B1A_random.py

```python
# ==============================================================================
"""RANDOM : demo for some functions from the 'random' module"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0"
__date__    = "2022-11-12"
__usage__   = """
Simply press <ENTER> at each pause"""
# ==============================================================================
from ezCLI import *
from random import shuffle, choice, sample, randrange, random, gauss
# ------------------------------------------------------------------------------
# Shuffle a list of letters
# ------------------------------------------------------------------------------
letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
pause(f"letters : {letters}")
test = list(letters); shuffle(test);
pause(f"shuffle : {''.join(test)}")
# ------------------------------------------------------------------------------
# Take random elements (with push back) from a list of letters
# ------------------------------------------------------------------------------
pause(f"choice : {' '.join(choice(letters) for n in range(20))}")
# ------------------------------------------------------------------------------
# Take random elements (without push back) from a list of letters
# ------------------------------------------------------------------------------
pause(f"sample : {' '.join(sample(letters, 20))}")
# ------------------------------------------------------------------------------
# Take random values from an integer range (0,100,5)
# ------------------------------------------------------------------------------
pause(f"randrange : {[randrange(0, 100, 5) for n in range(15)]}")
# ------------------------------------------------------------------------------
# Histogram of the integer range selection for 100000 random values
# ------------------------------------------------------------------------------
histo = [0] * 10
for n in range(100000): val = randrange(0, 10); histo[val] += 1
pause(f"histogram of randrange(0,10) for 100000 values :\n{histo}")
# ------------------------------------------------------------------------------
# Take random values with uniform distribution on [0,1)
# ------------------------------------------------------------------------------
pause(f"random : {[random() for n in range(10)]}")
# ------------------------------------------------------------------------------
# Histogram of the uniform distribution for 100000 random values
# ------------------------------------------------------------------------------
histo = [0] * 10
for n in range(100000): val = int(10*random()); histo[val] += 1
pause(f"histogram of random() for 100000 values :\n{histo}")
# ------------------------------------------------------------------------------
# Take random values with a gaussian distribution on R
# ------------------------------------------------------------------------------
pause(f"gauss(0,1) : {[gauss(0,1) for n in range(10)]}")
# ------------------------------------------------------------------------------
# Histogram of the gaussian distribution for 1000000 random values
# ------------------------------------------------------------------------------
def clamp(val,a,b): return a if val < a else b if val > b else int(val)
histo = [0] * 11
for n in range(100000): val = clamp(gauss(0,1),-5,5); histo[val+5] += 1
pause(f"histogram of gauss(0,1) for 100000 values :\n{histo}")
# ------------------------------------------------------------------------------
# Take random values with a gaussian distribution of mean=5 and variance=0.01
# ------------------------------------------------------------------------------
pause(f"gauss(5,0.01) : {[gauss(5,.01) for n in range(10)]}")
# ------------------------------------------------------------------------------
# Histogram of the gaussian distribution for 1000000 random values
# ------------------------------------------------------------------------------
def clamp(val,a,b): return a if val < a else b if val > b else int(val)
histo = [0] * 10
for n in range(100000): val = clamp(gauss(5,0.01),0,10); histo[val] += 1
pause(f"histogram gauss(5,0.01) for 100000 values :\n{histo}")
# ==============================================================================
```

B2A_txtfile.py

```python
# ==============================================================================
"""TXTFILE : demo for 'read_txt/write_txt' functions from the 'ezCLI' toolbox"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0"
__date__    = "2022-11-12"
__usage__   = """
Simply press <ENTER> at each pause"""
# ==============================================================================
from ezCLI import *
# ------------------------------------------------------------------------------
# Sample use cases for 'read_txt'
# ------------------------------------------------------------------------------
# read the whole content from file
txt = read_txt('test-txt.txt')
pause(f">>> read all lines :\n{txt}")

# return line at index 4 from file
txt = read_txt('test-txt.txt', 4)
pause(f">>> read line at index 4 :\n{txt}")

# return lines in range(4,7) from file
txt = read_txt('test-txt.txt', 4, 7)
pause(f">>> read lines in range(4,7) :\n{txt}")

# return the first 3 lines from file
txt = read_txt('test-txt.txt', None, 3)
pause(f">>> read the first 3 lines :\n{txt}")

# return the last 5 lines from file
txt = read_txt('test-txt.txt', -5, None)
pause(f">>> read the last 5 lines :\n{txt}")

# ------------------------------------------------------------------------------
# Sample use cases for 'write_txt'
# ------------------------------------------------------------------------------
# replace the whole file and return the new file content
txt = write_txt('test.txt', 'ccc\nccc\nccc')
pause(f">>> replace whole content :\n{txt}")

# insert line at index 2 and return new file content
txt = write_txt('test.txt', 'ddd', 2)
pause(f">>> insert line at index 2 :\n{txt}")

# insert line at end of file and return new file content
txt = write_txt('test.txt', 'eee', -1)
pause(f">>> insert line at end of file :\n{txt}")

# replace first line and return new file content
txt = write_txt('test.txt', 'aaa\nbbb', None, 1)
pause(f">>> replace the first line :\n{txt}")

# replace the last 2 lines and return new file content
```

```
txt = write_txt('test.txt', 'eee\nfff\nggg', -2, None)
pause(f">>> replace the last 2 lines :\n{txt}")
# ==============================================================================
```

