```
# File komega-v04a.py

# Author: G.Doeben-Henisch
# First date: September 4, 2020
# Last change: 12.September 2020

##################################
# Execution Environment of my local machine:
# (venv) gerd@gerd-ub2:~/env/komega/tst$ python3 komega-v01d.py
#

####################################
# GITHUB
#
# We use a github repository at:
# https://github.com/szmt/komega.git
#
# Im working from a unix-shell using the following github-commands:
# https://git-scm.com/docs/git

####################################
# BACKGROUND THEORY
#
# This code is a translation of a theory described in the blog
# https://www.uffmm.org
#
# Last document for the specification of this code:
#
'''
https://www.uffmm.org/2020/09/10/komega-requirements-no-4-version-4-basic-application-scenario/
'''

####################################
# ACTOR STORY
#
# In the specifications an actor story [AS] has been specified. This AS requires # some basic states which are dedicated for certain tasks to do:

'''
ACTOR STORY

S1: START
S2: EDIT P(roblem description)
S3: EDIT S (actual state)
S4: EDIT X (change rules)
S5: SIMULATION (Applying X to S)
S6: EVALUATION (After the simulation)
S7: STOP
'''

# MAIN IDEA
```

'''
According to the above mentioned actor story the user will be sitting in front of a system interface [SI] which works first only as a console.
In the beginning the user is placed in a start state S1 showing all options available.
The user can select one of these options and can from start state S1 reach all other states S2-S7.
'''

############################
IMPORTS

############################
SUPPORTING FUNCTION
#

No funtions yet

CLASSES
#
'''
For every state there exists one working class to do the job.
The special class 'Publish' in this code exists only because the interaction of the user with the system will happen with an interactive website which uses HTML and javascript. Here in this experimental environment a simple unix-console is used.
'''

import kcv4a as kc

#########################
Main Programm
#

###############
Start main loop
#
The loop will work as long as the value of the variable 'loop' is different to 'N'

loop='Y'
while loop!='N':

################################
STATE 1 : START
Show available options
Get feedback for selection
Confirm the selection
Distribute to different states

	kc.ast.menushow()

Ask back for selection number
	message='Enter a Number [1-7] for Menu Option \n'
	kc.pub.userinput(message)

```
# Evaluate the selection

        opt=kc.pub.opt
        kc.ast.badoption(opt)

#####################################################################
# Call to a class instance
#
#####################################################################
# Call to state Edit Problem P
################################
# STATE 2 : EDIT P
# Ask Questions related to P
# Collect all answers into one problem document
#

        if opt=='2':
                # Where You are
                kc.pub.show(kc.ap)

                #Interaction with Problem Class
                message='Enter your problem as it is now given in plain text\n'
                kc.pub.userinput(message)
                inp=kc.pub.opt
                kc.app.getproblem(inp)

                message='Enter your vision of a better state in the future in plain text\n'
                kc.pub.userinput(message)
                inp=kc.pub.opt
                kc.app.getvision(inp)

                message='Enter the name of the city you are in\n'
                kc.pub.userinput(message)
                inp=kc.pub.opt
                kc.app.getregion(inp)

                message='Time model [From, Until,Cycleunit [Y or M or D or H]]: '
                kc.pub.userinput(message)
                inp=kc.pub.opt
                kc.app.gettime(inp)

                message='Which kinds of persons are important? Write a list, comma separated
please : '
                kc.pub.userinput(message)
                inp=kc.pub.opt
                kc.app.getperson(inp)

                kc.app.problemTotal()

#
#####################################################################
```

```
# Call to state Edit Actual State S
################################
# STATE 3 : EDIT S
# Collect single expressions
# Collect all expressions into one  document describing S
#

        elif opt=='3':
                # Where You are
                kc.pub.show(kc.ass)

                Sloop='Y'
                while Sloop!='N':

                        # Interaction with actual state S class
                   message='Enter an expression for your state description  in plain text\n'
                   kc.pub.userinput(message)
                   inp=kc.pub.opt
                   kc.aas.getexpression(inp)
                   message="STOP Editing S = 'N', CONTINUE != 'N'  \n"
                   kc.pub.userinput(message)
                   inp=kc.pub.opt
                   Sloop=inp


#
#########################################################################
# Call to state Edit Change Rules X
################################
# STATE 4 : EDIT X
# Collect single expressions for change rules
# Collect all expressions into one  document describing X
#


        elif opt=='4':
                kc.pub.show(kc.ax)

#
#########################################################################
# Call to state  Simulation SIM
################################
# STATE 5 : Run the simulation
#


        elif opt=='5':
                kc.pub.show(kc.asim)

#
#########################################################################
# Call to state  Evaluation EV
```

```
#

        elif opt=='6':
                kc.pub.show(kc.aev)

#
######################################################################
# Call to state Stop STP
#

        elif opt=='7':
                kc.pub.show(kc.astp)

######################################################################
# End of Loop

# Clarify how to continue

        message="STOP MAIN LOOP = 'N', CONTINUE != 'N'  \n"
        kc.pub.userinput(message)
        inp=kc.pub.opt
        loop=inp

'''
TEST NEW PUBLISH CLASS

(venv) gerd@gerd-ub2:~/env/komega/tst$ python3 komega-v04a.py
1 is START
2 is EDIT P
3 is EDIT S
4 is EDIT X
5 is SIMULATION
6 is EVALUATION
7 is STOP
Enter a Number [1-7] for Menu Option
1
!!You have selected the state :
START
STOP MAIN LOOP = 'N', CONTINUE != 'N'
a
1 is START
2 is EDIT P
3 is EDIT S
4 is EDIT X
5 is SIMULATION
6 is EVALUATION
7 is STOP
Enter a Number [1-7] for Menu Option
2
!!You have selected the state :
EDIT P
Role : "Pedit"
```

