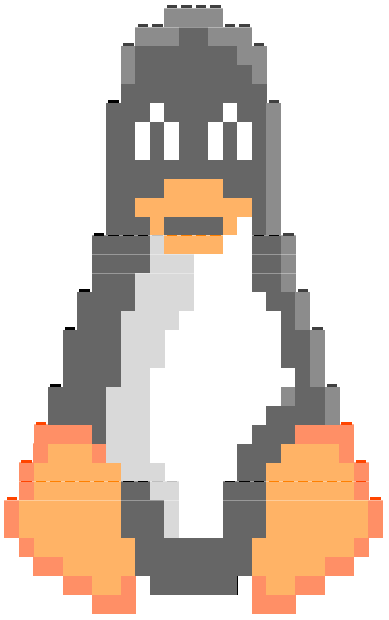
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΦΥΣ 140 Εισαγωγή στην Επιστημονική Χρήση Υπολογιστών
Χειμερινό Εξάμηνο 2023

Φώτης Πτωχός και Αλέξανδρος Αττίκης

Φροντιστήριο 5

10 Οκτωβρίου 2023
15:00 - 17:00



Φροντιστήριο 5

Παράδειγμα 1 Παραδείγματα χρήσης των συναρτήσεων της βιβλιοθήκης string όπως replace, find, split, join:

tutorial5/ex1.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex1.py
   python3 ex1.py
   script -q ex1.log python3 -i ex1.py


DESCRIPTION:
More examples of usage of string functions (find, replace, split, count, join)
'''

# Define my string variable that I will use in this example
msg = "And you run, and you run to catch up with the sun but it's sinking"


# Usage of replace(old, new, count) -> void
print("=== Replace the first \"run\" only with \"\RUN\":\n\t", msg.replace("run", "RUN", 1))
print("\n=== Replace the every instance of  \"o\" with \"\O\":\n\t", msg.replace("o", "O", -1))
print("\n=== Original string variable remains unchanged since functions create new variable:\n\t", msg)


# Usage of find(substring, [start], [end]) -> int
first  = msg.find("run", 0, -1)
second = msg.find("run", first+1, -1)
print("\n=== The result of msg[%d:%d] is:\n\t%s" % (first, second, msg[ first:second+len("run")] ) )
print("\n=== The result of msg[%d:%d] without the use of variables is exactly the same:\n\t%s" % (msg.find("run", 0, -1), msg.find("run", msg.find("run", 0, -1)+1, -1), msg[ first:second+len("run")] ) )


# Usage of split(separation, [maxSplit] ) -> list
print("\n=== The result of calling msg.split() on a string is a new list according to split criteria:\n\t%s" % (msg.split()) )
print("\n=== What happens if we change the splitting criteria to split(\"run\", -1):\n\t%s" % (msg.split("run", -1)) )
print("\n=== What happens if we change the splitting criteria to split(\"run\", +1):\n\t%s" % (msg.split("run", +1)) )
print("\n=== What happens if we change the splitting criteria to split(\"and\", -1):\n\t%s" % (msg.split("and", -1)) )


# Usage of count(substring, [start], [end]) -> int
```

```python
print("\n=== Count instances of letter \"u\" in string before comma:\n\t%s" % (
    msg.lower().count("u", 0, msg.find(","))) )
print("\n=== Count instances of letter \"u\" (incl. \"U\") in string after
    comma:\n\t%s" % (msg.lower().count("u", msg.find(","), -1)) )


# Finally, introduce the function join(iterable) -> string
print("\n=== Use join() to get a string that consists of all the elements in an
     iterable concatenated together with the specified separator in between each
     pair of elements:")

print("\t", '.'.join(['a', 'b', 'c']) )
print("\t", "-".join( [str(x) for x in range(10, -1, -1)]) )
print("\t", " | ".join( [str(x) for x in range(10, -1, -1)]) )

print("\n=== Result of calling \" \".join(msg.split()):\n\t", " ".join(msg.
    split()))

print("\n=== However, a string is also iterable so one can just do \"\".join(
    msg):\n\t", "".join(msg.upper()))
print("\n=== Or simply \"\".join(msg):\n\t", "".join(msg))