B3A_replace.py

```
# ==============================================================================
"""REPLACE : replace all occurrences of a string in a given text file"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0" # process a single file
__date__    = "2022-11-12"
__usage__   = """
User input : <oldstring> <newstring> <filename>
App output : modify 'filename' by replacing each 'oldstring' by 'newstring'
Note: use quotes if 'oldstring' or 'newstring' contains space characters"""
# ==============================================================================
from ezCLI import *
# ------------------------------------------------------------------------------
def replace(oldstring:str, newstring:str, name:str) -> str:
  """replace each occurrence of 'oldstr' by 'newstr' in file 'name'"""
  text = read_txt(name) # read whole content of file 'name'
  count = text.count(oldstring); #inspect()
  text = text.replace(oldstring, newstring); #inspect()
  write_txt(name, text) # write whole content of file 'name'
  return f"{name} : {count} strings have been replaced"
# ------------------------------------------------------------------------------
def parser(command:str) -> str:
  """extract arguments from 'command' before calling 'replace()'"""
  oldstring, newstring, name = parse(command); #inspect()
  return replace(oldstring, newstring, name)
# ==============================================================================
if __name__ == '__main__':
  userloop(parser, "Enter <oldstr> <newstr> <name>")
# ==============================================================================
```

B3B_replace.py

```
# ==============================================================================
"""REPLACE : replace all occurrences of a string in a set of text files"""
# ==============================================================================
__author__  = "Christophe Schlick"
__version__ = "2.0" # process an arbitrary set of files
__date__    = "2015-09-01"
__usage__   = """
User input : <oldstring> <newstring> <filename> [filename...]
App output : modify all 'filename' by replacing each 'oldstring' by 'newstring'
Note: use quotes if 'oldstring' or 'newstring' contains space characters"""
# ==============================================================================
from ezCLI import *
# ------------------------------------------------------------------------------
def replace(oldstring:str, newstring:str, name:str) -> str:
  """replace each occurrence of 'oldstr' by 'newstr' in file 'name'"""
  text = read_txt(name) # read whole content of file 'name'
  count = text.count(oldstring)
  text = text.replace(oldstring, newstring)
  write_txt(name, text)
  return "%s : %s strings have been replaced" % (name, count)
# ------------------------------------------------------------------------------
def parser(command:str) -> str:
  """extract arguments from 'command' before calling 'replace()'"""
  oldstring, newstring, *names = parse(command); inspect()
  return '\n'.join(replace(oldstring, newstring, name) for name in names)
# ==============================================================================
if __name__ == '__main__':
  userloop(parser, "Enter <oldstr> <newstr> <name> [name...]")
# ==============================================================================
```

