In [2]:
```
print('Part 1: Primer')
print('\n1: Lists')


#1) Make a list with the spelled-out number strings 'one', 'two', 'three', 'four'
print('\n1. Make a list with the spelled-out number strings 'one', 'two', 'three'
myList = ['one', 'two', 'three', 'four', 'five']
listp = str(myList)
print('myList = ' + listp)

#2) Remove 'three' from the list using positional indexing.

print('\n2. Remove 'three' from the list using positional indexing.')
myList.remove('three')
print(myList)


#3) Check if 'four' is in the list.

print('\n3. Check if 'four' is in the list.')

if 'four' in myList: print("Four is present in the list")

#4) Append 'six' to the end of the list, then print the length of the list.

print('\n4. Append 'six' to the end of the list, then print the length of the lis
myList.append('six')
print("Length of the list after appending six is: ",len(myList))

#5) Print the contents of the list, but also next to each item print the length o
print('\n5. Print the contents of the list, but also next to each item print the
listl = []
for i in range(len(myList)) :
    length = len(myList[i])
    print(myList[i], length )
    listl.insert(i,length)


 #6) Create a list only of the lengths of the strings and show your result. You c
print('\n6. Create a list only of the lengths of the strings and show your result
print(listl)


print('\n2: Dictionaries')

#1) Make a dictionary with the keys be English words as below, and the values be
print('\n1. Make a dictionary with the keys be English words as below, and the va
#apple - Apfel
print('apple - Apfel')
#apples - Äpfel
print('apples - Äpfel')
#I - Ich
print('I - Ich')
#and - und
print('and - und')
#like - mag
```

```
print('like - mag')
#strawberries - Erdbeeren
print('strawberries - Erdbeeren')

engdict = {
  "apple": "Apfel",
  "apples": "Äpfel",
  "I": "Ich",
  "and": "und",
  "like": "mag",
  "strawberries": "Erdbeeren"
}
print()
print(engdict.items())

#2) Use the dictionary to look up the translation for ‘apple’ and ‘like’.
print('\n2. Use the dictionary to look up the translation for ‘apple’ and ‘like’.
x = engdict["apple"]
print('Traslation for apple is ' + x)
x = engdict["like"]
print('Traslation for like is ' + x)
#3) Make a variable var with the string “I like apples and strawberries”
print('\n3. Make a variable var with the string “I like apples and strawberries”'
var = 'I like apples and strawberries'
print(var)

#4) Now create a list from var with each word a separate item (this is a string s
print('\n4. Now create a list from var with each word a separate item (this is a
st = var.split()
print(st)

#5) Iterate through the list you’ve created and replace any word in your dictiona
print('\n5. Iterate through the list you’ve created and replace any word in your
for i in range(0,len(st)):
    word = st[i]
    y = engdict.get(word)
    st[i] = y

print(st)

#6) Now take your new list and turn it into a string with spaces between the word
print('\n6. Now take your new list and turn it into a string with spaces between

var = ' '.join(st)

print(var)

print('\n3: Arrays')

#1) Create an array of zeros of size 8 x 8 and print the data type of the array.
print('\n1. Create an array of zeros of size 8 x 8 and print the data type of the

import numpy as np

arr = np.zeros(shape = (8,8))

print(arr)
```

```
#2) Fill the array with the numbers 1 to 64 first by row, then by column. You may
print('\n2. Fill the array with the numbers 1 to 64 first by row, then by column.
x = 1
for i in range(0,8):
    for j in range(0,8):
          arr[i][j] = x
          x = x+1
print(arr)

#3) Transpose the array.
print('\n3. Transpose the array.')
transpose = np.zeros(shape = (8,8))
# Iterate through rows
for i in range(0,8):
   #Iterate through columns
   for j in range(0,8):
        transpose[j][i] = arr[i][j]

arr = transpose
print(arr)

#4) Print only the top 4 rows and columns.
print('\n4. Print only the top 4 rows and columns.')
res = np.zeros(shape = (4,4))
# Iterate through rows
for i in range(0,4):
   #Iterate through columns
   for j in range(0,4):
        res[i][j] = arr[i][j]

print(res)

#5) Make a 1D array out of your 2D array with the numbers 1 to 64 in order (note
print('\n5. Make a 1D array out of your 2D array with the numbers 1 to 64 in orde
re = np.zeros(shape = (1,64))
# Iterate through rows
y = 0
for i in range(0,8):
     for j in range(0,8):
           re[0][y] = arr[j][i]
           y = y+1

print(re)

#6) Now take that 1D array you made from before and reshape it back to the origir
print('\n6. Now take that 1D array you made from before and reshape it back to th

# Iterate through rows
y = 0
for i in range(0,8):
     for j in range(0,8):
           arr[i][j] = re[0][y]
           y = y+1

print(arr)
```