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== Replace the first "run" only with "\RUN":
   And you RUN, and you run to catch up with the sun but it's sinking

=== Replace the every instance of  "o" with "\O":
   And yOu run, and yOu run tO catch up with the sun but it's sinking

=== Original string variable remains unchanged since functions create new
    variable:
   And you run, and you run to catch up with the sun but it's sinking

=== The result of msg[8:21] is:
  run, and you run

=== The result of msg[8:21] without the use of variables is exactly the same
    :
  run, and you run

=== The result of calling msg.split() on a string is a new list according to
     split criteria:
  ['And', 'you', 'run,', 'and', 'you', 'run', 'to', 'catch', 'up', 'with', '
    the', 'sun', 'but', "it's", 'sinking']

=== What happens if we change the splitting criteria to split("run", -1):
  ['And you ', ', and you ', " to catch up with the sun but it's sinking"]

=== What happens if we change the splitting criteria to split("run", +1):
  ['And you ', ", and you run to catch up with the sun but it's sinking"]

=== What happens if we change the splitting criteria to split("and", -1):
  ['And you run, ', " you run to catch up with the sun but it's sinking"]

=== Count instances of letter "u" in string before comma:
  2

=== Count instances of letter "u" (incl. "U") in string after comma:
  5

=== Use join() to get a string that consists of all the elements in an
    iterable concatenated together with the specified separator in between
    each pair of elements:
   a.b.c
   10-9-8-7-6-5-4-3-2-1-0
   10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0

=== Result of calling " ".join(msg.split()):
   And you run, and you run to catch up with the sun but it's sinking

=== However, a string is also iterable so one can just do "".join(msg):
   AND YOU RUN, AND YOU RUN TO CATCH UP WITH THE SUN BUT IT'S SINKING

=== Or simply "".join(msg):
   And you run, and you run to catch up with the sun but it's sinking

=== Quit!
```

Τι έγινε με την τελευταία εντολή print του προγράμματος μας;

Παράδειγμα 2 Παράδειγμα για τη διαχείριση αντικειμένων τύπου string με τη μέθοδο split(), που χρησιμοποιείται για να χωριστεί μια συμβολοσειρά σε μια λίστα από υπο-συμβολοσειρές, βάσει ενός καθορισμένου διαχωριστικού χαρακτήρα:

tutorial5/ex2.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex2.py
   python3 ex2.py
   script -q ex2.log python3 -i ex2.py


DESCRIPTION:
The split() method for strings that takes the generic form:
str.split( sep=" ", maxsplit=-1 )


LINKS:
https://www.w3schools.com/python/ref_string_split.asp
'''
# Define a string variables
phrase = "It's... Monty Python's Flying Circus"

# Let's try some split settings
print("=== default split (1)")
mySplit = phrase.split()
print(mySplit)
print(type(mySplit))
print(len(mySplit))

print("\n=== one-space split (1)")
mySplit = phrase.split(" ", 1)
print(mySplit)
print(type(mySplit))
print(len(mySplit))


print("\n=== two-spaces split (1)")
mySplit = phrase.split(" ", 2)
print(mySplit)
print(type(mySplit))
print(len(mySplit))


# Let's now change the splitting variable
print("\n=== default y split (2)")
mySplit = phrase.split("y", -1)
print(mySplit)
print(type(mySplit))
print(len(mySplit))

print("\n=== one-space split (2)")
```

```python
mySplit = phrase.split("y", +1)
print(mySplit)
print(type(mySplit))
print(len(mySplit))


print("\n=== two-spaces split (2)")
mySplit = phrase.split("y", +2)
print(mySplit)
print(type(mySplit))
print(len(mySplit))


# Let's finish off with another two examples
print("\n=== default split (3)")
mySplit = phrase.split("...", -1)
print(mySplit)

print("\n=== default split (4)")
mySplit = phrase.split(".", -1)
print(mySplit)

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== default split (1)
["It's....", 'Monty', "Python's", 'Flying', 'Circus']
<class 'list'>
5

=== one-space split (1)
["It's....", "Monty Python's Flying Circus"]
<class 'list'>
2

=== two-spaces split (1)
["It's....", 'Monty', "Python's Flying Circus"]
<class 'list'>
3

=== default y split (2)
["It's.... Mont", ' P', "thon's Fl", 'ing Circus']
<class 'list'>
4

=== one-space split (2)
["It's.... Mont", " Python's Flying Circus"]
<class 'list'>
```

```
2

=== two-spaces split (2)
["It’s.... Mont", ’ P’, "thon’s Flying Circus"]
<class ’list’>
3

=== default split (3)
["It’s", ". Monty Python’s Flying Circus"]

=== default split (4)
["It’s", ’’, ’’, ’’, " Monty Python’s Flying Circus"]

=== Quit!
```

