
        else:
            print "-O-",
    print
    # Work out condition
    if klingons == False:
        condition="Green"
    else:
        condition="Red"
    # But docked status overrides Red/Green
    port=ent_pos-1
    starboard=ent_pos+1
    if port >= 0:
        if current_sector[port]==2:
            condition="Docked"
    if starboard <= 63:
        if current_sector[starboard]==2:
            condition="Docked"
    # Return condition status
    return(condition)

def helm(galaxy,sector,energy,cur_sec,epos,stardate):
    direction=int(raw_input('Course direction(1-9)? '))
    if direction >=1 and direction <=9 and direction !=5:
        # Work out the horizontal and vertical co-ordinates
        # of the Enterprise in the current sector
        # 0,0 is top left and 7,7 is bottom right
        horiz=epos/8
        vert=epos-horiz*8
        # And calculate the direction component of our course vector
        hinc,vinc=calcvector(direction)
        # How far do we need to move?
        warp=int(raw_input('Warp (1-63)? '))
        # If warp selected is in legal range move Enterprise
        if warp >= 1 and warp <= 63:
            # Check there is sufficient energy
            if warp <= energy:
                # Reduce energy by warp amount
                energy = energy - warp
                # Remove Enterprise from current position
                cur_sec[epos] = 0
                # Calculate the new stardate
                stardate = stardate + (0.1*warp)
                # For the moment, assume movement leaves us in original sector
                out = False
                # Move the Enterprise warp units in the specified direction
                i=1
```

```
                while i <= warp and out == False:
                    # Calculate the movement vector
                    vert = vert + vinc
                    horiz = horiz + hinc
                    # Are we in the original sector still?
                    if vert < 0 or vert > 7 or horiz < 0 or horiz > 7:
                        out=True
                        # Calculate new sector and join ends of the galaxy
                        sector=join(sector+8*(horiz/8)+(vert/8))
                    else:
                        # If we are in the original sector we can't go through
                        # solid objects! So reset course postion 1 click
                        # Inefficient - does this for warp steps even if we
                        # can't move.
                        if cur_sec[vert+8*horiz] != 0:
                            vert=vert-vinc
                            horiz=horiz-hinc
                        # Put the Enterprise in the new position
                        epos=vert+8*horiz
                    i=i+1
            else:
                print "Too little energy left. Only ",energy," units remain"
        else:
            print "The engines canna take it, captain!"
    else:
        print "That's not a direction the Enterprise can go in, captain!"
    return(sector,energy,epos,stardate)

def lrs(galaxy,sector):
    # Print out the klingons/starbase/stars values from the
    # neighbouring eight sectors (and this one)
    time.sleep(0.2)
    print
    for i in range (-8,9,8):
        for j in range (-1,2):
            # Join the ends of the galaxy together
            sec=join(sector+j+i)
            print "%03d" % galaxy[sec],
            time.sleep(0.2)
        print
    print

def phasers(condition,shields,energy,sector,epos,ksec):
    power=int(raw_input('Phaser energy? '))
    if power <= energy:
        # Reduce available energy by amount directed to phaser banks
        energy=energy-power
        # Divide phaser power by number of klingons in the sector if there are
        # any present! Space can do funny things to the mind ...
        if ksec > 0:
            power=power/ksec
            # Work out the vertical and horizotal displacement of Enterprise
            horiz=epos/8
            vert=epos-(8*horiz)
            # Check all of the 64 positions in the sector for Klingons
            for i in range (0,64):
                if sector[i]<0:
                    # We have a Klingon!
                    # Work out its horizontal and vertical displacement
                    horizk=i/8
                    vertk=i-(8*horizk)
                    # Work out distance from Klingon to Enterprise
                    z=horiz-horizk
                    y=vert-vertk
                    dist=1
                    while ((dist+1)*(dist+1))<(z*z+y*y):
                        dist=dist+1
                    # Klingon energy is negative, so add on the phaser power
```

