Quick Exercises 1

1. Implement a function that takes as input three variables, and returns the largest of the three. Do this without using the Python max() function! Make one version without any local variable and another with one local variable.
2. Write a function “centenario” that will take Name, and year of birth as inputs, check if year of birth is int and cast it to int if not, and print name together with the text explaining when the person is to have 100 years (hint: use isinstance)

   call to function: centenario(Antonio, 1967)

   output: Antonio will reach 100 years in 2067.

Quick Exercises 1

1a)
```
def max_of_three (a, b, c):
    if a>b :
        if a>c :
            return a
        else:
            return c
    else:
        if b>c :
            return b
        else:
            return c
```

Quick Exercises 1

1.b)
```
def max_of_three (a, b, c):
    max_var=a
    if max_var<b:
        max_var=b
    if  max_var<c:
        max_var=c
    return max_var
```

2)
```
def centenario (nombre, year):
    if not isinstance(year, int):
        year=int(year)
    print "%s will reach 100 years in %d" %(nombre, year+100)
    return
```

Quick Exercises 2

1. Write a function to calculate the number of words, number of lines, and length of a string the same way the wc command does in the command line
2. Write a Python program to remove the n^{th} index character from a string. If the input string is empty print warning.

Quick Exercises 2

1) def wc(x):
    chars= 0
    for char in str1:
        chars+= 1
    words=len(x.split(“”))
    lines=len(x.split(“\n”))
    return (lines, words, chars)

2) def remove_char(str, n):
    if not str:
        print (“String is empty”)
        return
    else:
        first_part = str[:n]
        last_part = str[n+1:]
        return first_part + last_part

Quick Exercises 3

1. While inside python, go to ~/Data/opentraveldata/ and list the files. Repeat the same for /home/dsc/Data/us_dot/otp and ~/Data/us_dot/traffic/.

   Use the list of visited directories from dhist and write for loop which will return for each visited directory its name and number of files inside.

2. Write a function that will take text file and pattern as input parameters, and return the number of occurances of case insensitive pattern inside a text (similar to: grep -i –o pattern file | wc -l)

3. Open Finn.txt file, read lines into a list. Remove trailing white spaces from each line . Write the resulting list to the new file. How many lines does the new file have? (hint: empty list is made with [ ])

4. Open Finn.txt file, read lines into a list. Create a new version of Finn_nbl.txt with no blank lines.

5. Reset the workspace. Obtain the difference in number of lines between original Finn file and and the one without blank lines and print the result. (hint: use wc)

Quick Exercises 3

1)

```
for i in _dh:
        wc=!(ls -l $i| wc –l)
        print(i, int(wc[0])-1)
```

2)

```
def remove_char(str, n):
    if not str:
        print (“String is empty”)
        return
    else:
        first_part = str[:n]
        last_part = str[n+1:]
        return first_part + last_part
```

Quick Exercises 3

3)

```
path='~/Data/shell/Finn.txt'
f=open(path)
lines=[]
for line in f:
    lines.append(line)
f.close()
len(lines)
!wc -l Finn.txt

no_eol=[]
for line in lines:
   no_eol.append(line.rstrip())
len (no_eol)

f_out=open("Finn_no_eol.txt", 'w')
for line in no_eol:
   f_out.writeline(line)
f_out.close()
```

Quick Exercises 3

4)

```
path='~/Data/shell/Finn.txt'
f=open(path)
lines=[]
for line in f:
    lines.append(line)
f.close()
len(lines)


f_out=open("Finn_no_bl.txt", 'w')
for line in lines:
   if len(line)>2:
   f_out.writeline(line)
f_out.close()
```

What is the size of new line character in this file?

```
path='~/Data/shell/Finn.txt'
f=open(path)
lines=[]
for line in f:
    lines.append(line)

lines[:10] #get first 10 lines
lines[2]
lines[1] #is this just an empty line?
len(lines[1]) #what is its size?
```

Quick Exercises 3

5)

```
%reset
lines_no_bl=!wc -l Finn_no_bl.txt | cut -d " " -f 1
lines_org=!wc -l Finn.txt | cut -d " " -f 1
line_diff=int(lines_org[0])-int(lines_no_bl[0])
```

Quick Excercises 4

1. For a sequence [1, 2, 3, 4, 5, 6, 7, 8] get the squared values using the lambda function.
2. Prepare a list with 10 names. Make a code that will put all vowels to capitals and every other character to lower letters.
3. Prepare again a list with 10 names. Make a function with two input variables: list, and character; that returns a list of names containing one or more of input characters ’s inside the name.
4. Reverse word order from the input string
5. Create a function that accepts string as search string and returns number of lines with that string in a command history (hint : use a in b)
6. Write a Python function that takes a list of words and returns the length of the longest one.

Quick Excercises 4

1) a=[1, 2, 3, 4, 5, 6, 7, 8]
result=map(lambda x: x**2, a)

2) names=["Bob", "Igor", "Anita", "Israel", "Thomas", "Bill", "Ronaldo", "Messi"]

```
new_names=[]
for name in names:
      new_name='' #this is empty string
for letter in name:
   if letter in "aeiouAEIOU":
         new_name = new_name + letter.upper()
   else:
         new_name = new_name + letter.lower()
new_names.append(new_name)
```

Quick Excercises 4

3)

```
def inside(name_list, ch):
    names_of_interest = [ ]
    for name in name_list:
        if name.count(ch) >= 1:
            names_of_interest.append(name)
    return names_of_interest
```

Quick Excercises 4

4) Some of the possible solutions…

```
def reverse_v1(x):
    y = x.split()
    result = []
    for word in y:
        result.insert(0,word)
    return " ".join(result)
```

```
def reverse_v2(x):
  y = x.split()
  return " ".join(y[::-1])
```

```
def reverse_v3(x):
  y = x.split()
  return " ".join(reversed(y))
```

```
def reverse_v4(x):
  y = x.split()
  y.reverse()
  return " ".join(y)
```

Quick Excercises 4

```
5) def count_histlines_with(x):
        coun_lines=0
        for i in ln:
            if x in i:
                coun_lines +=1
         return coun_lines

6) def find_longest_word(words_list):
        word_len = []
        for n in words_list:
             word_len.append((len(n), n))
         word_len.sort()
         return word_len[-1][1]
```