Παράδειγμα 3 Απλό παράδειγμα ανάγνωσης και αποθήκευσης δεδομένων με τη χρήση αρχείων:

tutorial5/ex3.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex3.py
   python3 ex3.py
   script -q ex3.log python3 -i ex3.py


DESCRIPTION:
Introduction to open a file, writing, and closing. File modes:

"r" ..: Read - Opens a file for reading, error if file does not exist
"a" ..: Append - Opens file for appending, creates file if it does not exist
"w" ..: Write - Opens file for writing, creates file if it does not exist
"x" ..: Create - Creates file, returns error if file exists


LINKS:
https://www.w3schools.com/python/python_file_handling.asp
https://www.askpython.com/python/built-in-methods/open-files-in-python
'''

# Variable definition
filePath = "ex3.txt" # NOTE: full path needed
fileMode = "w"
print("=== Opening file '%s' in '%s' mode" % (filePath, fileMode), end="!\n")
f = open(filePath, fileMode)
f.write("Hello world!")
f.close()


# Lets read the file
fileMode = "r"
print("\n=== Opening file '%s' in '%s' mode" % (filePath, fileMode), end=".\n")
f = open(filePath, fileMode)

contents = f.read()
print("\n=== The file contents are \"%s\". They are of type %s" % (contents,
   type(contents)), end="\n")


# Demonstrate that you cannot write to a file that is open in "read" mode
try:
    f.write("This will fail to write because the file is in '%s' mode!" % (
   fileMode) )
except:
    print("\n=== Closing file '%s'" % (filePath), sep=' ', end='\n') #, file=
   sys.stdout, flush=False)
    f.close()

```

```python

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== Opening file 'ex3.txt' in 'w' mode!

=== Opening file 'ex3.txt' in 'r' mode.

=== The file contents are "Hello world!". They are of type <class 'str'>

=== Closing file 'ex3.txt'

=== Quit!
```

Παράδειγμα 4 Άλλο παράδειγμα για ανάγνωση και αποθήκευση δεδομένων με τη χρήση αρχείων:

tutorial5/ex4.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex4.py
   python3 ex4.py
   script -q ex4.log python3 -i ex4.py


DESCRIPTION:
Further information to file handling (open, writing, and closing). Reminder on
    file modes:
"r" ..: Read - Opens a file for reading, error if file does not exist
"a" ..: Append - Opens file for appending, creates file if it does not exist
"w" ..: Write - Opens file for writing, creates file if it does not exist
"x" ..: Create - Creates file, returns error if file exists


LINKS:
https://www.w3schools.com/python/python_file_open.asp
https://www.tutorialspoint.com/python/file_seek.htm
https://www.w3schools.com/python/ref_file_seek.asp
'''

# Variable definition
filePath = "ex4.txt"
fileMode = "w"
print("=== Open file '%s' in '%s' mode" % (filePath, fileMode) )
f = open(filePath, fileMode)
for i in range(0, 11, +1):
    msg = "L%d\n" % (i)
    f.write(msg)
f.close()


# Lets read the file
fileMode = "r"
f = open(filePath, fileMode)
print("\n=== Opened file '%s' in '%s' mode:" % (f.name, f.mode), end="\n")
for line in f:
    print(line, end="")


# Read again the file with alterative ways
print("\n=== Use seek() to move back to beginning of a file (must go back to
    line 0 of file before reading again!)")
f.seek(0)
print("\n=== Retrieve the file contents as a LIST with readlines():", end="\n")
fList = f.readlines() # list
for row in fList:
    print(row, end="")
# Keep file open


print("\n=== Must use seek() again to set the position of the read/write
    pointer within the file to begining")
f.seek(0)
print("\n=== Retrieve the file contents as a STRING with f.read():")
fString = f.read()
print(fString)
print(type(fString))


print("\n=== Retrieve the file contents as a LIST with f.read().split()", end="
    :\n")
f.seek(0) # if not used then get empty list!
fList = f.read().split()
print(fList)
print(type(fList))


print("\n=== Loop over all lines in the file", end=":\n")
f.seek(0)
for line in f:
    print(line, end="")
f.close()


print("\n=== Alternatively, can loop over the contents list as follows", end="
    :\n")
for iRow in range(len(fList)):
    print("fList[%d] = %s" % (iRow, fList[iRow]), end="\n")

print("\n=== Finally, we can do the same thing with the "".join( list) method
    as well:", "\n".join(fList), end="\n\n=== Quit!\n")


quit()
```

Αποτέλεσμα:

```
=== Open file 'ex4.txt' in 'w' mode

=== Opened file 'ex4.txt' in 'r' mode:
L0
L1
L2
L3
L4
L5
```

```
L6
L7
L8
L9
L10

=== Use seek() to move back to beginning of a file (must go back to line 0
   of file before reading again!)

=== Retrieve the file contents as a LIST with readlines():
L0
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10

=== Must use seek() again to set the position of the read/write pointer
   within the file to begining

=== Retrieve the file contents as a STRING with f.read():
L0
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10

<class ’str’>

=== Retrieve the file contents as a LIST with f.read().split():
[’L0’, ’L1’, ’L2’, ’L3’, ’L4’, ’L5’, ’L6’, ’L7’, ’L8’, ’L9’, ’L10’]
<class ’list’>

=== Loop over all lines in the file:
L0
L1
L2
L3
L4
L5
```

```
L6
L7
L8
L9
L10

=== Alternatively, can loop over the contents list as follows:
fList[0] = L0
fList[1] = L1
fList[2] = L2
fList[3] = L3
fList[4] = L4
fList[5] = L5
fList[6] = L6
fList[7] = L7
fList[8] = L8
fList[9] = L9
fList[10] = L10

=== Finally, we can do the same thing with the .join( list) method as well:
   L0
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10

=== Quit!
```

Παράδειγμα 5 Τα list comprehensions είναι μία πιο συνοπτική σύνταξη για την δημιουργία λίστας και ένας πολύ βολικός τρόπος για να μετασχηματίζεις λίστες δημιουργώντας νέες λίστες. Η βασική τους δομή φαίνεται στο παρακάτω παράδειγμα:

tutorial5/ex5.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex5.py
   python3 ex5.py
   script -q ex5.log python3 -i ex5.py


DESCRIPTION:
Simple examples of list comprehension

'''
# Create an empty list. Fill with integers from 10 to 0
myList = []
for i in range(100, -1, -1):
    myList.append(i)
print("\n=== Printing list created in for-loop:\n\t", myList, end="\n")