Name : "ap"
Enter your problem as it is now given in plain text
Need a book
Feedback Problem Now :
Need a book
Enter your vision of a better state in the future in plain text
Get it from library
Feedback Problem Future :
Get it from library
Enter the name of the city you are in
Dieburg
Feedback Problem Region :
Dieburg
Time model [From, Until,Cycleunit [Y or M or D or H]]: 2020-Sept,2020Dec,M
Feedback Problem TimeModel :
['2020-Sept', '2020Dec', 'M']
Which kinds of persons are important? Write a list, comma separated please : Students, Librarians
Feedback Problem Persons :
['Students', ' Librarians']
Feedback Problem All :
['Need a book', 'Get it from library', 'Dieburg', '2020-Sept,2020Dec,M', 'Students, Librarians']
STOP MAIN LOOP = 'N', CONTINUE != 'N'
a
1 is START
2 is EDIT P
3 is EDIT S
4 is EDIT X
5 is SIMULATION
6 is EVALUATION
7 is STOP
Enter a Number [1-7] for Menu Option
3
!!You have selected the state :
EDIT S
Role : "Sedit"
Name : "ass"
Enter an expression for your state description in plain text
One
Feedback Your last expression :
One
Feedback Your document S so far :
['One']
STOP Editing S = 'N', CONTINUE != 'N'
a
Enter an expression for your state description in plain text
Two
Feedback Your last expression :
Two
Feedback Your document S so far :
['One', 'Two']
STOP Editing S = 'N', CONTINUE != 'N'
a

```
Enter an expression for your state description  in plain text
three
Feedback Your last expression :
three
Feedback Your document S so far :
['One', 'Two', 'three']
STOP Editing S = 'N', CONTINUE != 'N'
N
STOP MAIN LOOP = 'N', CONTINUE != 'N'
a
1 is START
2 is EDIT P
3 is EDIT S
4 is EDIT X
5 is SIMULATION
6 is EVALUATION
7 is STOP
Enter a Number [1-7] for Menu Option
7
!!You have selected the state :
STOP
Role : "STOP"
Name : "astp"
STOP MAIN LOOP = 'N', CONTINUE != 'N'
N
(venv) gerd@gerd-ub2:~/env/komega/tst$

'''

################################################################################


# File kcv3a.py

# Author: G.Doeben-Henisch
# First date: September 6, 2020
# Last date: September  12, 2020

#################################
# CLASS DEFINITIONS

class Start:
        def __init__(self):
                self.menulist = ['START','EDIT P','EDIT S', 'EDIT
X','SIMULATION','EVALUATION','STOP']

        def menushow(self):
                i=0  # Counter for menu-loop
                for state in self.menulist:
                        i=i+1
                        message="str(i)+' is '+state"
                        pub.useroutput(eval(message))
```

```
        def badoption(self,opt):
                if int(opt)<1 or int(opt)>7:
                        message='!!You have selected a bad option'
                        pub.useroutput(message)

                if int(opt)>0 and int(opt)<8:
                        message='!!You have selected the state :\n'+self.menulist[int(opt)-1]
                        pub.useroutput(message)

#####################################################################

class Actor:
        def __init__(self,role,name):
                self.role = role
                self.name = name

#####################################################################

class Publish():

        def show(self,other):
                print('Role : "%s"'%other.role)
                print('Name : "%s"'%other.name)

        def useroutput(self,message):
                print(message)

        def userinput(self,message):
                self.opt=input(message)




#####################################################################
# CLASS PROBLEM
'''
MAIN IDEA

A main window W1 with a menu showing all possible questions to be
answered.