```
                    # corrected for distance
                    sector[i]=sector[i]+int(power/dist)
                    if sector[i]>=0:
                        # Set this part of space to be empty
                        sector[i]=0
                        # Decrement sector klingons
                        ksec=ksec-1
                        print "Klingon destroyed!"
                        time.sleep(0.2)
                    else:
                        # We have a hit on Enterprise's shields if not docked
                        if condition != "Docked":
                            damage=int(power/dist)
                            shields=shields-damage
                            print "Hit on shields: ",damage," energy units"
                            time.sleep(0.2)
    else:
        print "Not enough energy, Captain!"
    return(shields,energy,sector,ksec)

def photontorpedoes(torpedoes,sector,epos,ksec):
    if torpedoes < 1:
        print "No photon torpedoes left, captain!"
    else:
        direction=int(raw_input('Fire in direction(1-4,6-9)? '))
        if direction >=1 and direction <=9 and direction !=5:
            time.sleep(0.2)
            # Work out the horizontal and vertical co-ordinates
            # of the Enterprise in the current sector
            # 0,0 is top left and 7,7 is bottom right
            horiz=epos/8
            vert=epos-horiz*8
            # And calculate the direction to fire the torpedo
            hinc,vinc=calcvector(direction)
            # A torpedo only works in the current sector and stops moving
            # when we hit something solid
            out = False
            while out == False:
                 # Calculate the movement vector
                vert = vert + vinc
                horiz = horiz + hinc
                # Is the torpedo still in the sector?
                if vert < 0 or vert > 7 or horiz < 0 or horiz > 7:
                    out=True
                    print "Torpedo missed"
                else:
                    # Have we hit an object?
                    if sector[vert+8*horiz] == 2:
                        # Oh dear - taking out a starbase ends the game
                        out=True
                        sector[vert+8*horiz] = 0
                        energy=0
                        print "Starbase destroyed"
                    elif sector[vert+8*horiz] == 3:
                        # Shooting a torpedo into a star has no effect
                        out=True
                        print "Torpedo missed"
                    elif sector[vert+8*horiz] < 0:
                        # Hit and destroyed a Klingon!
                        out=True
                        sector[vert+8*horiz] = 0
                        ksec = ksec - 1
                        print "Klingon destroyed!"
            # One fewer torpedo
            torpedoes = torpedoes-1
        else:
            print "Your command is not logical, Captain."
    return(torpedoes,sector,ksec)
```

```
def addshields(energy,shields):
    # Add energy to shields
    power=int(raw_input('Energy to shields? '))
    if ((power > 0) and (energy >= power)):
        energy = energy - power
        shields = shields + power
    return (energy,shields)

def calcvector(direction):
    # Convert numeric keypad directions to that of the original game
    # NK 7 = 7
    # NK 4 = 6
    # NK 1 = 5
    # NK 2 = 4
    # NK 3 = 3
    # NK 6 = 2
    # NK 9 = 1
    # NK 8 = 0
    # This could be rather more elegant if I didn't bother doing this!
    # However, I'm trying to stay true to the spirit of the original
    # BASIC listing ...
    if direction == 4:
        direction = 6
    elif direction == 1:
        direction = 5
    elif direction == 2:
        direction = 4
    elif direction == 6:
        direction = 2
    elif direction == 9:
        direction = 1
    elif direction == 8:
        direction = 0
    # Work out the direction increment vector
    # hinc = horizontal increment
    # vinc = vertical increment
    if direction < 2 or direction > 6:
        hinc = -1
    elif direction > 2 and direction < 6:
        hinc = 1
    else:
        hinc = 0
    if direction < 4 and direction > 0:
        vinc = 1
    elif direction > 4:
        vinc = -1
    else:
        vinc = 0
    return(hinc,vinc)

def join(sector):
    # Join the ends of the galaxy together
    if sector < 0:
        sector = sector + 64
    if sector > 63:
        sector = sector - 63
    return(sector)

def showhelp():
    # Print out the command help
    print "1 - Helm"
    print "2 - Long Range Scan"
    print "3 - Phasers"
    print "4 - Photon Torpedoes"
    print "5 - Shields"
    print "6 - Resign"
```

```
if __name__ == '__main__':
    import random
    import time
    main()
```