# Easier with list comprehension
myList = [ i for i in range(100, -1, -1)]
print("\n=== List created with list comprehension:\n\t", myList, end="\n")


# Create a new list with even (odd) numbers; any integer that is (is not)
    exactly divisible by 2
evenList= [x for x in myList if x % 2 == 0]
oddList = [y for y in myList if y % 2 !=0 ]
print("\n=== List of odd numbers using list comprehension:\n\t", oddList, end="
    \n")
print("\n=== List of even numbers using list comprehension:\n\t", evenList, end
    ="\n")

newList= [z for z in myList if z % 2 == 0 if z!=0]
print("\n=== List of even numbers but without zero:\n\t", newList, end="\n")


print("\n=== More Conditional list comprehension:")
newList = [w if w%10==0 and w>50 else -w  for w in myList]
print("\t %8s %10s" % ("newList", newList) )


print("\n=== More conditional list comprehension:\n\t", end="")
myList = [str(x)+' is EVEN' if x%2==0 else str(x)+' is ODD' for x in range(1,
    11, 1) ]
#print("\t %8s %10s" % ("myList",  myList) )
print("\n\t".join(myList) )

```

```python

# Looping simultaneously over 2 lists
print("\n=== Using zip() to loop over 2 lists simultaneously:")
for odd,even in zip(oddList, evenList):
    print("\todd={0:3d}, even={1:3d}".format(odd, even) )


# Summing over a list with sum()
oneList = [1 for i in range(0, 10, 1)]
print("\n=== Summing over a list with sum():\n\toneList=%s, sum(oneList)=%d" %
    (oneList, sum(oneList)) )

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== Printing list created in for-loop:
   [100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83,
    82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65,
    64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47,
    46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29,
    28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11,
    10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

=== List created with list comprehension:
   [100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83,
    82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65,
    64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47,
    46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29,
    28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11,
    10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

=== List of odd numbers using list comprehension:
   [99, 97, 95, 93, 91, 89, 87, 85, 83, 81, 79, 77, 75, 73, 71, 69, 67, 65,
    63, 61, 59, 57, 55, 53, 51, 49, 47, 45, 43, 41, 39, 37, 35, 33, 31, 29,
    27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 1]

=== List of even numbers using list comprehension:
   [100, 98, 96, 94, 92, 90, 88, 86, 84, 82, 80, 78, 76, 74, 72, 70, 68, 66,
    64, 62, 60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30,
    28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2, 0]

=== List of even numbers but without zero:
   [100, 98, 96, 94, 92, 90, 88, 86, 84, 82, 80, 78, 76, 74, 72, 70, 68, 66,
    64, 62, 60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30,
    28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

=== More Conditional list comprehension:
    newList [100, -99, -98, -97, -96, -95, -94, -93, -92, -91, 90, -89, -88,
     -87, -86, -85, -84, -83, -82, -81, 80, -79, -78, -77, -76, -75, -74,
    -73, -72, -71, 70, -69, -68, -67, -66, -65, -64, -63, -62, -61, 60, -59,
    -58, -57, -56, -55, -54, -53, -52, -51, -50, -49, -48, -47, -46, -45,
    -44, -43, -42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31,
    -30, -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17,
    -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0]

=== More conditional list comprehension:
  1 is ODD
  2 is EVEN
  3 is ODD
  4 is EVEN
  5 is ODD
  6 is EVEN
  7 is ODD
  8 is EVEN
  9 is ODD
  10 is EVEN