(a) Describe the problem P: What is given and what is the intended future state?
(b) Describe the intended real part of the world (space).
(c) Describe the time model T : which time period, which cycles.
(d) Which kinds of actors are seen as being important for the problem and its  future?
(e) Some other assumptions.
'''


class Problem(Actor):
```

```
        def getproblem(self,inp):
                self.problemNow = inp
                message='Feedback Problem Now :\n'+self.problemNow
                pub.useroutput(message)

        def getvision(self,inp):
                self.problemFuture = inp
                message='Feedback Problem Future :\n'+self.problemFuture
                pub.useroutput(message)

        def getregion(self,inp):
                self.problemRegion = inp
                message='Feedback Problem Region :\n'+self.problemRegion
                pub.useroutput(message)

        def gettime(self,inp):
                self.problemTime = inp
                self.problemTM = self.problemTime.split(',')
                message='Feedback Problem TimeModel :\n'+str(self.problemTM)
                pub.useroutput(message)

        def getperson(self,inp):
                self.problemPerson = inp
                self.problemPRS = self.problemPerson.split(',')
                message='Feedback Problem Persons :\n'+str(self.problemPRS)
                pub.useroutput(message)

        def problemTotal(self):
                self.problemAll =[]
                self.problemAll.append(self.problemNow)
                self.problemAll.append(self.problemFuture)
                self.problemAll.append(self.problemRegion)
                self.problemAll.append(self.problemTime)
                self.problemAll.append(self.problemPerson)
                message='Feedback Problem All :\n'+str(self.problemAll)
                pub.useroutput(message)

########################################################################
# CLASS S(tate Description)
'''
IDEA:
This state should allow in the final version the editing of the texts S and X in parallel. Additionally
one should be able to call from within this state(s) the simulation mode to test whether the actual
texts are working.

FOR NOW:
In this first experimental version one has to work either with the stae S or with the state X
separatedly. Simulation would be a follow up state.

TASK:
Input all data which are necessary for the S-state (including sectioning
and extended texts with details)
```

ACTORS:
Human experts.

SYSTEM INTERFACE:
A main window W1 offering the editing of a text consisting of individual statements. Every statement can be edited separately and repeatedly.

ACTIONS:
Select either a given statement for editing or edit a new statement or stop.

IMPLEMENTATION:
Using the list-construct of python to collect expressions, because lists are ordered and mutable and allow many intersting operations.
'''

```
class AState(Actor):

        def __init__(self):
                self.stateAll = []

        def getexpression(self,inp):
                self.expression = inp
                self.stateAll.append(inp)
                message='Feedback Your last expression :\n'+str(self.expression)
                pub.useroutput(message)
                message='Feedback Your document S so far :\n'+str(self.stateAll)
                pub.useroutput(message)

################################
# CLASS INSTANCES

ast=Start()

ap=Actor("Pedit","ap")
app=Problem("PPedit","app")

ass=Actor('Sedit','ass')
aas=AState()
ax=Actor('Xedit','ax')
asim=Actor('SIM','asim')
aev=Actor('EVAL','aev')
astp=Actor('STOP','astp')

pub=Publish()
```