Part 1: Primer

1: Lists

1. Make a list with the spelled-out number strings ‘one’, ‘two’, ‘three’, ‘fou
r’, and ‘five’ in that order and call it myList.
myList = ['one', 'two', 'three', 'four', 'five']

2. Remove ‘three’ from the list using positional indexing.
['one', 'two', 'four', 'five']

3. Check if ‘four’ is in the list.
Four is present in the list

4. Append ‘six’ to the end of the list, then print the length of the list.
Length of the list after appending six is: 5

5. Print the contents of the list, but also next to each item print the length
of the string (e.g. one is 3, four is 4) using a for loop.
one 3
two 3
four 4
five 4
six 3

6. Create a list only of the lengths of the strings and show your result. You c
an use the loop before to fill the list.
[3, 3, 4, 4, 3]

2: Dictionaries

1. Make a dictionary with the keys be English words as below, and the values be
the translation. You can use this language example (German) or choose your own.
Note: you need to make sure all of these words are represented as strings, in q
uotes.
apple - Apfel
apples - Äpfel
I - Ich
and - und
like - mag
strawberries - Erdbeeren

dict_items([('apple', 'Apfel'), ('apples', 'Äpfel'), ('I', 'Ich'), ('and', 'un
d'), ('like', 'mag'), ('strawberries', 'Erdbeeren')])

2. Use the dictionary to look up the translation for ‘apple’ and ‘like’.
Traslation for apple is Apfel
Traslation for like is mag

3. Make a variable var with the string “I like apples and strawberries”
I like apples and strawberries

4. Now create a list from var with each word a separate item (this is a string
split operation).
['I', 'like', 'apples', 'and', 'strawberries']

5. Iterate through the list you’ve created and replace any word in your diction

5. Iterate through the list you've created and replace any word in your diction
ary with the translation.
['Ich', 'mag', 'Äpfel', 'und', 'Erdbeeren']

6. Now take your new list and turn it into a string with spaces between the wor
ds.
Ich mag Äpfel und Erdbeeren

3: Arrays

1. Create an array of zeros of size 8 x 8 and print the data type of the array.
```
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]
```

2. Fill the array with the numbers 1 to 64 first by row, then by column. You ma
y want to use a for loop inside a for loop to do this.
```
[[ 1.  2.  3.  4.  5.  6.  7.  8.]
 [ 9. 10. 11. 12. 13. 14. 15. 16.]
 [17. 18. 19. 20. 21. 22. 23. 24.]
 [25. 26. 27. 28. 29. 30. 31. 32.]
 [33. 34. 35. 36. 37. 38. 39. 40.]
 [41. 42. 43. 44. 45. 46. 47. 48.]
 [49. 50. 51. 52. 53. 54. 55. 56.]
 [57. 58. 59. 60. 61. 62. 63. 64.]]
```

3. Transpose the array.
```
[[ 1.  9. 17. 25. 33. 41. 49. 57.]
 [ 2. 10. 18. 26. 34. 42. 50. 58.]
 [ 3. 11. 19. 27. 35. 43. 51. 59.]
 [ 4. 12. 20. 28. 36. 44. 52. 60.]
 [ 5. 13. 21. 29. 37. 45. 53. 61.]
 [ 6. 14. 22. 30. 38. 46. 54. 62.]
 [ 7. 15. 23. 31. 39. 47. 55. 63.]
 [ 8. 16. 24. 32. 40. 48. 56. 64.]]
```

4. Print only the top 4 rows and columns.
```
[[ 1.  9. 17. 25.]
 [ 2. 10. 18. 26.]
 [ 3. 11. 19. 27.]
 [ 4. 12. 20. 28.]]
```

5. Make a 1D array out of your 2D array with the numbers 1 to 64 in order (note
the column vs row issue, you may need transposes.)
```
[[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
  19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36.
  37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54.
  55. 56. 57. 58. 59. 60. 61. 62. 63. 64.]]
```

6. Now take that 1D array you made from before and reshape it back to the origi
nal 2D array.
```
[[ 1.  2.  3.  4.  5.  6.  7.  8.]
 [ 9. 10. 11. 12. 13. 14. 15. 16.]
```

```
[ 9. 10. 11. 12. 13. 14. 15. 16.]
[17. 18. 19. 20. 21. 22. 23. 24.]
[25. 26. 27. 28. 29. 30. 31. 32.]
[33. 34. 35. 36. 37. 38. 39. 40.]
[41. 42. 43. 44. 45. 46. 47. 48.]
[49. 50. 51. 52. 53. 54. 55. 56.]
[57. 58. 59. 60. 61. 62. 63. 64.]]
```