=== Using zip() to loop over 2 lists simultaneously:
  odd= 99, even=100
  odd= 97, even= 98
  odd= 95, even= 96
  odd= 93, even= 94
  odd= 91, even= 92
  odd= 89, even= 90
  odd= 87, even= 88
  odd= 85, even= 86
  odd= 83, even= 84
  odd= 81, even= 82
  odd= 79, even= 80
  odd= 77, even= 78
  odd= 75, even= 76
  odd= 73, even= 74
  odd= 71, even= 72
  odd= 69, even= 70
  odd= 67, even= 68
  odd= 65, even= 66
  odd= 63, even= 64
  odd= 61, even= 62
  odd= 59, even= 60
  odd= 57, even= 58
  odd= 55, even= 56
  odd= 53, even= 54
  odd= 51, even= 52
  odd= 49, even= 50
  odd= 47, even= 48
  odd= 45, even= 46
  odd= 43, even= 44
  odd= 41, even= 42
  odd= 39, even= 40
  odd= 37, even= 38
  odd= 35, even= 36
  odd= 33, even= 34
  odd= 31, even= 32
  odd= 29, even= 30
  odd= 27, even= 28
  odd= 25, even= 26
  odd= 23, even= 24
  odd= 21, even= 22
  odd= 19, even= 20
  odd= 17, even= 18
  odd= 15, even= 16
  odd= 13, even= 14
  odd= 11, even= 12
  odd=  9, even= 10
  odd=  7, even=  8
  odd=  5, even=  6
  odd=  3, even=  4
  odd=  1, even=  2

=== Summing over a list with sum():
  oneList=[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], sum(oneList)=10

=== Quit!
```

Παράδειγμα 6 Μια συνάρτηση μπορεί να εφαρμοστεί σε όλα τα στοιχεία ενός επαναληπτέου (tuple, array, list) χρησιμοποιώντας την μέθοδο map:

tutorial5/ex6.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex6.py
   python3 ex6.py
   script -q ex6.log python3 -i ex6.py


DESCRIPTION:
Simple example of the usage of the map() method, which is used to apply a
specified function to each item in an iterable (e.g., a list, tuple, or string)
and return another iterable (a map object) containing the results.
'''
def xSquared(x):
    return x ** 2

def twoSum(x, y):
    return x + y

def threeSum(x, y, z):
    return x + y + z


# Define a list of numbers
numList = [2*i for i in range(5)]
print("=== My original list is:", numList, sep="\n\t", end="\n")


# 1) Define a map that : Square each number using a lambda function
print("\n=== Square each number by mapping xSquared() to numList iterable:\n\t"
    , list( map(xSquared, numList)) )


# 2) Define a map to  convert numList to strings
strList = list( map(str, numList) )
print("\n=== Convert each number to string by mapping str to numList:", strList
    , sep="\n\t", end="\n")


# 3) Calculate the lengths of words in a list
lenList = list( map(len, ["one", "two", "three"]) )
print("\n=== Convert each number to string length by mapping len to strList:",
    lenList, sep="\n\t", end="\n")


# 4) Add two lists element-wise
list1 = [1, 1, 2]
list2 = [0, 1, 1]
sumList = map(twoSum, list1, list2)
```

```python
print("\n=== The new sum list is:", list(sumList), sep="\n\t", end="\n")


# 5) Using map with multiple iterables of different lengths
list3 = [1, 1, 2, 5, 10, 0, 100]
multList = list( map(threeSum, list1, list2, list3) )
print("\n=== The multiple variable list is:", list(multList), sep="\n\t", end="
    \n")

# Alternatively, this can also be done with list comprehension and zip !
multList = [ x + y + z for x,y,z in zip(list1, list2, list3) ]
print("\n=== The multiple variable list with list comprehension is:", list(
    multList), sep="\n\t", end="\n")

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== My original list is:
  [0, 2, 4, 6, 8]

=== Square each number by mapping xSquared() to numList iterable:
   [0, 4, 16, 36, 64]

=== Convert each number to string by mapping str to numList:
  ['0', '2', '4', '6', '8']

=== Convert each number to string length by mapping len to strList:
  [3, 3, 5]

=== The new sum list is:
  [1, 2, 3]

=== The multiple variable list is:
  [2, 3, 5]

=== The multiple variable list with list comprehension is:
  [2, 3, 5]

=== Quit!
```