G12-CS-20190912_100000

September 18, 2019

1 Dragon Academy

2 G12 ICS4U Computer Science 11/09/2019 - 12/09/2019

3 Plotting

[1]:
```
import matplotlib.pyplot as plt

x = list( range(-10,11) )
y = list( range(20,41) )
plt.xlabel('x')
plt.ylabel('y')
plt.plot( x, y, 'or')
plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

[2]:
```
import numpy as np
```

[3]:
```
X = np.linspace(-10,11,100)
Y = 2*X+ 15
```

[4]:
```
X[:5], Y[:5]
```

[4]:
```
(array([-10.        ,  -9.78787879,  -9.57575758,  -9.36363636,
         -9.15151515]),
 array([-5.        , -4.57575758, -4.15151515, -3.72727273, -3.3030303 ]))
```

3.1 Exercise: Plot the parabola 3*(x-7)^2-12

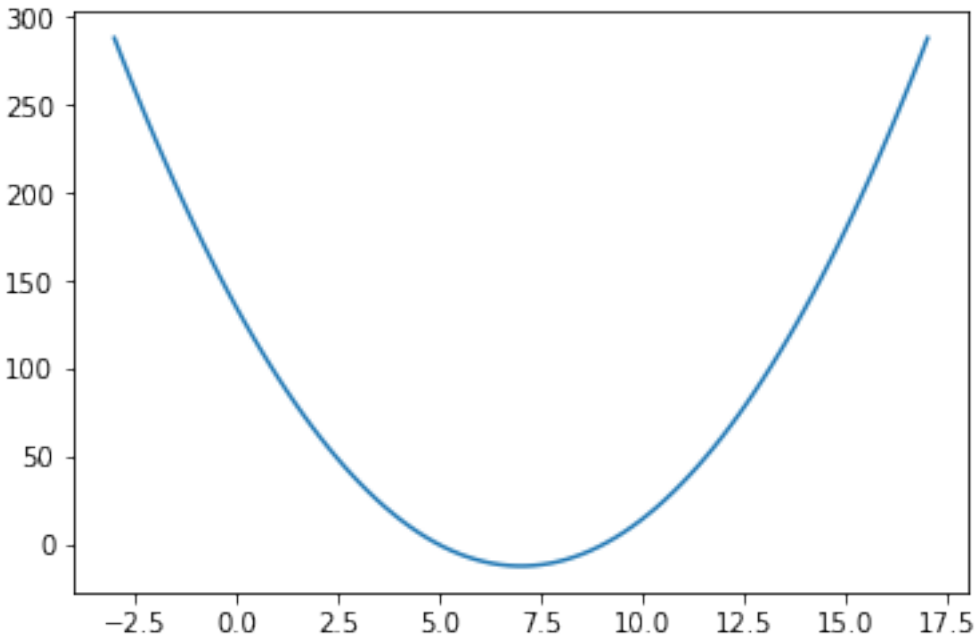
Make sure the plot is more or less symmetrical w/r to the minimum

[5]:
```
X = np.linspace(-3,17,100)
Y =3*(X-7)**2 - 12
# let's slice X to see the first 10 entries
X[:10], Y[:10]
```

[5]:
```
(array([-3.        , -2.7979798 , -2.5959596 , -2.39393939, -2.19191919,
        -1.98989899, -1.78787879, -1.58585859, -1.38383838, -1.18181818]),
 array([288.        , 276.00122436, 264.2473217 , 252.73829201,
        241.47413529, 230.45485155, 219.68044077, 209.15090297,
        198.86623814, 188.82644628]))
```

[6]:
```
plt.plot(X,Y)
```

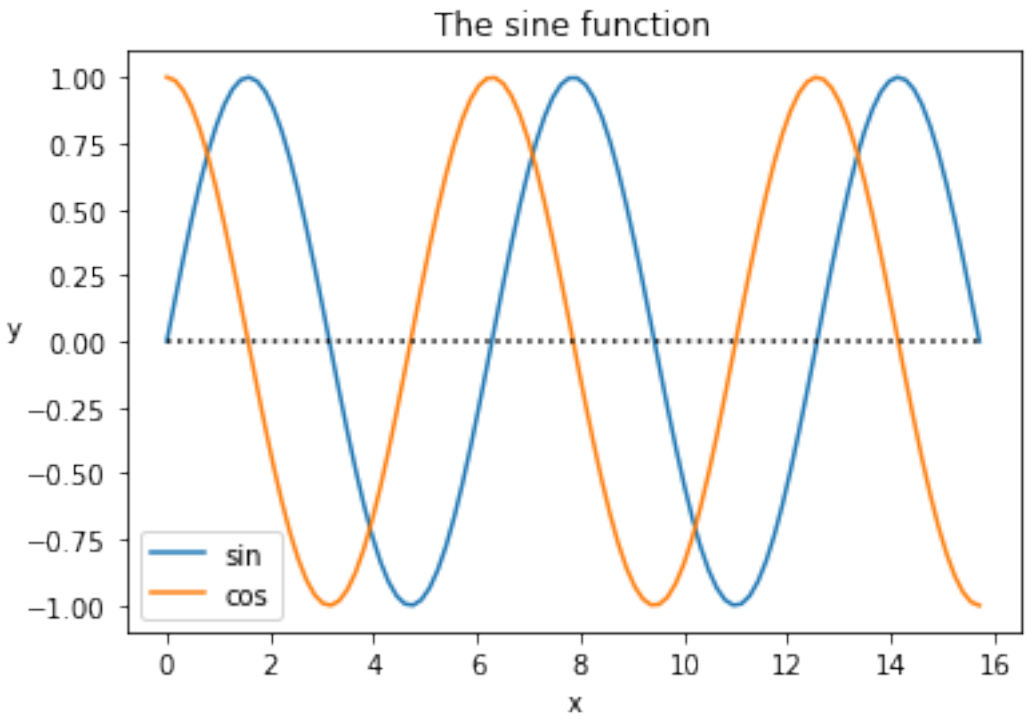
[6]:
```
[<matplotlib.lines.Line2D at 0x112ba8b00>]
```



3.2 Exercise: plot the sin(x) function