B4A_csvfile.py

```
# ==============================================================================
"""CSVFILE : demo for 'read_csv/write_csv' functions from the 'ezCLI' toolbox"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0"
__date__    = "2022-11-12"
__usage__   = """
Simply press <ENTER> at each pause"""
# ==============================================================================
from ezCLI import *
# ------------------------------------------------------------------------------
# Sample use cases for 'read_csv'
# ------------------------------------------------------------------------------
# first read the file as a text file to show its raw content
txt = read_txt('test-csv.txt')
pause(f">>> read CSV file as a TXT file :\n{txt}")

# return the matrix stored in file and apply 'convert' to all cells
csv = read_csv('test-csv.txt')
pause(f">>> read CSV file as a matrix :\n{csv}")

# return the matrix stored in file but keep all cells as strings
csv = read_csv('test-csv.txt', raw=True)
pause(f">>> read CSV file as a matrix of strings :\n{csv}")

# ------------------------------------------------------------------------------
# Sample use cases for 'write_csv'
# ------------------------------------------------------------------------------
# create a sample non-rectangular 3D matrix containing 16 different cells
mat = [[list('abcde'), 'aa,bb,cc', ('dd','ee')], [1,(2,3),[4,5,6]]]
pause(f">>> create a sample 3D matrix :\n{mat}")

# replace the whole file and return the new file content
csv = write_csv('test.txt', mat)
pause(f">>> write matrix as a CSV file:\n{csv}")

# insert new block at head and return new file content
csv = write_csv('test.txt', '1,2,3\n4,5,6', 0)
pause(f">>> insert new block at head:\n{csv}")

# insert new block at tail and return new file content
csv = write_csv('test.txt', [7,8,9], -1)
pause(f">>> insert new block at tail:\n{csv}")

# replace the first two blocks and return new file content
csv = write_csv('test.txt', mat[1], None, 2)
pause(f">>> replace the first two blocks:\n{csv}")

# replace the last two blocks and return new file content
csv = write_csv('test.txt', mat[0], -2, None)
pause(f">>> replace the last two blocks:\n{csv}")
# ==============================================================================
```

B5A_inifile.py

```
# ==============================================================================
"""INIFILE : demo for 'read_ini/write_ini' functions from the 'ezCLI' toolbox"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0"
```

```
__date__    = "2022-11-12"
__usage__   = """
Simply press <ENTER> at each pause"""
# ==============================================================================
from ezCLI import *
# ------------------------------------------------------------------------------
# Sample use cases for 'read_ini'
# ------------------------------------------------------------------------------
# first read the file as a text file to show its raw content
txt = read_txt('test-ini.txt')
pause(f">>> read INI file as a TXT file :\n{txt}")

# return the data stored in file and apply 'convert' to all items
ini = read_ini('test-ini.txt')
pause(f">>> read INI file and convert all data type :\n{ini}")

# return the data stored in file but keep all items as strings
ini = read_ini('test-ini.txt', raw=True)
pause(f">>> read INI file as keep all data as strings :\n{ini}")

# ------------------------------------------------------------------------------
# Sample use cases for 'write_ini'
# ------------------------------------------------------------------------------
# sample data containing 5 properties spread in 3 sections
items = {'':{'a':1,'b':2},'AAA':{'aa':1.2,'bb':''},'BBB':{'cc':'#'}}
pause(f">>> sample structured data : \n{items}")

# replace the whole file and return the new file content
ini = write_ini('test.txt', items)
pause(f">>> write structured data as an INI file : \n{ini}")

# insert new section at tail and return new file content
ini = write_ini('test.txt', '\n[CCC]\nzz = 0', -1)
pause(f">>> insert new section at tail : \n{ini}")

# replace property stored at line 4 and return new file content
ini = write_ini('test.txt', {'aa':2.5}, 4, 5)
pause(f">>> replace property stored at line 4 : \n{ini}")
# ==============================================================================
```

B6_scores.py

```
# ==============================================================================
"""SCORES : demo for using INI files to store high scores for games"""
# ==============================================================================
__author__  = "Christophe Schlick modified by Philippe Blasi"
__version__ = "1.0"
__date__    = "2022-11-12"
__usage__   = """
Simply press <ENTER> at each pause"""
# ==============================================================================
from ezCLI import *

# read and parse the INI file 'scores.ini'
scores = read_ini('scores.ini')
pause(f">>> read and parse 'scores.ini':\n{scores}")

# each section in 'scores.ini' corresponds to a specific game configuration
# [rows,cols,goals]
for key in scores:
  print(f"found game configuration = '{key}'")

# get high score dictionary for specific game configuration [12,12,4]
config = '12,12,4'; hscores = scores[config]
pause(f">>> show current high score list for config '{config}':\n{hscores}")

# insert a new 'score = name' couple at the end of dictionary
hscores['587'] = 'tata'
pause(f">>> show new high score list for config '{config}':\n{hscores}")

# write 'scores+.ini' to store modified high scores
write_ini('scores+.ini', scores)

# WARNING : 'score = name' lines in INI files are always sorted alphabetically,
# and not numerically, so wrong sorting may be obtained. The usual solution is
# to pad with zeros: for instance, '123' < '32' but '032' < '123' as expected
#
# ==============================================================================
```