In [57]:
```
print('Part 2: Applications')

print('1. Word counts')

print('i.First convert the string to a list with each word a separate item in the

#string = 'how much wood would a woodchuck chuck if a woodchuck could chuck wood

string = input('Enter a string: ')

l = string.split()
print(l)

print('ii. Then use a dictionary to associate each word with a count. Note, the c
d = {}
for i in range(len(l)):
    if l[i] in d.keys():
        d[l[i]] += 1
    else:
        d[l[i]] = 1
print(d)
print('iii. Print each word and its count afterwards, and test with an interestir
for i,j in d.items() :
    print(i,j)
```

```
Part 2: Applications
1. Word counts
i.First convert the string to a list with each word a separate item in the li
st
Enter a string: how much wood would a woodchuck chuck if a woodchuck could ch
uck wood
['how', 'much', 'wood', 'would', 'a', 'woodchuck', 'chuck', 'if', 'a', 'woodc
huck', 'could', 'chuck', 'wood']
ii. Then use a dictionary to associate each word with a count. Note, the dict
ionary won’t be able to increment a key unless you add it first, so you may h
ave to check to see if it exists before setting the original count of a word
to 1.
{'how': 1, 'much': 1, 'wood': 2, 'would': 1, 'a': 2, 'woodchuck': 2, 'chuck':
2, 'if': 1, 'could': 1}
iii. Print each word and its count afterwards, and test with an interesting b
lock of text that will have multiple words counted multiple times.
how 1
much 1
wood 2
would 1
a 2
woodchuck 2
chuck 2
if 1
could 1
```

In [59]:
```
#Betty Botter bought a bit of butter The butter Betty Botter bought was a bit bit
print('1. Word counts')

print('i.First convert the string to a list with each word a separate item in the

#string = 'how much wood would a woodchuck chuck if a woodchuck could chuck wood

string = input('Enter a string: ')

l = string.split()
print(l)

print('ii. Then use a dictionary to associate each word with a count. Note, the c
d = {}
for i in range(len(l)):
    if l[i] in d.keys():
        d[l[i]] += 1
    else:
        d[l[i]] = 1
print(d)
print('iii. Print each word and its count afterwards, and test with an interestir
for i,j in d.items() :
    print(i,j)
```

```
1. Word counts
i.First convert the string to a list with each word a separate item in the li
st
Enter a string: Betty Botter bought a bit of butter The butter Betty Botter b
ought was a bit bitter And made her batter bitter But a bit of better butter
makes better batter So Betty Botter bought a bit of better butter Making Bett
y Botter bitter batter better
['Betty', 'Botter', 'bought', 'a', 'bit', 'of', 'butter', 'The', 'butter', 'B
etty', 'Botter', 'bought', 'was', 'a', 'bit', 'bitter', 'And', 'made', 'her',
'batter', 'bitter', 'But', 'a', 'bit', 'of', 'better', 'butter', 'makes', 'be
tter', 'batter', 'So', 'Betty', 'Botter', 'bought', 'a', 'bit', 'of', 'bette
r', 'butter', 'Making', 'Betty', 'Botter', 'bitter', 'batter', 'better']
ii. Then use a dictionary to associate each word with a count. Note, the dict
ionary won't be able to increment a key unless you add it first, so you may h
ave to check to see if it exists before setting the original count of a word
to 1.
{'Betty': 4, 'Botter': 4, 'bought': 3, 'a': 4, 'bit': 4, 'of': 3, 'butter':
4, 'The': 1, 'was': 1, 'bitter': 3, 'And': 1, 'made': 1, 'her': 1, 'batter':
3, 'But': 1, 'better': 4, 'makes': 1, 'So': 1, 'Making': 1}
iii. Print each word and its count afterwards, and test with an interesting b
lock of text that will have multiple words counted multiple times.
Betty 4
Botter 4
bought 3
a 4
bit 4
of 3
butter 4
The 1
was 1
bitter 3
```

```
And 1
made 1
her 1
batter 3
But 1
better 4
makes 1
So 1
Making 1
```

Type *Markdown* and LaTeX: α^2

In [4]:
```
print('2. Adding an Array Border')

print('\nTest with an array in the center that has non-zero elements. First print

import numpy as np

r = int(input('enter number of rows: '))
c = int(input('enter number of columns: '))

print('Enter the values in the matrix')

mat = np.zeros(shape = (r,c))
newmat = np.zeros(shape=(r+2,c+2))

for i in range(r):
    for j in range(c):
         mat[i][j] = int(input())
         newmat[i+1][j+1] = mat[i][j]

print('Below is the input matrix :')
print(mat)

print('Below is the Output matrix :')
print(newmat)
```

```
2. Adding an Array Border

Test with an array in the center that has non-zero elements. First print the or
iginal array, then print the new array with the added border of 0's.
enter number of rows: 3
enter number of columns: 4
Enter the values in the matrix
1
1
1
1
1
1
1
1
1
1
1
1
Below is the input matrix :
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
Below is the Output matrix :
[[0. 0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0.]]
```