[7]:
```
X =np.linspace(0,5*np.pi,100)
Y = np.sin(X)
plt.plot(X,Y)
Y2 = np.cos(X)
plt.plot(X,Y2)
plt.plot(X,X*0,':k') # : stands for dotted; k for black
plt.legend(['sin','cos'])
plt.xlabel('x')
plt.ylabel('y',rotation=0)  # notice the y-axis label is facing straight up.␣
 ↪Try removing the "rotation=0" parameter
plt.title("The sine function")
```

[7]:
```
Text(0.5, 1.0, 'The sine function')
```



3.3 Summary

1. matplotlib.pyplot as plt
    1. plt.plot(x,y)
    2. plt.title, plt.legend,....
2. numpy as np
    1. x=np.linspace(-3,8,50)
    2. np.pi
    3. np.sin(x) , np.cos(x), np.tan(x) , np.exp(x), np.sqrt(x),...

4 I/O

1. Interactive. Reading one input line at a time.
2. Reading a file
3. Writing to a file
4. Appending to a file

Question: What about reading data from say a temperature sensor?

Answer: Both in OSX and Linux/*nix, the rule of thumb is **everything is a file**. *Hence, the answer is we would use the same functions as for reading a file! ;-) Windows, of course, does thing differently:* everything is a class/object *...Advanced: You can read more about this and other* paradigms* in this stackexchange thread as well as here, here and here

4.1 1 Basic Interactive Input

[8]:
```
n = 36347 # input() # delete the number en uncomment the call to input()␣
 ↪function to see this in effect
print("Your input was:"+str(n) )
```

```
Your input was:36347
```

4.2 2 Reading Files

Several ways:

1. Whole text as a string: open() + read()
2. Splitted into lines all in an array. Two variations of same theme:
    1. `with open(...) as f:` f becomes iterable, i.e., we can loop through it!!
    2. `lines = [ line for line in open(...)]`: List comprehension + open

4.2.1 1. Reading whole text as str

[9]:
```
# 1st way
fn='G9.ipynb'
text = open(fn).read()
text[:10]
```

[9]:
```
'{\n "cells"'
```

[10]:
```
text[:100]
```

[10]:
```
'{\n "cells": [\n  {\n   "cell_type": "markdown",\n   "metadata": {},\n
"source": [\n    "# G9 Intro to C'
```

4.2.2 2. Getting the file content splitted into lines. Several variations on the same theme:

1. with open() as f:

[11]:
```
with open(fn) as f:
    lmx=10
    for line in f:
        print(line)
        if lmx<1: break
        lmx-=1
```

```
{

 "cells": [

  {
```

```
 "cell_type": "markdown",

 "metadata": {},

 "source": [

  "# G9 Intro to Computer Technology 2019-2020"

 ]

},

{

 "cell_type": "markdown",
```

2. open() + List-comprehension (LC): Basic idea of LC is building the list content on the fly using a loop

[12]:
```
lines = [ line for line in open(fn)]
lines[:10]
```

[12]:
```
['{\n',
 ' "cells": [\n',
 '  {\n',
 '   "cell_type": "markdown",\n',
 '   "metadata": {},\n',
 '   "source": [\n',
 '    "# G9 Intro to Computer Technology 2019-2020"\n',
 '   ]\n',
 '  },\n',
 '  {\n']
```

[13]:
```
# Use a with-as contruct + for loop to make the variable lines contain all␣
 ↪lines of fn
lines=[]
with open(fn) as f:
    for line in f:
        lines.append(line)
lines[:10]
```

[13]:
```
['{\n',
 ' "cells": [\n',
 '  {\n',
 '   "cell_type": "markdown",\n',
 '   "metadata": {},\n',
 '   "source": [\n',
 '    "# G9 Intro to Computer Technology 2019-2020"\n',
```

```
 '   ]\n',
 '  },\n',
 '  {\n']
```

[14]:
```
lines[:5]
```

[14]:
```
['{\n',
 ' "cells": [\n',
 '  {\n',
 '   "cell_type": "markdown",\n',
 '   "metadata": {},\n']
```

[15]:
```
a=[12,4,2,6]
a.append(78)
a
```

[15]:
```
[12, 4, 2, 6, 78]
```

4.2.3 3. Writing to a file

First open it for writing, then write to it:

[16]:
```
# Writing the first 10 lines of the previous file we opened into a file name␣
 ↪foo.txt

of='foo.txt'

with open(of,'w') as f: # NOTICE the 'w' parameter: it means for writing!
    for line in lines[:10]:
        f.write(line)

# Let's check everything went ok: Read out foo.txt
with open(of,'r') as f: # NOTICE the 'r' parameter: it means for reading!
    i=1
    for line in f:
        print(i,":",line)
        i+=1
```

```
1 : {

2 :  "cells": [

3 :   {

4 :    "cell_type": "markdown",

5 :    "metadata": {},

6 :    "source": [
```

```
7 :     "# G9 Intro to Computer Technology 2019-2020"

8 :    ]

9 :   },

10 :   {
```

4.2.4 4. Appending Content to a file

Let's append a line to our previous foo.txt file and check that all indeed worked as expected.

[17]:
```
msg = "This line got appended at the end of this file by me Sep. 2019 :-)"
msg2 = "you know what? I decided to append a second one right after the␣
 ↪previous line.\nDid this work as we expected?"

with open(of,'a') as f: # NOTICE the 'a' parameter: it means for appending!
    f.write(msg)
    f.write(msg2)

# Let's check now again foo.txt's content
with open(of,'r') as f:
    i=1
    for line in f:
        print(i,':',line)
        i+=1
```

```
1 : {

2 :  "cells": [

3 :   {

4 :    "cell_type": "markdown",

5 :    "metadata": {},

6 :    "source": [

7 :     "# G9 Intro to Computer Technology 2019-2020"

8 :    ]

9 :   },

10 :   {
```

```
11 : This line got appended at the end of this file by me Sep. 2019 :-)you know
what? I decided to append a second one right after the previous line.

12 : Did this work as we expected?
```

Exercise:

1. **Rerun the previous two cells. How many copies of the appended messages `msg` and `msg2` do we get this time in foo.txt? Why?**
2. **Did the appending really work as expected? Explain why? What should we change?**

4.3 Assignment T1-1: Due Thu Sep 19th 2019

Problem A: Deficient, Perfect, and Abundant
Input file: dpa.in
Output file: dpa.out

Write a program that repeatedly reads a positive integer, determines if the integer is deficient, perfect, or abundant, and outputs the number along with its classification.

A positive integer, n, is said to be perfect if the sum of its proper divisors equals the number itself. (Proper divisors include 1 but not the number itself.) If this sum is less that n, the number is deficient, and if the sum is greater than n, the number is abundant.

The input starts with the number of integers that follow. For each of the following integers, your program should output the classification, as given below. You may assume that the input integers are greater than 1 and less than 32500.

Sample input

3
4
6
12

Sample output

4 is a deficient number.
6 is a perfect number.
12 is an abundant number.

Submission Once you have check that your code works, submit just the code (not the whole notebook) in a file as attachment msantos @ dragonacademy.org with the Subject:"*Assignment T1-1*"

One way you can create the code file from Jupyter is creating a new notebook, copy&pasting your solution in just one cell and then exporting the notebook (see `File` menu)

4.4 Note

Remember the 4 steps in order to solve programatically a problem:

1. Gather intuition about the problem.
2. Writing the Logic of the solution
3. Pseudo-Code
4. Implementing the algorithm

[ ]: