LIVE 2 : Let's solve slightly 'harder' problems than in the previous session

- Yesterday's focus: Simple programs, reading error-messages, reading references, fixing bugs
- Today's focus: Mathematical programming problems relevant to ML/AI, common-errors, testing code, loops/iteration and recursion.
- Prereq: Python-programming. I will introduce "new" concepts as we go.
- I have asked some of these problems and thier variations in actual interviews.

In [97]:

```
# check Python version to avoid version-related bugs/errors
import sys
print (sys.version)
```

```
3.7.3 (default, Mar 27 2019, 09:23:15)
[Clang 10.0.1 (clang-1001.0.46.3)]
```

Problem 1: Find peaks/max in an list

- problem definition

In [98]:

```
A = [1,3,4,5,7,6,4,5,10,1];
print(A)
```

```
[1, 3, 4, 5, 7, 6, 4, 5, 10, 1]
```

In [99]:

```
# Peaks: 7,10

# Arrive at the logic and what coding-constructs we ahev to use to solve it.


# Peak: A[i-1] <= A[i] >= A[i+1]

for i in range(1, A.size - 1): # 1 and -1 are important
    if (A[i] >= A[i-1]) and (A[i] >= A[i+1]):
        print(A[i]);
```

```
---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-99-71afdfbe8dc6> in <module>
      6 # Peak: A[i-1] <= A[i] >= A[i+1]
      7
----> 8 for i in range(1, A.size - 1): # 1 and -1 are important
      9     if (A[i] >= A[i-1]) and (A[i] >= A[i+1]):
     10         print(A[i]);

AttributeError: 'list' object has no attribute 'size'
```

In [5]:

```
print(len(A));
```

```
10
```

In [6]:

```
for i in range(1, len(A) - 1): # 1 and -1 are important
    if (A[i] >= A[i-1]) and (A[i] >= A[i+1]):
        print(A[i]);
```

```
7
10
```

In [8]:

```
# There is a bug in the above solution. Find it...
```

In [9]:

```
A = [11,3,4,5,7,6,4,5,10,15];

for i in range(1, len(A) - 1): # 1 and -1 are important
    if (A[i] >= A[i-1]) and (A[i] >= A[i+1]):
        print(A[i]);
```

```
7
```

In [11]:

```
# what about 11 and 15?

# boundary case: DONT MISS THEM. Very important in interviews
if A[0] >= A[1]:
    print(A[0])

for i in range(1, len(A) - 1): # 1 and -1 are important
    if (A[i] >= A[i-1]) and (A[i] >= A[i+1]):
        print(A[i]);

# boundary case
if A[len(A)-1] >= A[len(A)-2]:
    print(A[len(A)-1])
```

```
11
7
15
```

In [12]:

```
# LESSON: Donot forget boundary cases

#Question: Time Complexity = ?
```

Exercise:

1. Find peaks which are max values in a window of size 2 on both sides.
2. Use scipy.signal.find_peaks
   [https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html
   (https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html)]

Additional reading: More efficient algos using D&Q @
http://courses.csail.mit.edu/6.006/spring11/lectures/lec02.pdf
(http://courses.csail.mit.edu/6.006/spring11/lectures/lec02.pdf)

Problem 2: Permutations of a list

- Problem definition: lst = [1,2,3]

In [101]:

```
# Using libraries: the easy way
from itertools import permutations

p = permutations([1, 2, 3]) # refer: https://docs.python.org/3/library/itertool
s.html#itertools.permutations

# what is an iterable in Python [https://www.pythonlikeyoumeanit.com/Module2_Ess
entialsOfPython/Iterables.html]

# print the permutations
for i in p:
    print(i)
```

```
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)
```

In [15]:

```
# No fun!
# Write your own code. Any suggestions?
# Break the problem into smaller sub-problems


[1,2,3]
[1,3,2]
[2,1,3]
[2,3,1]
[3,1,2]
[3,2,1]
```

Out[15]:

```
[3, 2, 1]
```

In [16]:

```
# source: https://stackoverflow.com/questions/13109274/python-recursion-permutat
ions

# NOT an optimal code.

def permutation(s):
   if len(s) == 1:
     return [s]

   perm_list = [] # resulting list
   for a in s:
     remaining_elements = [x for x in s if x != a]
     z = permutation(remaining_elements) # permutations of sub-list

     for t in z:
       perm_list.append([a] + t)

   return perm_list

s = [1,2,3];
p = permutation(s);
print(p)
```

```
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
```

In [17]:

```
# Time-Complexity: ?
```

Exercise: [Combinations] All possible ways to pick 3 elements out of 5 elements

- HINT: Use recursion. Lets work out the logic first! "Pick one and recurse"
- Google "combinations in Python" for libraries-based solution.

Problem 3: Solving for x: sin(x) = cos(x)

- Plot cos(x)-sin(x) using google: "plot cos(x)-sin(x)"

In [19]:

```
# one possible value of x lies in [0, 1]
# Any suggestions?
```

In [21]:

```
# Observation: cos(x)-sin(x) is monotonic in [0,1]
# HINT: -----
```

In [36]:

```
import math

def f(x):
    return math.cos(x) - math.sin(x);

# init
x_l =0;
x_u =1;
x = (x_u + x_l)/ 2;

#iterate
while ( f(x) > 0.001): # till we are very close to zero

    print(x_l, x_u)

    x = (x_u + x_l)/2 ; # middle point
    if f(x) > 0: # adjust x_l
        x_l = x;
    else:
        x_u = x; # adjust x_u

    print(f(x), x_l, x_u)

print("x:" + str(x) + "\t f(x): " + str(f(x)))
```

```
0 1
0.39815702328616975 0.5 1
0.5 1
0.05005010885048666 0.75 1
0.75 1
-0.12654664407270177 0.75 0.875
x:0.875  f(x): -0.12654664407270177
```

In [37]:

```
import math

def f(x):
    return math.cos(x) - math.sin(x);

# init
x_l =0;
x_u =1;
x = (x_u + x_l)/ 2;

#iterate
while ( math.abs(f(x)) > 0.001): # till we are very close to zero

    print(x_l, x_u)

    x = (x_u + x_l)/2 ; # middle point
    if f(x) > 0: # adjust x_l
        x_l = x;
    else:
        x_u = x; # adjust x_u

    print(f(x), x_l, x_u)

print("x:" + str(x) + "\t f(x): " + str(f(x)))
```

```
---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-37-207102861846> in <module>
     10
     11 #iterate
---> 12 while ( math.abs(f(x)) > 0.001): # till we are very close to
zero
     13
     14     print(x_l, x_u)

AttributeError: module 'math' has no attribute 'abs'
```

In [103]:

```
import math

def f(x):
    return math.cos(x) - math.sin(x);

# init
x_l =0;
x_u =1;
x = (x_u + x_l)/ 2;

#iterate
while ( abs(f(x)) > 0.0001): # till we are very close to zero

    print(x_l, x_u)

    x = (x_u + x_l)/2 ; # middle point
    if f(x) > 0: # adjust x_l
        x_l = x;
    else: # adjust x_u
        x_u = x;

    print(f(x), x_l, x_u)

print("x:" + str(x) + "\t f(x): " + str(f(x)))
```

```
0 1
0.39815702328616975 0.5 1
0.5 1
0.05005010885048666 0.75 1
0.75 1
-0.12654664407270177 0.75 0.875
0.75 0.875
-0.038323093040207645 0.75 0.8125
0.75 0.8125
0.005866372111545948 0.78125 0.8125
0.78125 0.8125
-0.01623034166690185 0.78125 0.796875
0.78125 0.796875
-0.005182142923325084 0.78125 0.7890625
0.78125 0.7890625
0.0003421172042541 4427 0.78515625 0.7890625
0.78515625 0.7890625
-0.002420017475350922 0.78515625 0.787109375
0.78515625 0.787109375
-0.0010389506309586016 0.78515625 0.7861328125
0.78515625 0.7861328125
-0.0003484167548866157 0.78515625 0.78564453125
0.78515625 0.78564453125
-3.149775409938549e-06 0.78515625 0.785400390625
x:0.785400390625         f(x): -3.149775409938549e-06
```

LESSON: Bisection-method uses binary search in the given interval.

Exercise: Solve for x in x^5 - x^4 + 2*x^3 - x^2 + x = 3

- Follow the same steps as above

Additional reading: Newton-Raphson Method

- Explnantion: https://brilliant.org/wiki/newton-raphson-method/ (https://brilliant.org/wiki/newton-raphson-method/) [Faster than bisection]
- Code: https://www.geeksforgeeks.org/program-for-newton-raphson-method/ (https://www.geeksforgeeks.org/program-for-newton-raphson-method/)
- We will cover gradient based methods when we arrive at Optimization topics in the course.

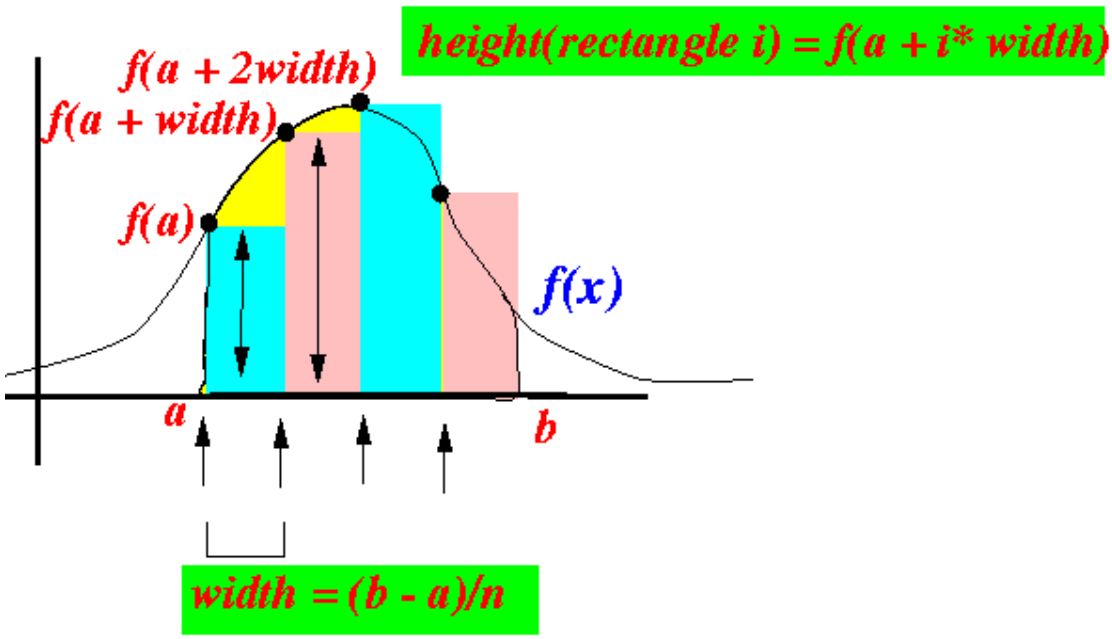
Problem 4: Find area under a curve sin(x)/x in the interval [-10, 10]

- Gogole "plot sin(x)/x"
- Sorry, "Gogol" is a russian author. Please "Google"!

In [40]:

```
# images-source: http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/rectangle-method.html
from IPython.display import Image
Image(url= "http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/FIGS/rectangle05.gif")
```

Out[40]:



In [41]:

```
Image(url= "http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/FIGS/rectangle06.gif")
```

Out[41]: