# Ploting με τη βιβλιοθήκη matplotlib

Σε αυτή τη διάλεξη θα ασχοληθούμε με τη δημιουργία γραφικών παραστάσεων μέσα από τη βιβλιοθήκη matplotlib [wiki]

Πριν ξεκινήσουμε να αναφέρουμε ότι το "πλοτάρισμα" δεν ανήκει παραδοσιακά στο μάθημα του προγραμματισμού, αλλά σε αυτό της ανάλυσης δεδομένων. Παρόλα αυτά, ο προγραμματισμός στη βιολογία συνήθως ταυτίζεται με την ανάλυση κάποιων δεδομένων.

## Το πλοτάρισμα είναι τέχνη!

Το καλό plot, δεν έχει να κάνει ούτε με το πόσο καλοί προγραμματιστές είσαστε αλλά ούτε και με τη ποιότητα των δεδομένων σας (αν και τα 2 βοηθάνε). Το καλό πλοτ έχει να κάνει με την αισθητική σας, την αντίληψή σας για τα χρώματα και την αίσθηση του μέτρου (οὐκ ἐν τῷ πολλῷ τὸ εὖ).

Για αυτό καλό είναι πριν (ή και μετά..) από αυτή τη διάλεξη να διαβάσουμε:

- Ten Simple Rules for Better Figures
  - Απόσπασμα: "All the figures for this article were produced using matplotlib, and figure scripts are available from https://github.com/rougier/ten-rules."
- Visual Analytics in Omics: why, what, how? by Jan Aerts
- Σχετικά με τον χειρισμό εικόνων: Creating clear and informative image-based figures for scientific publications
- Matplotlib for papers (Ίσως είναι λίγο παλιό αυτό)
- Πως να **MHN** κάνουμε plots:
  - Misleading graph
  - Google: σκαι γράφημα
- Εμπνεόμαστε από τα πολύ καλά plots του tableau

Ο συνηθισμένος τρόπος για να εισάγουμε (import) τη matplotlib είναι:

In [2]:
```
import matplotlib.pyplot as plt
```

Στη συνέχεια για κάθε νέο plot πρέπει να δημιουργούμε τα εξής δύο αντικείμενα το fig και το ax

In [4]:
```
fig, ax = plt.subplots()
```

Γενικότερα: η matplotlib έχει τρία βασικά αντικείμενα για τον χειρισμό των plots:

- ax : Χειρισμός των αξόνων, τις περισσότερες φορές θα ασχολούμαστε με αυτό το αντικείμενο
- fig : Χειρισμός του plot ως εικόνα.
- plt : Αυτό είναι είναι το pyplot αντικείμενο το οποίο έχουμε κάνει import. Περιέχει τις βασικές μεθόδους του ax και του fig αντικειμένου. Υπάρχει σαν βοηθητικό αντικείμενο, απλούστευσης ώστε να μην χρειάζεται κάποιος να "παίζει" με δύο αντικείμενα. π.χ. μπορεί κάποιος να γράψει: `ax.plot()` ή `plt.plot()`

παρομοίως μπορεί να γράψει `fig.show()` ή `plt.show()` . Παρόλα αυτά υπάρχουν μέθοδοι που είναι μέρος του ax αντικειμένου και δεν είναι του plt αντικειμένου. Και δεδομένου ότι η python είναι μία γλώσσα "There should be one-- and preferably only one --obvious way to do it." η ύπαρξη αυτών των επιλογών θεωρώ ότι είναι έξω από το πνεύμα της γλώσσας (είναι κυρίως ιστορικοί λόγοι που γίνεται αυτό). Για αυτό και δεν θα (πολύ) ασχοληθούμε με τη plt!

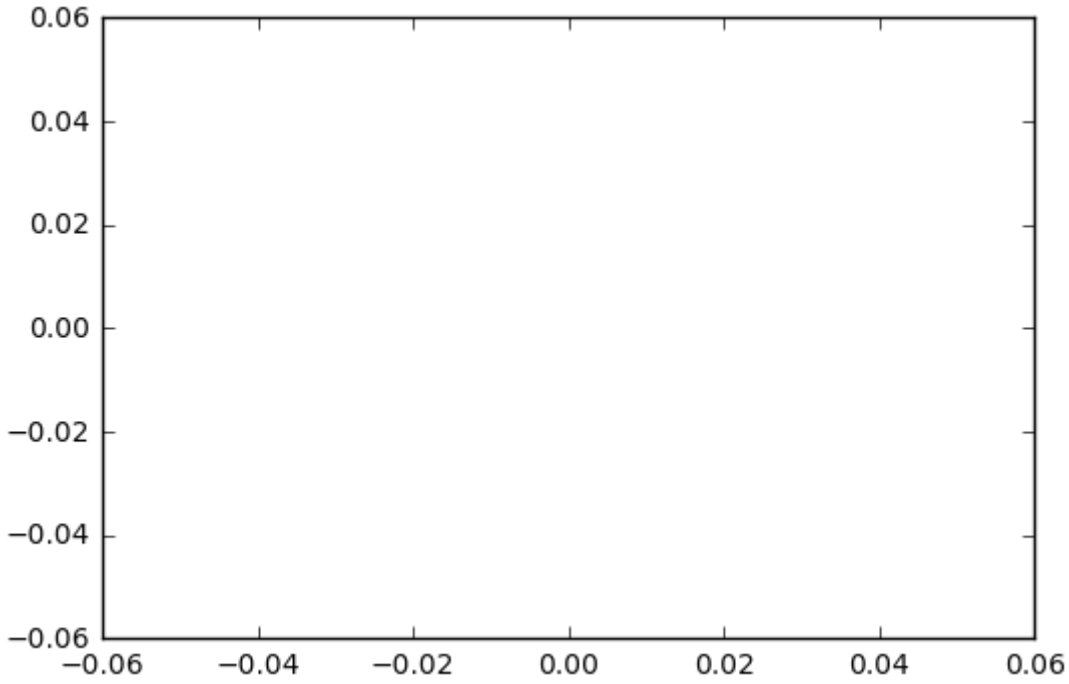
Ας φτιάξουμε ένα άδειο plot!

In [5]:
```
ax.plot()
```

Out[5]:
```
[]
```

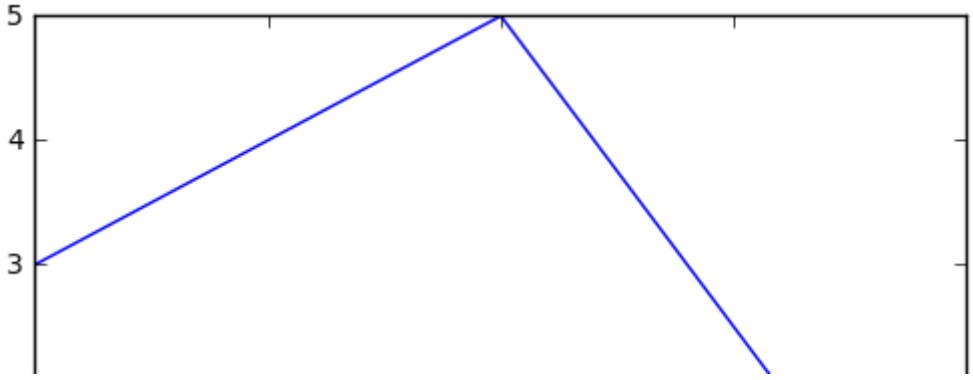
Για να εμφανίσουμε ένα plot χρησιμοποιούμε την εντολή `show()` της plt

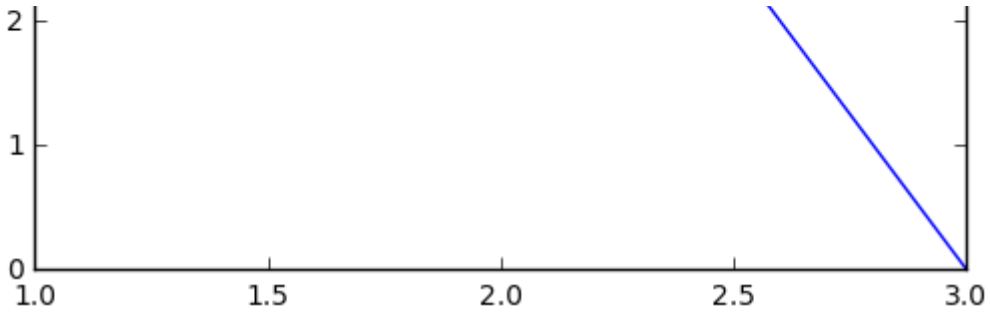
In [7]:
```
plt.show()
```




Η `ax.plot` δέχεται μία μεγάλη ποικιλία από ορίσματα. Τα δύο πρώτα ορίσματα είναι δύο λίστες. Η πρώτη περιέχει τις συντεταγμένες στον άξονα X των στοιχείων που θέλουμε να κάνουμε plot, και η δεύτερη τις συντεταγμένες στον άξονα Y. π.χ. Για να εμφανίσουμε μία τεθλασμένη γραμμή που περνάει από τα σημεία: (1,3), (2,5), (3,0) :

In [11]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0])
plt.show()
```






Η plot δέχεται και ένα τρίτο όρισμα το οποίο είναι το "στυλ" της γραμμής. Αποτελείται από δύο μέρη: χρώμα και στυλ. Το χρώμα μπορεί να είναι (http://matplotlib.org/api/colors_api.html):

- b: blue
- g: green
- r: red
- c: cyan
- m: magenta
- y: yellow
- k: black
- w: white

Σε περίπτωση που θέλουμε να κάνουμε plot κάποια γραμμή τότε το στυλ μπορεί να είναι είτε ένα από (http://matplotlib.org/api/lines_api.html#matplotlib.lines.Line2D.set_linestyle):

- '-' or 'solid' solid line
- '--' or 'dashed' dashed line
- '-.' or 'dashdot' dash-dotted line
- ':' or 'dotted' dotted line
- 'None' draw nothing
- ' ' draw nothing
- '' draw nothing

Σε περίπτωση που θέλουμε να κάνουμε plot μόνο τα σημεία (και όχι γραμμές) τότε οι επιλογές είναι (http://matplotlib.org/api/markers_api.html):

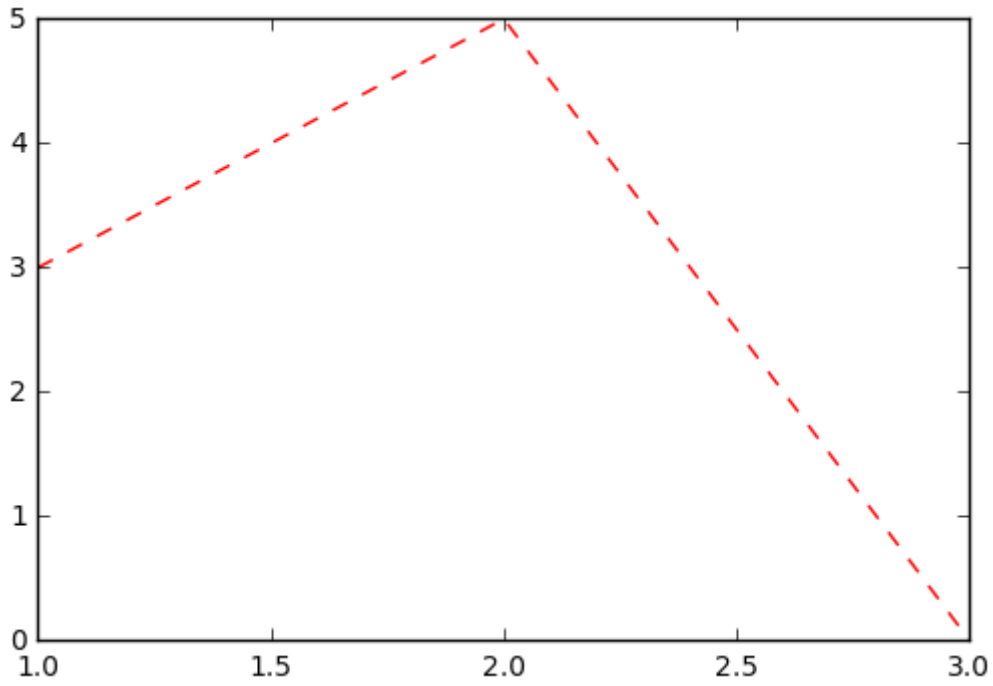
- ”.” point
- ”,” pixel
- “o” circle
- “v” triangle_down
- “^” triangle_up
- “<” triangle_left
- “>” triangle_right
- “1” tri_down
- “2” tri_up
- “3” tri_left
- “4” tri_right
- “8” octagon

- “s” square
- “p” pentagon
- “*” star
- “h” hexagon1
- “H” hexagon2
- “+” plus
- “x” x
- “D” diamond
- “d” thin_diamond
- “|” vline
- “_” hline
- TICKLEFT tickleft
- TICKRIGHT tickright
- TICKUP tickup
- TICKDOWN tickdown
- CARETLEFT caretleft
- CARETRIGHT caretright
- CARETUP caretup
- CARETDOWN caretdown
- “None” nothing
- None nothing
- ” “ nothing
- “” nothing

Υπάρχουν περισσότερες επιλογές και δυνατότητες για "καστομιές"¨: http://matplotlib.org

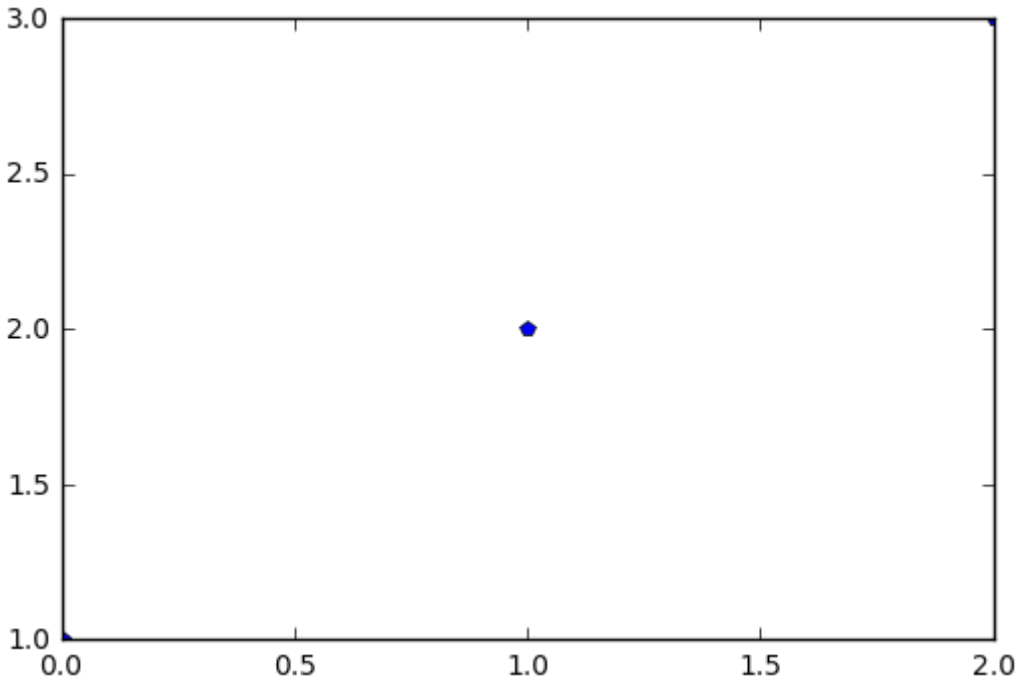
Οπότε για να πλοτάρουμε μία κόκκινη διακεκομένη γραμμή:

In [14]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0], 'r--')
plt.show()
```



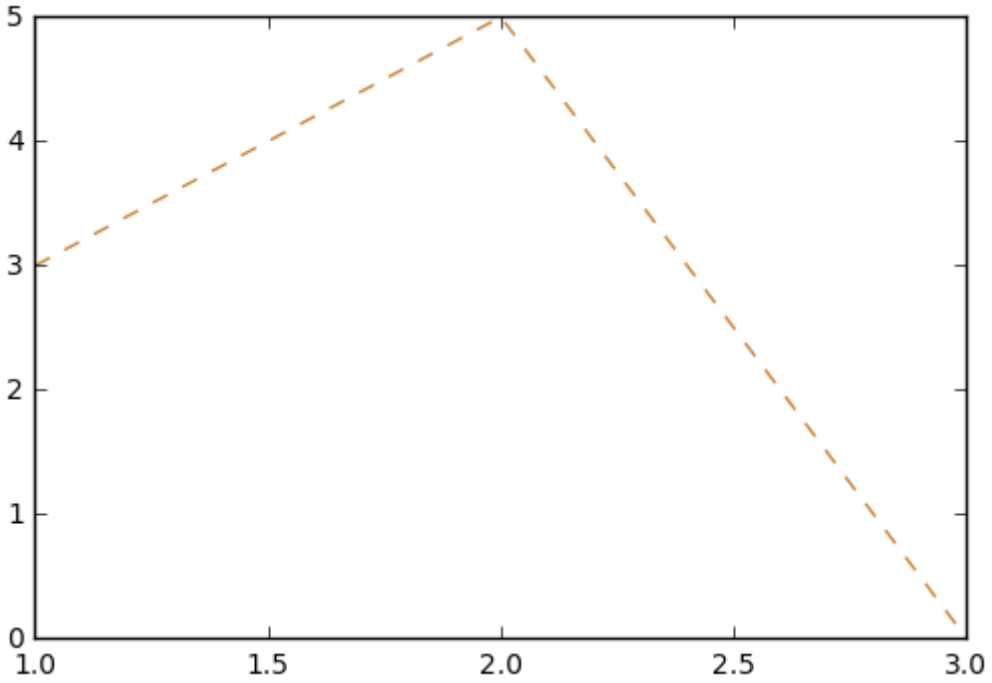
Για να πλοτάρουμε μπλε πεντάγωνα:

In [18]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], 'bp') # ΠΡΟΣΟΧΗ! εδώ πλοτάρουμε (1,1), (2,2), (3,3). bp: l
plt.show()
```



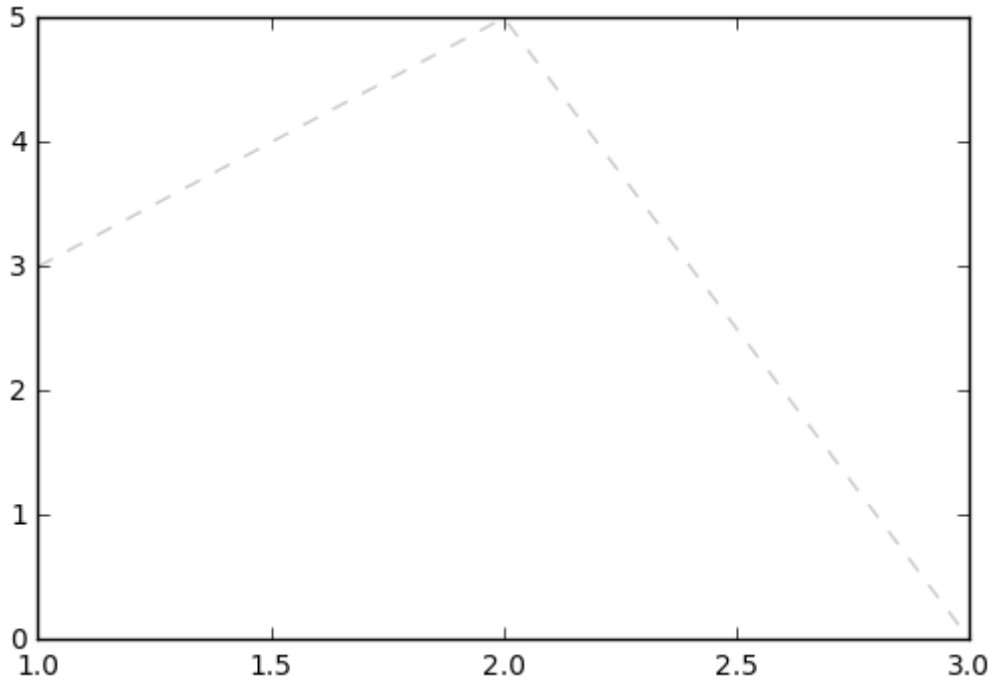

Φυσικά υπάρχει πιο πλούσιος τρόπος να ορίσουμε ένα χρώμα με την παράμετρο "c"

In [19]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0], '--', c="peru")
plt.show()
```



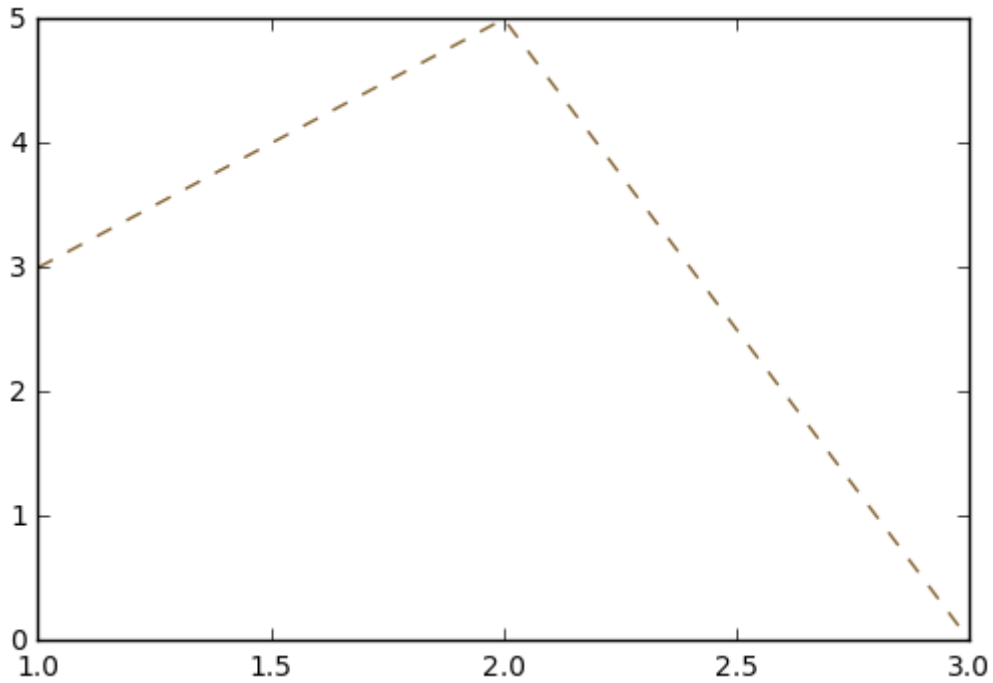

Ναι, υπάρχει χρώμα που λέγεται "peru". Πλήρη λίστα με ονόματα υπάρχει εδώ: (http://matplotlib.org/examples/color/named_colors.html) . Εναλλακτικά μπορείτε να χρησιμοποιείσετε μία τιμή από το 0.0 μέχρι το 1.0 για να τυπώσετε σε ένα "grayscale" όπου το 0.0 είναι το μαύρο και το 1.0 είναι το άσπρο:

In [20]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0], '--', c="0.8")
plt.show()
```



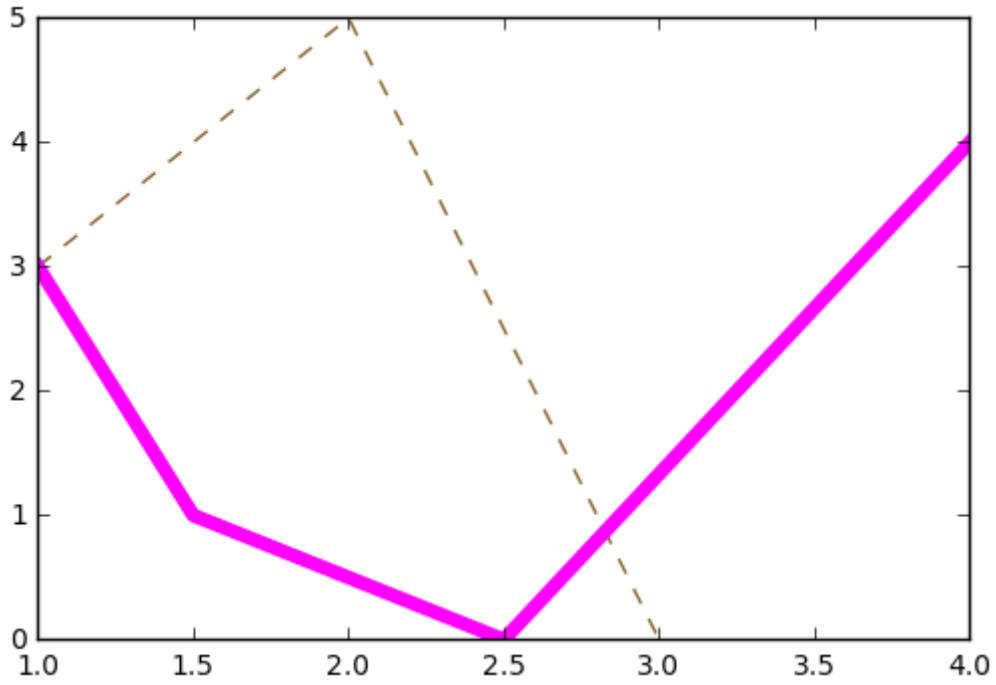

Φυσικά μπορείτε να χρησιμοποιήσετε ένα οποιοδίποτε RGB χρώμα. Υπάρχουν πολλά sites που μπορείς να επιλέξεις ένα χρώμα π.χ: http://htmlcolorcodes.com/

In [21]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0], '--', c="#876635")
plt.show()
```



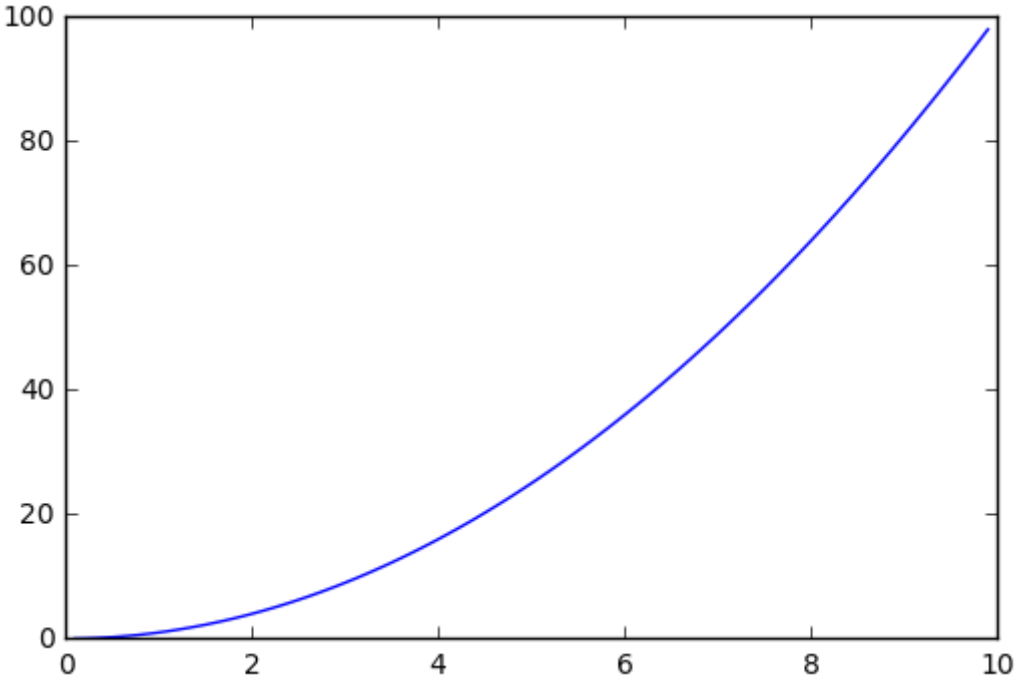

Μπορούμε να χρησιμοποιήσουμε πολλές φορές τη plot:

In [24]:
```
fig, ax = plt.subplots()
ax.plot([1,2,3], [3,5,0], '--', c="#876635")
ax.plot([1, 1.5, 2.5, 4], [3,1, 0, 4], '-', c="magenta", linewidth=5) # Pa
plt.show()
```



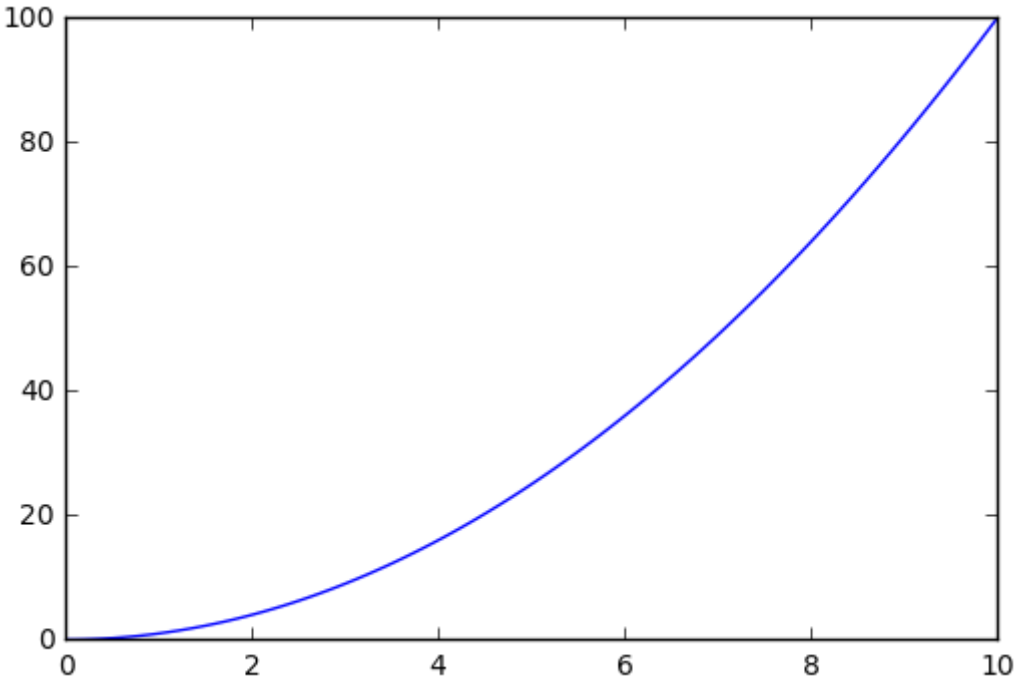

In [ ]:
```
Μπορούμε επίσης να κάνουμε plot μία συνάρτηση υπολογίζοντας τα X και τα Y
```

In [25]:
```
def f(x):
    return x**2 # X square

X = [x/10.0 for x in range(1,100)]
Y = [f(x) for x in X]

fig, ax = plt.subplots()
ax.plot(X,Y)
plt.show()
```



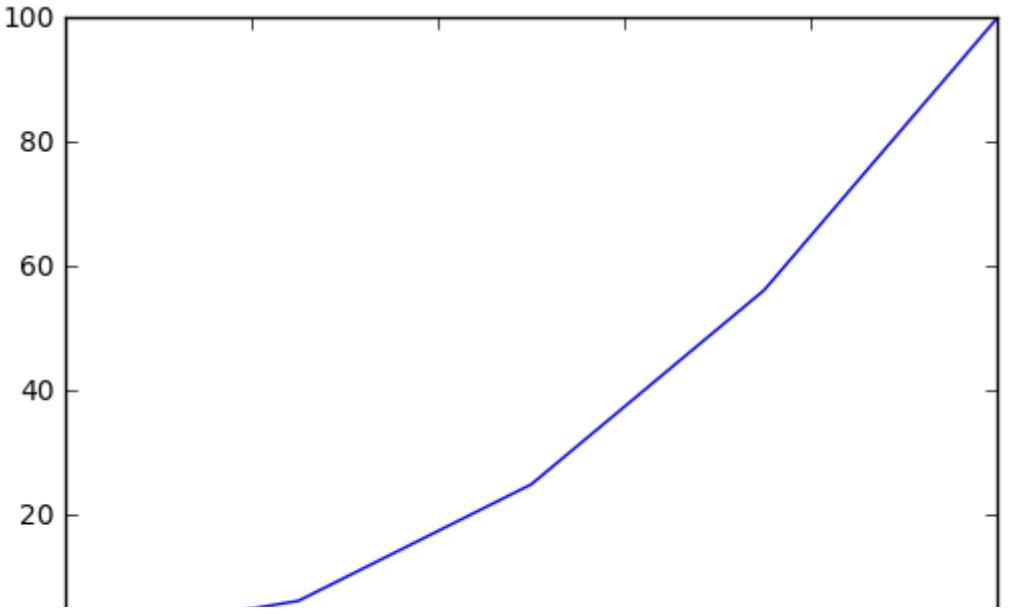

Ένας καλύτερος τρόπος για να πλοτάρουμε συναρτήσεις είναι να χρησιμοποιήσουμε τη linspace της numpy. Η linspace(a,b,c) δημιουργεί μία αριθμητική πρόοδο από a μέχρι b, έτσι ώστε να υπάρχουν συνολικά c στοιχεία.

In [27]:
```
import numpy as np

X = np.linspace(0, 10, 100) # 0 = min X, 10 = max X, 100 = resolution...
Y = [f(x) for x in X]

fig, ax = plt.subplots()
ax.plot(X,Y)

plt.show()
```




Το 100 στη linspace μπορούμε να το δούμε και ως την "ανάλυση της γραφικής παράστασης"

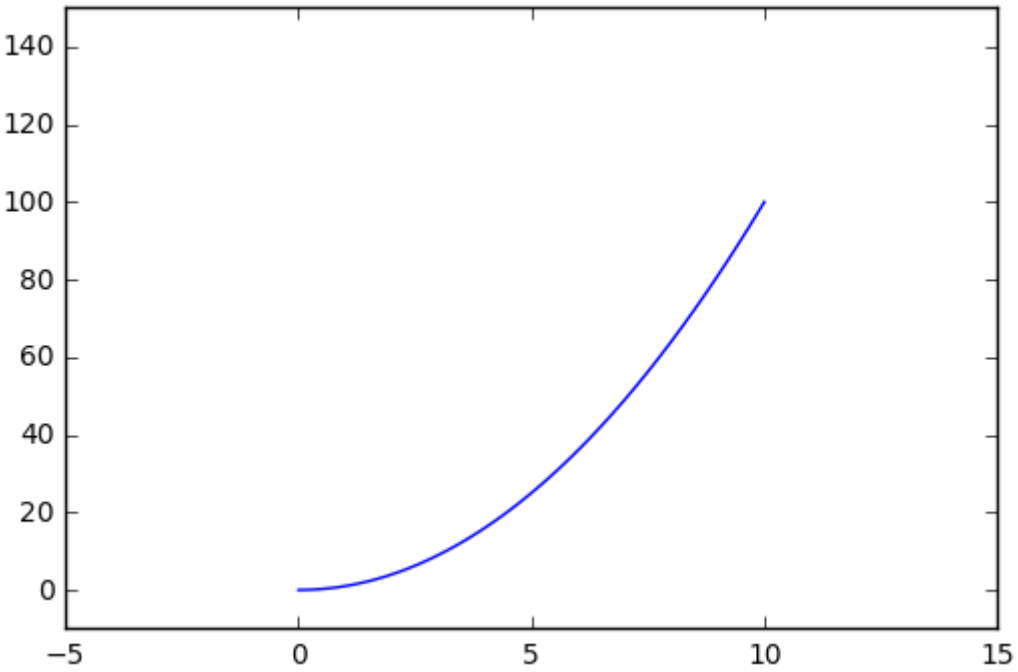
In [30]:
```
X = np.linspace(0, 10, 5) # 5 = resolution
Y = [f(x) for x in X]

fig, ax = plt.subplots()
ax.plot(X,Y)

plt.show()
```




Παρατηρήστε πόσο "σπαστή" είναι η γραφική παράσταση

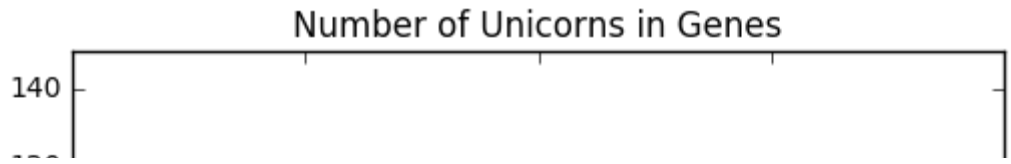
Με τη συνάρτηση ax.set_xlim, ax.set_ylim αλλάζουμε τα όρια των αξόνων

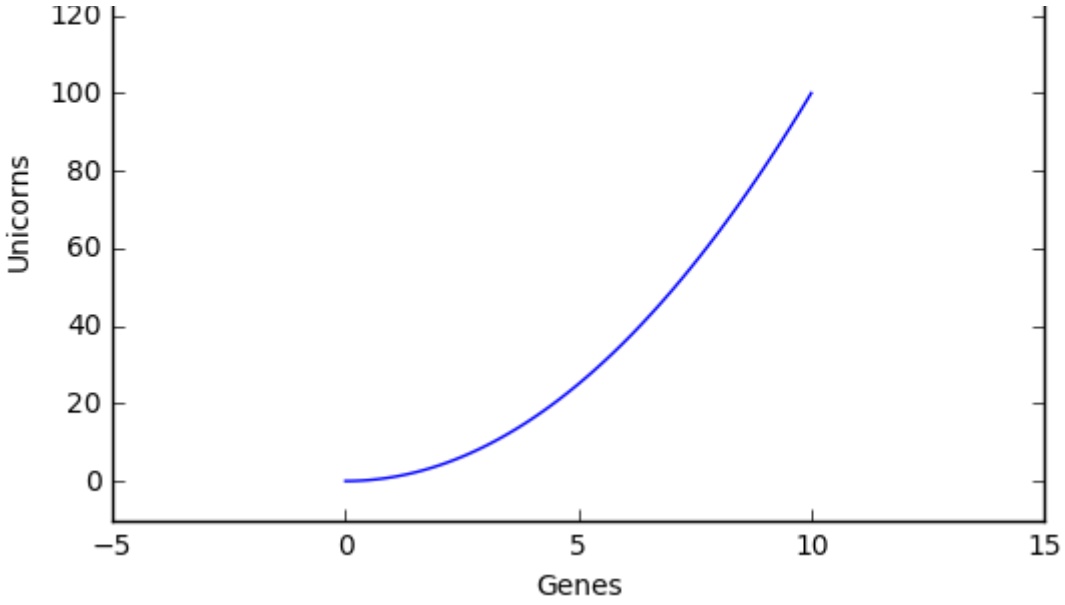
In [32]:
```
fig, ax = plt.subplots()

X = np.linspace(0, 10, 100) # 0 = min X, 10 = max X, 100 = resolution...
Y = [f(x) for x in X]

ax.set_xlim(-5, 15)
ax.set_ylim(-10, 150)
ax.plot(X,Y)

plt.show()
```




Μπορούμε επίσης να βάλουμε labels στους άξονες και σε όλο το πλοτ:

In [33]:
```
fig, ax = plt.subplots()

X = np.linspace(0, 10, 100) # 0 = min X, 10 = max X, 100 = resolution...
Y = [f(x) for x in X]

ax.set_xlim(-5, 15)
ax.set_ylim(-10, 150)
ax.plot(X,Y)

ax.set_xlabel("Genes")
ax.set_ylabel("Unicorns")
ax.set_title("Number of Unicorns in Genes")


plt.show()
```



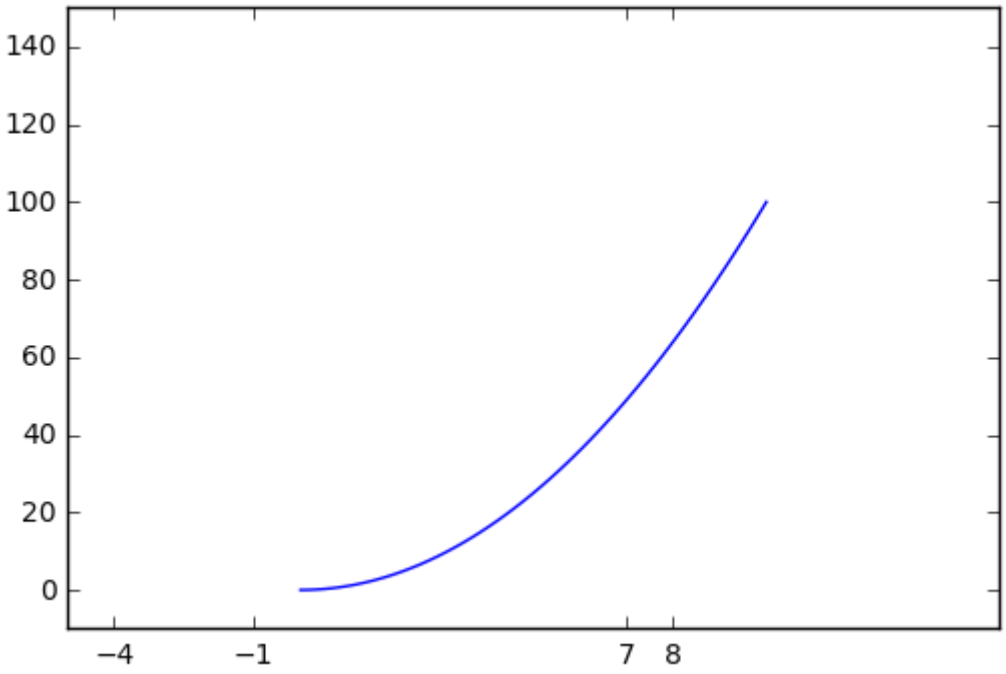



Μπορείτε να ορίσετε μέγεθος, γραμματοσειρά και στυλ στα labels

In [36]:
```
fig, ax = plt.subplots()

X = np.linspace(0, 10, 100) # 0 = min X, 10 = max X, 100 = resolution...
Y = [f(x) for x in X]

ax.set_xlim(-5, 15)
ax.set_ylim(-10, 150)
ax.plot(X,Y)

ax.set_xlabel("Genes")
ax.set_ylabel("Unicorns")
ax.set_title("Number of Unicorns in Genes", fontname="Comic Sans MS", font

plt.show()
```

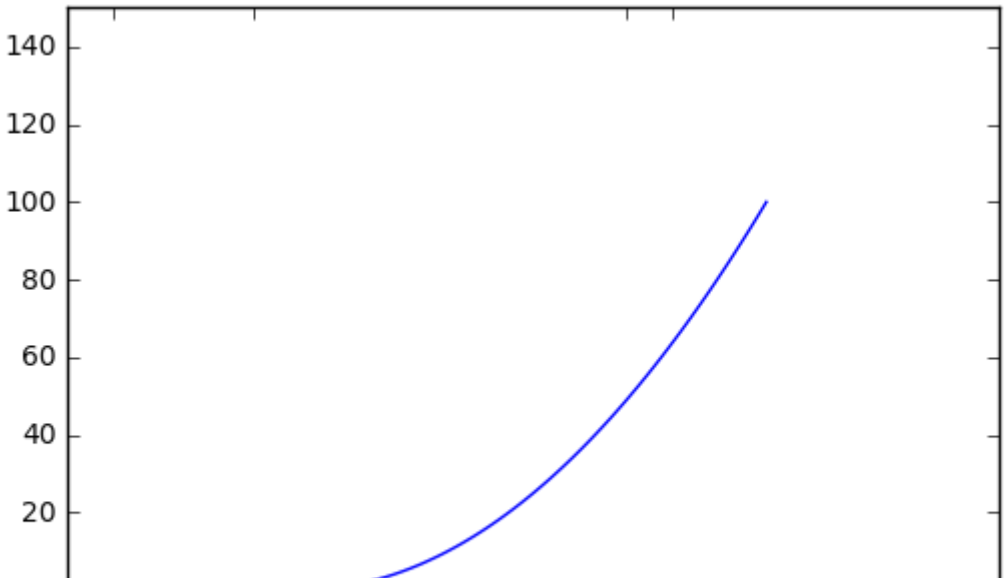
Με τη ax.set_xticks μπορείτε να ορίσεττε εσείς ποια ticks θα φαίνονται σε κάθε άξονα

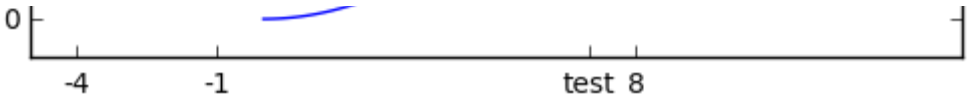
In [38]:
```
fig, ax = plt.subplots()
ax.set_xlim(-5, 15)
ax.set_ylim(-10, 150)
ax.plot(X,Y)
ax.set_xticks([-4,-1,7,8])
plt.show()
```




Μπορείτε να αλλάξετε και το label του tick:

In [45]:
```
fig, ax = plt.subplots()
ax.set_xlim(-5, 15)
ax.set_ylim(-10, 150)
ax.plot(X,Y)
ax.set_xticks([-4,-1,7,8])

ticks = ax.get_xticks().tolist()
ticks[2] = "test"
ax.set_xticklabels(ticks)

plt.show()
```

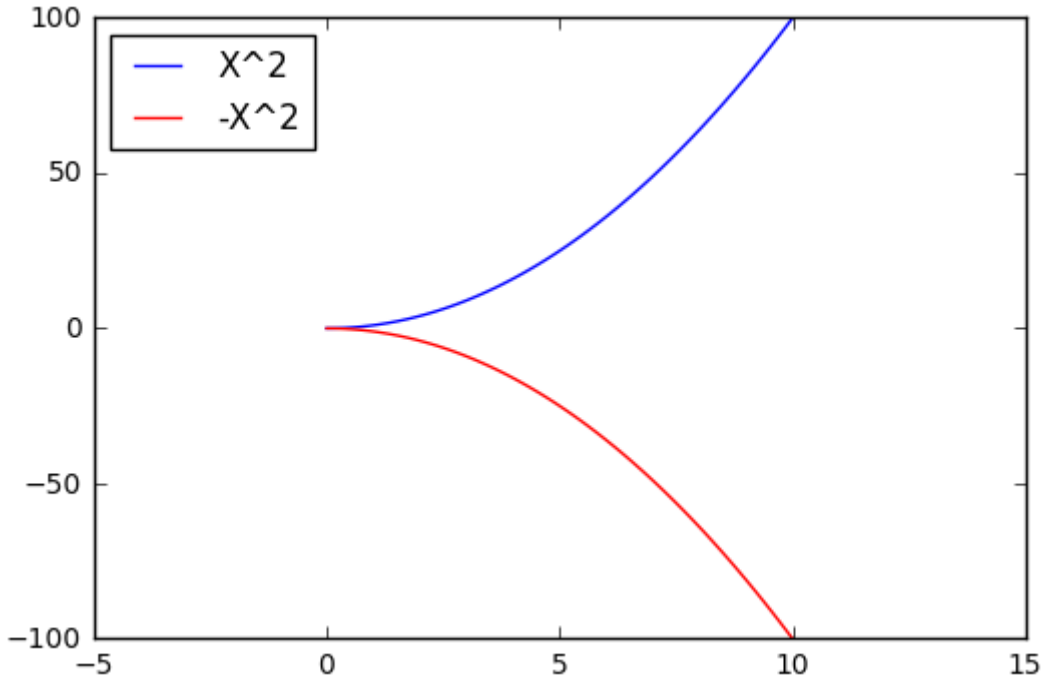





Η συνάρτηση plot επιστρέφει έναν πίνακα από legends. Αυτά τα legends μπορούμε αν θέλουμε να να τα προσθέσουμε στο plot

In [62]:
```
fig, ax = plt.subplots()
ax.set_xlim(-5, 15)
ax.set_ylim(-100, 100)
legends = ax.plot(X,Y, 'b', X, [-y for y in Y], 'r')
print (legends)
plt.legend(legends, ["X^2", "-X^2"], loc=2) # loc=2 shmainei panw aristera
plt.show()
```

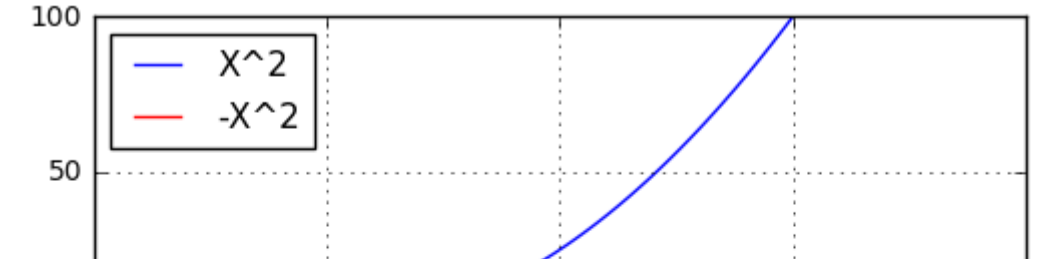
```
[<matplotlib.lines.Line2D object at 0x10e3c2208>, <matplotlib.lines.Line2D
object at 0x10df87d68>]
```

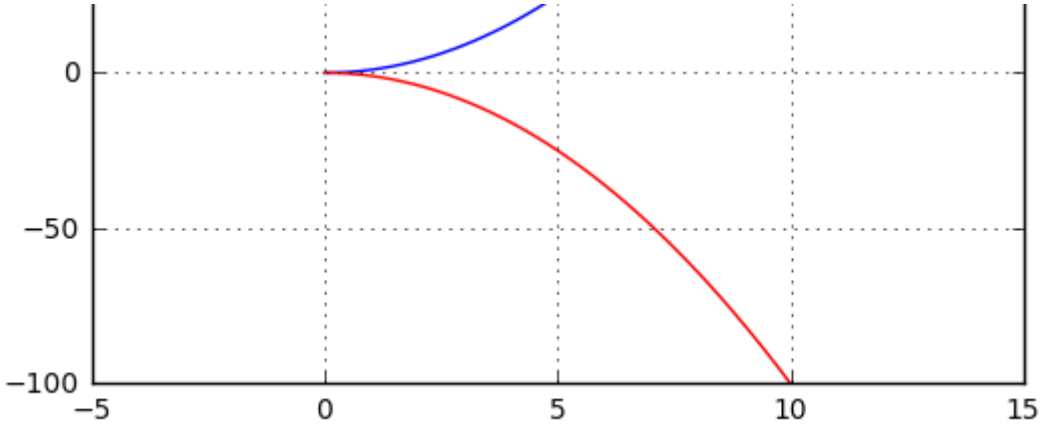



Περισσότερα για το loc (location του legend δείτε εδώ: http://matplotlib.org/api/legend_api.html#matplotlib.legend.Legend)

Επίσης μπορούμε να προσθέσουμε ενα grid:

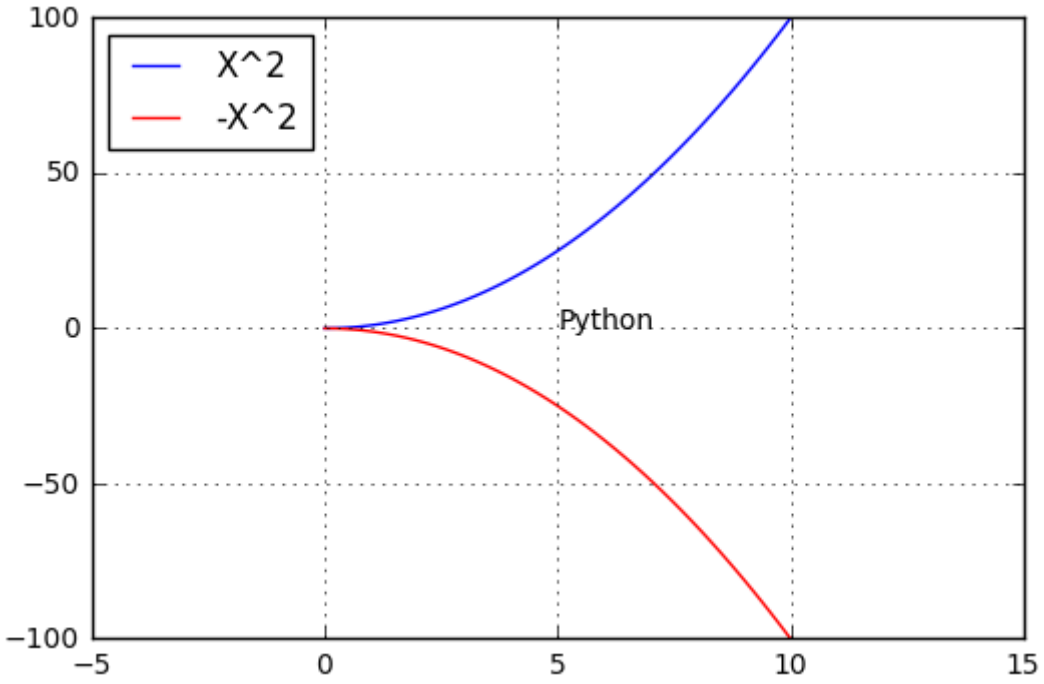
In [64]:
```
fig, ax = plt.subplots()
ax.set_xlim(-5, 15)
ax.set_ylim(-100, 100)
legends = ax.plot(X,Y, 'b', X, [-y for y in Y], 'r')

ax.grid(True)

plt.legend(legends, ["X^2", "-X^2"], loc=2) # loc=2 shmainei panw aristera
plt.show()
```






Προσθέστε ένα κείμενο στο σημείο X,Y :

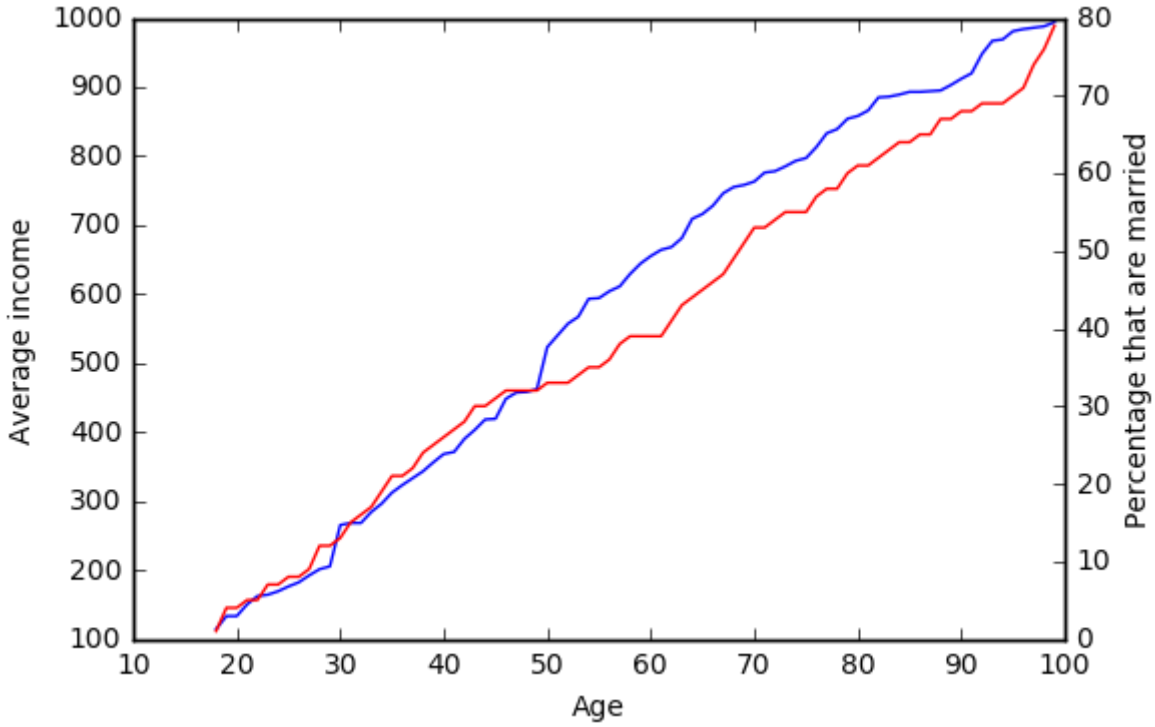
In [65]:
```
fig, ax = plt.subplots()
ax.set_xlim(-5, 15)
ax.set_ylim(-100, 100)
legends = ax.plot(X,Y, 'b', X, [-y for y in Y], 'r')
ax.grid(True)

ax.text(5,0,"Python")

plt.legend(legends, ["X^2", "-X^2"], loc=2) # loc=2 shmainei panw aristera
plt.show()
```




Επίσης υπάρχει η annotate με την οποία μπορείτε να βάλετε βελάκια

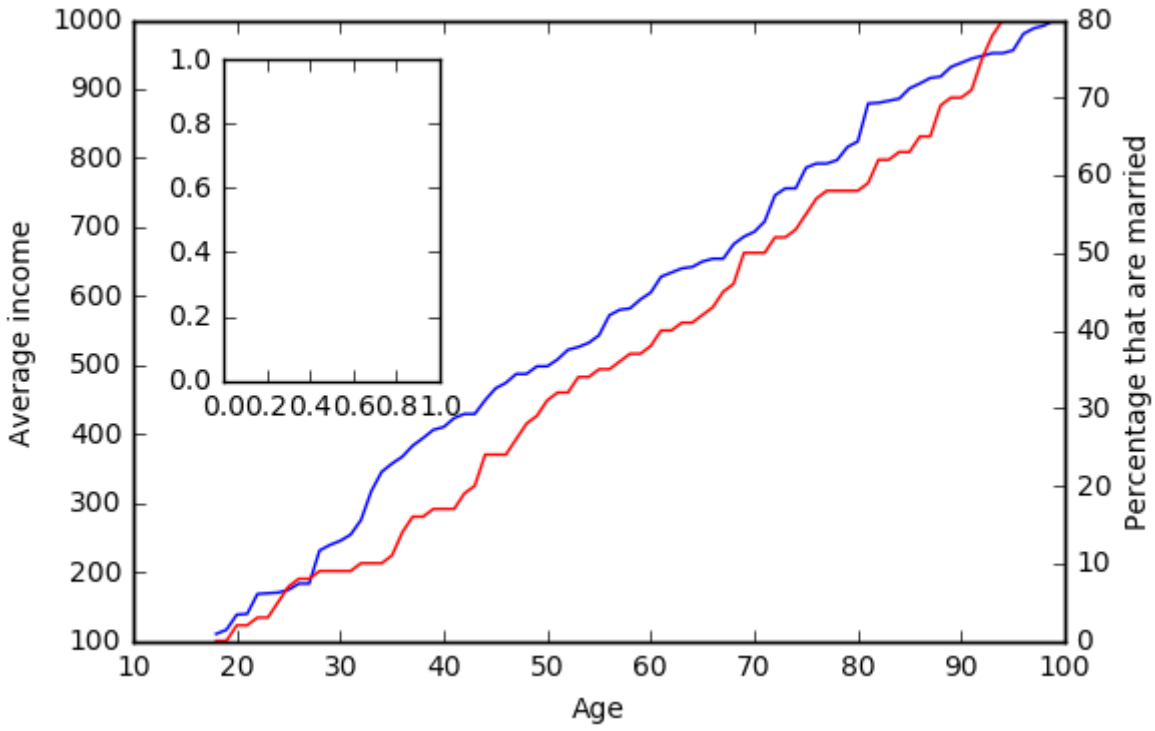
Πολλές φορές θέλουμε να τυπώσουμε δύο plots τα οποία να μοιράζονται τον ίδιο άξονα. Ας υποθέσουμε π.χ. ότι θέλουμενα μοιράζονται το άξονα X :

In [70]:
```
fig, ax = plt.subplots()
import random

age = [x for x in range(18,100)]
income = sorted([random.randint(100,1000) for x in age])
percentage_married = sorted([random.randint(0, 80) for x in age])

legends_income = ax.plot(age, income, 'b')

# Dhmiourgoume ena antigrafo tou axona X
ax_new = ax.twinx()

# Plotaroume to deutero plot panw se auto
legends_married = ax_new.plot(age, percentage_married, 'r')

ax.set_xlabel("Age")
ax.set_ylabel("Average income")
ax_new.set_ylabel("Percentage that are married")

plt.show()
```

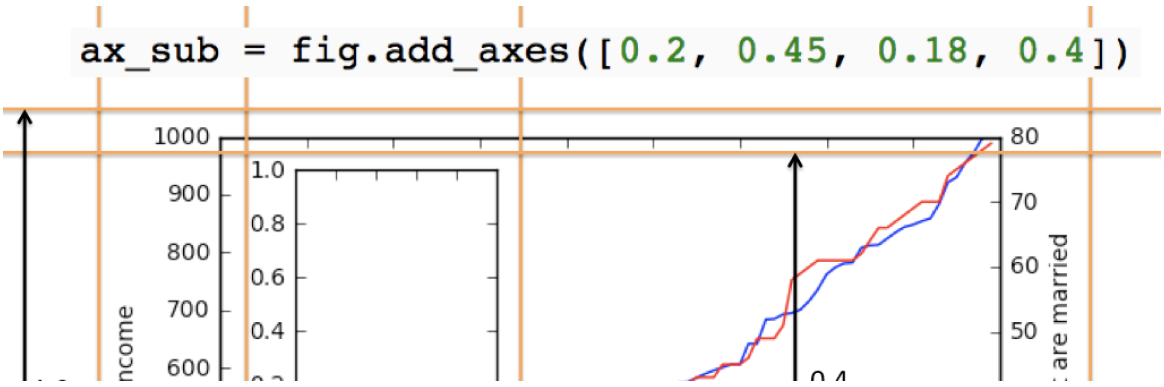


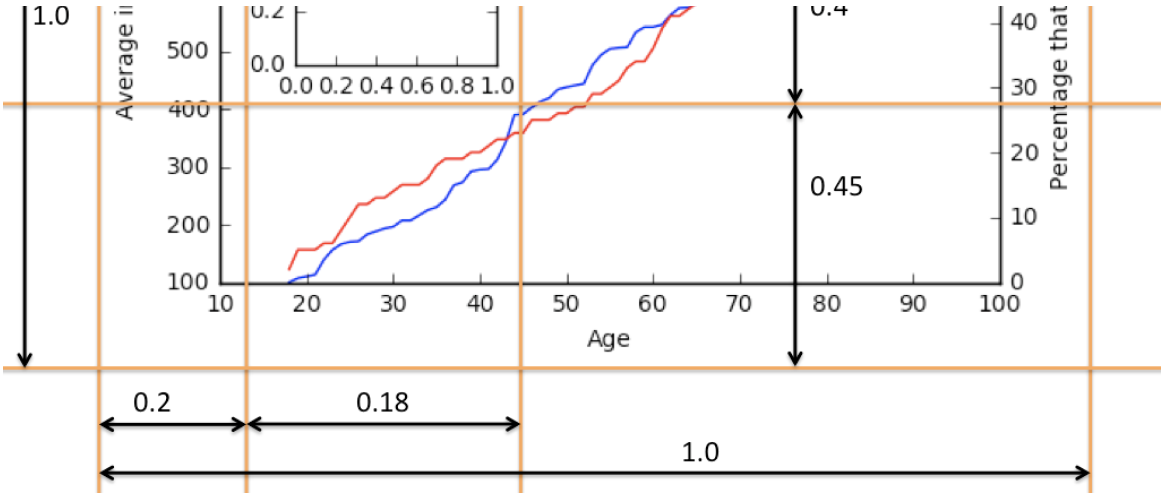

Επίσης μπορούμε να προσθέσουμε ένα ολόκληρο νέο πλότ μέσα σε ένα παλιό! Αυτό το κάνουμε με την εντολή fig.add_axes(). Η add_axes **ΑΓΝΟΕΙ** το μέγεθος των αξόνων (π.χ. ο Χ έχει μέγεθος από 10 μέχρι 100 παραπάνω). Αντίθετα θεωρεί ότι ΟΛΟ το πλοτ είναι ένα καρτεσιανό γινόμενο [0,1]Χ[0,1] . Με την add_axes ορίζουμε τις διαστάσεις του νέου plot πάνω στο παλιό. Παράδειγμα

In [82]:
```
fig, ax = plt.subplots()
import random

age = [x for x in range(18,100)]
income = sorted([random.randint(100,1000) for x in age])
percentage_married = sorted([random.randint(0, 80) for x in age])

legends_income = ax.plot(age, income, 'b')

# Dhmiourgoume ena antigrafo tou axona X
ax_new = ax.twinx()

# Plotaroume to deutero plot panw se auto
legends_married = ax_new.plot(age, percentage_married, 'r')

ax.set_xlabel("Age")
ax.set_ylabel("Average income")
ax_new.set_ylabel("Percentage that are married")


# Φτιάχνουμε νέο υπο-πλοτ:
ax_sub = fig.add_axes([0.2, 0.45, 0.18, 0.4])


plt.show()
```




Η σημασιολογία των παραμέτρων της `add_axes` φαίνεται στο παρακάτω σχήμα:

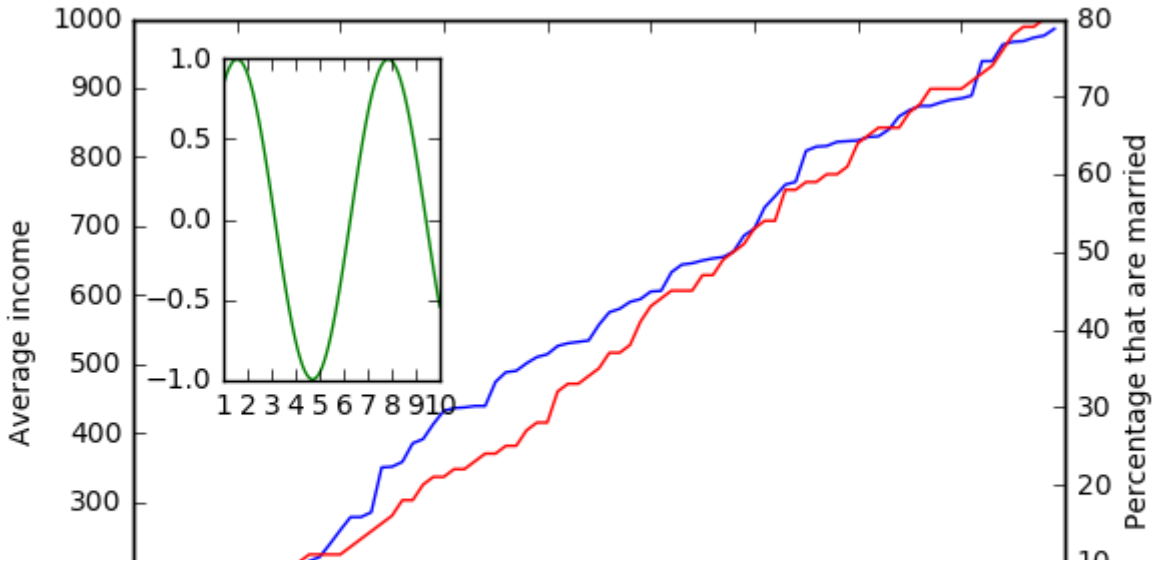


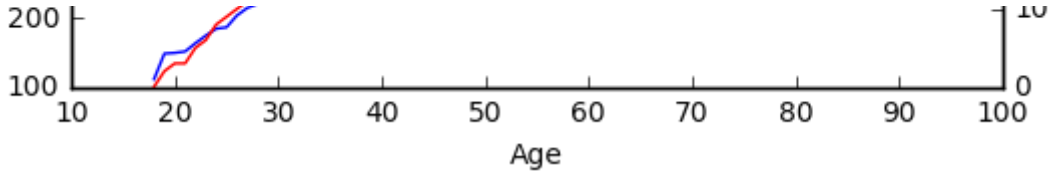



Ας πλοτάρουμε μέσα στο sub-plot:

In [85]:

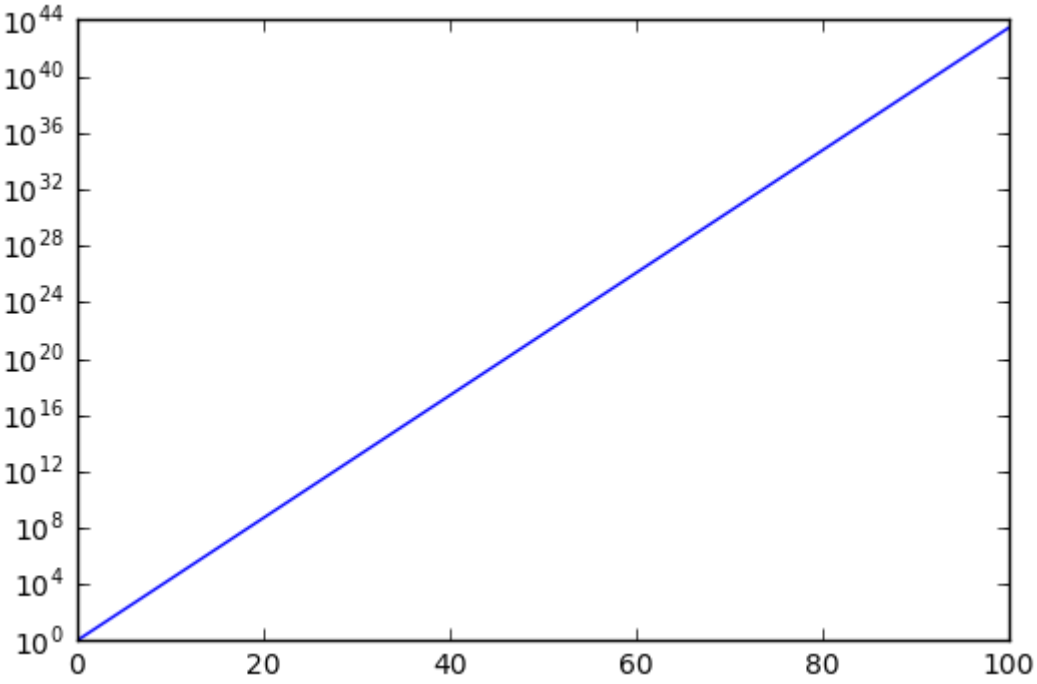
```
fig, ax = plt.subplots()
import random

age = [x for x in range(18,100)]
income = sorted([random.randint(100,1000) for x in age])
percentage_married = sorted([random.randint(0, 80) for x in age])

legends_income = ax.plot(age, income, 'b')

# Dhmiourgoume ena antigrafo tou axona X
ax_new = ax.twinx()

# Plotaroume to deutero plot panw se auto
legends_married = ax_new.plot(age, percentage_married, 'r')

ax.set_xlabel("Age")
ax.set_ylabel("Average income")
ax_new.set_ylabel("Percentage that are married")


# Φτιάχνουμε νέο υπο-πλοτ:
ax_sub = fig.add_axes([0.2, 0.45, 0.18, 0.4])
sub_X = np.linspace(1,10,100)
sub_Y = np.sin(sub_X)
ax_sub.plot(sub_X,sub_Y, 'g')

plt.show()
```






Μπορούμε επίης να αλλάξουμε τη κλίμακα των αξόνων σε λογαριθμική:

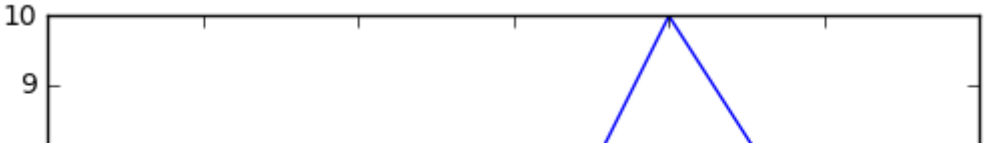
In [88]:
```
fig, ax = plt.subplots()

X = np.linspace(0,100,1000)
Y = np.exp(X)

ax.plot(X,Y)

ax.set_yscale("log")
plt.show()
```

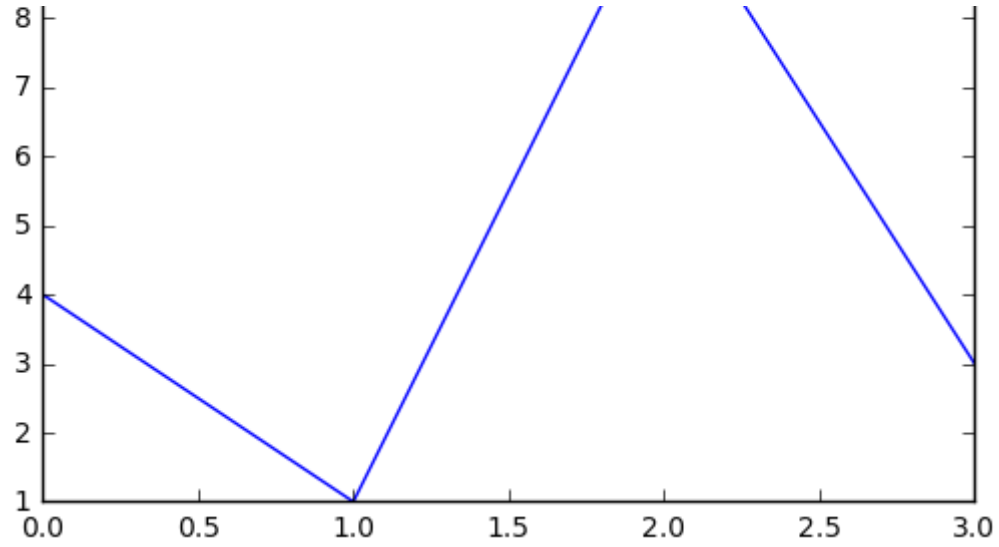



Μπορούμε να σώσουμε στο δίσκο ένα πλοτ μέσω της "plt.savefig()". Ανάλογα με την κατάληξη που θα βάλετε στο όνομα του αρχείου, θα το σώσει και σε διαφορετικό format. **ΠΡΟΣΟΧΗ!** τη savefig πρέπει να τη καλείτε ΠΡΙΝ τη plt.show()

In [87]:
```
fig, ax = plt.subplots()

ax.plot([4,1,10,3])

plt.savefig("figure.png")
plt.savefig("figure.jpg")
plt.savefig("figure.eps")
plt.savefig("figure.tiff")
plt.savefig("figure.pdf")

plt.show()
```

```
<matplotlib.figure.Figure at 0x10e54dd68>
```



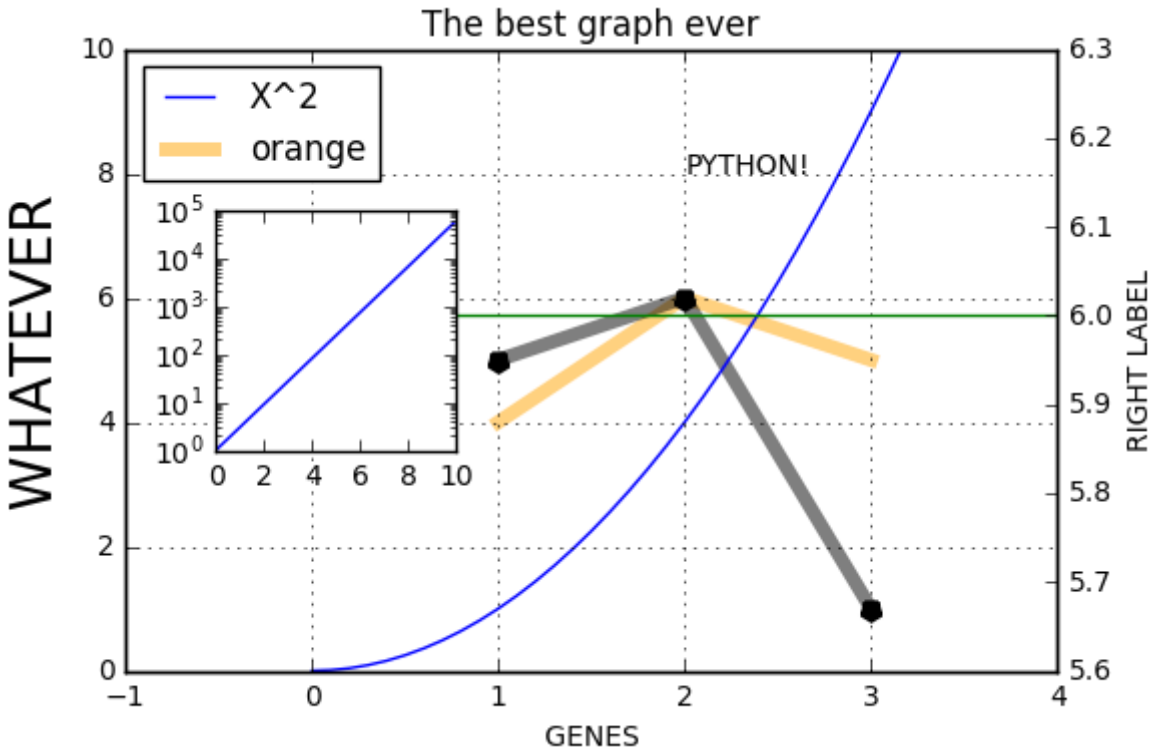



Και το παράδειγμα που δουλέψαμε στη διάλεξη:

In [89]:

```
def x2(x):
    return x*x

fig, ax = plt.subplots()
leg_orange, = ax.plot([1,2,3], [4,6,5,], '-', color="orange", lw="5", alpha
ax.plot([1,2,3], [5,6,1,], '-', color="black", lw="5", alpha=0.5)
ax.plot([1,2,3], [5,6,1,], '*', color="black", lw="5", mew=5)
X = [x/10.0 for x in range(0,100)]
Y = [x2(x) for x in X]
leg_x2, = ax.plot(X, Y)

ticks = ax.get_xticks()
print (ticks)
#ticks = [str(x) + " a " for x in ticks]
# ax.set_xticklabels(ticks)
#ax.set_xticks([x for x in range(-1,5, 2)])
ax.set_xlabel(" GENES ")
ax.set_ylabel(" WHATEVER", fontsize=20)
ax.set_title("The best graph ever")
ax.grid(True)

ax_new = ax.twinx()
ax_new.plot([0,4], [6,6], color="green")
ax_new.set_ylabel("RIGHT LABEL")

ax.set_xlim(-1,4)
ax.set_ylim(0,10)

ax.text(2, 8, "PYTHON!")

plt.legend([leg_x2, leg_orange], ["X^2", "orange"], loc=2)
ax2 = fig.add_axes([0.2, 0.4, 0.2, 0.3])
X2 = [x/10.0 for x in range(0,100)]
Y2 = [3**x for x in X2]
ax2.set_yscale("log")

ax2.plot(X2, Y2)

plt.savefig("figure.png")
plt.savefig("figure.jpg")
plt.savefig("figure.eps")
plt.savefig("figure.tiff")
plt.show()
```

```
[  0.   2.   4.   6.   8.  10.]
```

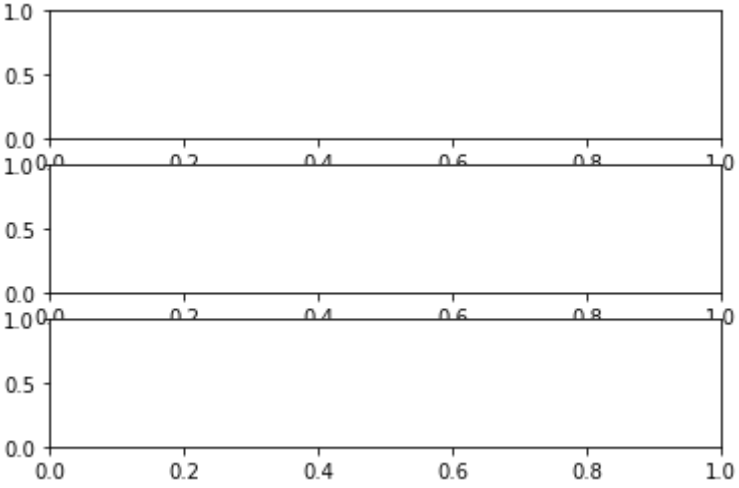



## subplots

Μπορούμε να φτιάξουμε πολλά plots τα οποία να είναι στοιχισμένα σε ένα grid. Αυτό γίνεται αν στη subplots δώσουμε έναν ή δύο αριθμούς από subplots:

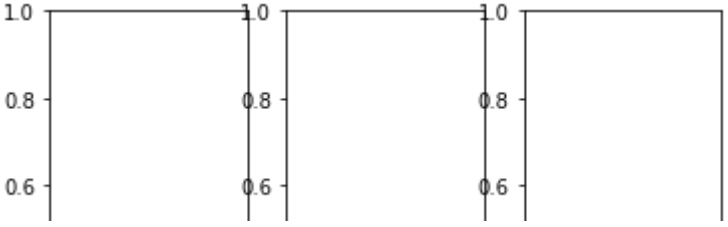
3 subplots κάθετα:

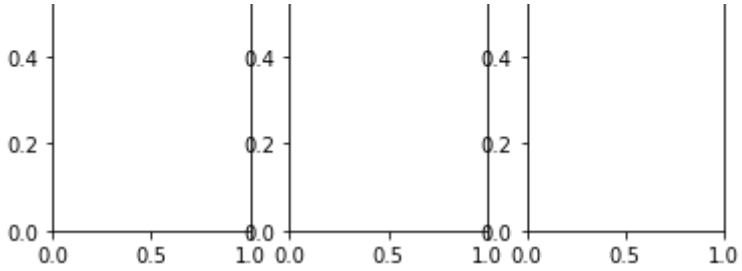
In [3]:
```
fig, ax = plt.subplots(3)
```



3 subplots οριζόντια:

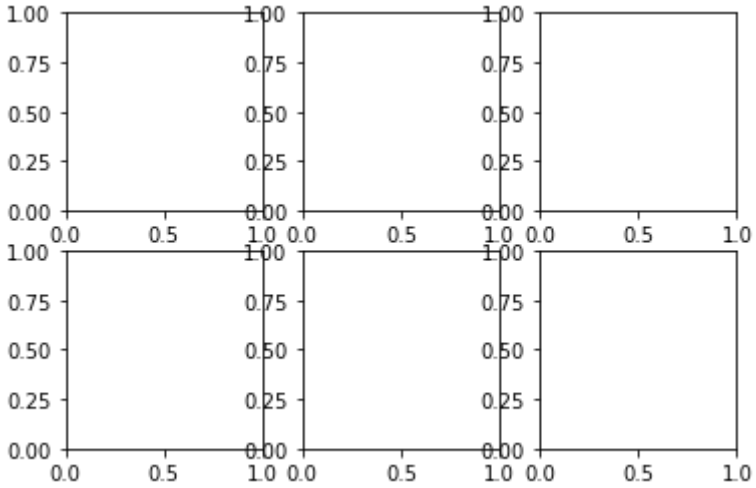
In [4]:
```
fig, ax = plt.subplots(1,3)
```





Ένα grid από plots 2X3:

In [5]:
```
fig, ax = plt.subplots(2,3)
```



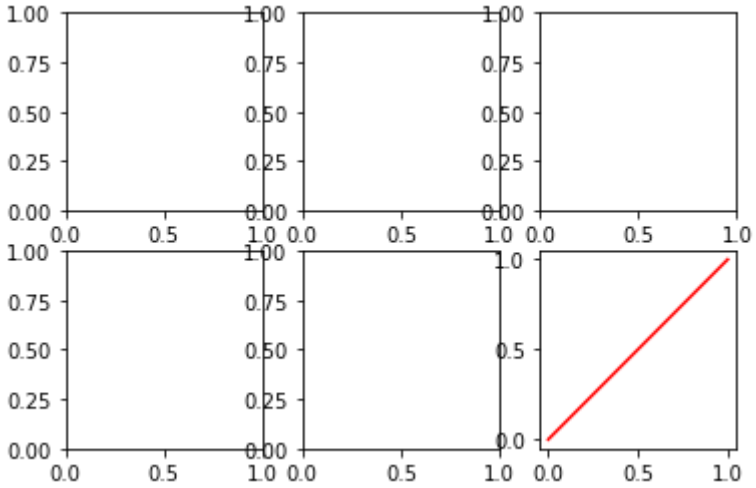
Παρατηρούμε ότι το `ax` είναι ένας πίνακας 2X3:

In [6]:
```
ax.shape
```

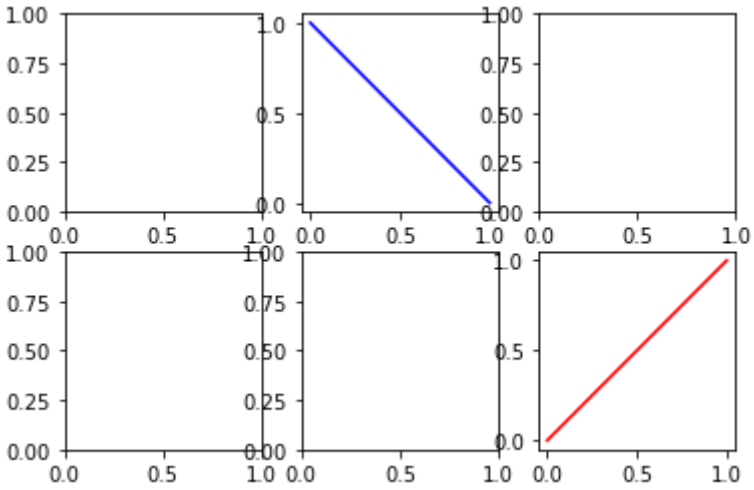
Out[6]:
```
(2, 3)
```

Άρα μπορούμε να αναφερθούμε σε οποιοδήποτε στοιχείο αυτού του πίνακα για να κάνουμε "γεμίσουμε" κάποια από αυτά τα sub-plots:

In [8]:
```
fig, ax = plt.subplots(2,3)
ax[1][2].plot([0,1], [0,1], 'r') # Το κάτω δεξιά
plt.show()
```



In [9]:
```
fig, ax = plt.subplots(2,3)
ax[1][2].plot([0,1], [0,1], 'r') # Το κάτω δεξιά
ax[0][1].plot([0,1], [1,0], 'b') # Το πάνω μεσαίο
plt.show()
```



## Κάποιες σημειώσεις:

- Η matplotlib είναι τεράστια! Ελέγξτε πόσο διαφορετικά plots μπορείτε να φτιάξετε: http://matplotlib.org/gallery.html .
- Υπάρχουν βιβλιοθήκες για plot-άρισμα συγκεκριμμένου τύπου δεδομένων οι οποίες βασίζονται στη matplotlib. Μία από τις καλύτερες (IMHO) είναι η seaborn η οποία προσανατολίζεται σε πλοτάρισμα κατανομών.
- Η matplotlib έχει "κατηγορηθεί" ότι είναι καλή μεν αλλά δεν παράγει publication ready plots. Δηλαδή αισθητικά θέλουν λίγο δουλειά ακόμα πριν μπουν σε ένα paper. Διαβάστε σχετικά: https://github.com/jbmouret/matplotlib_for_papers , http://blog.dmcdougall.co.uk/publication-ready-the-first-time-beautiful-reproducible-plots-with-matplotlib/ , http://www.reddit.com/r/bioinformatics/comments/2cnv9g/polishing_your_python_matplotlib_plots_for/ , http://nipunbatra.github.io/2014/08/latexify/

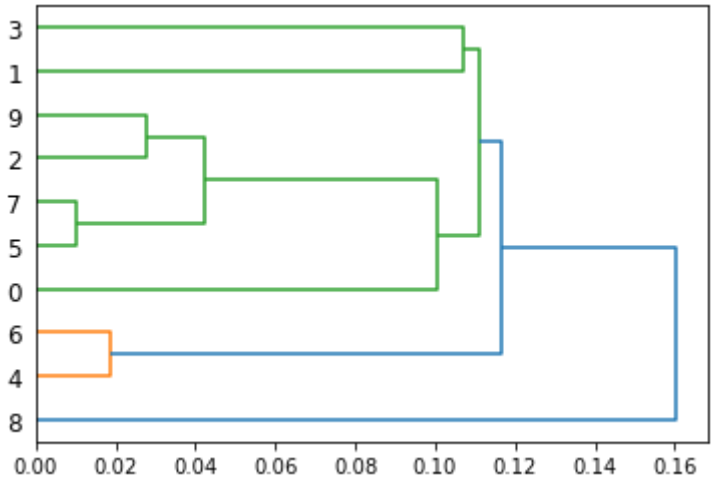
## Dendrogram

Ένα άλλο είδος plots ιδιαίτερα χρησιμο στη φυλογενετική είναι τα δεδρογράμματα. Για παράδειγμα ας υποθέσουμε ότι έχουμε 10 αντικείμενα και υπολογίζουμε όλες τις αποστάσεις ανά δύο (pairwise distances):

In [10]:
```
import random

o = [random.random() for x in range(10)]

def distance(a,b): # Μία απλή συνάρτηση απόστασης: η απόλυτη διαφορά
    return abs(a-b)
```

In [42]:
```
# Υπολογίζουμε τις αποστάσεις μεταξύ όλων των ζευγαριών

from itertools import combinations
all_distances = [distance(x,y) for x,y in combinations(o,2)]
```

In [43]:
```
# Φτιάχνουμε το δεδρόγραμμα:

from scipy.cluster.hierarchy import dendrogram, linkage

l = linkage(all_distances)
dn = dendrogram(l, orientation='right')
```




In [46]:
```
# Μπορούμε να επιβεβαιώσουμε ότι το δενδρόγραμμα είναι σωστό βλέποντας τις
# Π.χ. οι κόμβοι 7 και 5 είναι το πιο κοντινό ζευγάρι
for i,x in enumerate(o):
    print (i,x)
```

```
0 0.535941403079773
1 0.4246640716083616
2 0.6639266980035059
3 0.3177651541741866
4 0.2011939407247051
5 0.7161232515336856
6 0.18269991970324606
7 0.7061565998207685
8 0.022555869599750866
9 0.6362676402564162
```

# plotly

Η matplotlib ανοίκει στη κατηγορία των μη-interactive plots. Δλδ δεν μπορείς να αλληλεπιδράσεις με το plot (αν και στη τελευταία έκδοση έχουν προσθέσει κάποιες βασικές δυνατότητες). Φτιάχνει plots για παρουσιάσεις / papers και όχι για visual exploration! Στη 2η κατηγορία ανοίκουν άλλες βιβλιοθήκες όπως η bokeh και η plot.ly.

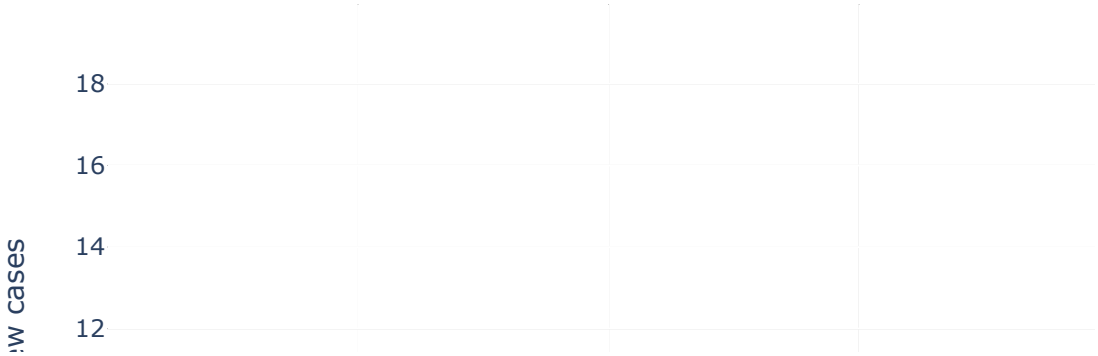
Η plotly χρειάζεται εγκατάστηση:

```
!pip install plotly
```

Ας δούμε μερικά παραδείγματα με τη plotly:

Κλασσικό ploy:

In [11]:
```
import plotly.express as px

fig=px.line(
    x=range(1,11),
    y=[2,4,6,7,8,9,10,15,18,19],
    labels={'x':'Day','y':'Number of new cases'})
fig.update_layout({'plot_bgcolor': '#d7dade','paper_bgcolor': '#d7dade'})
fig.show()
```



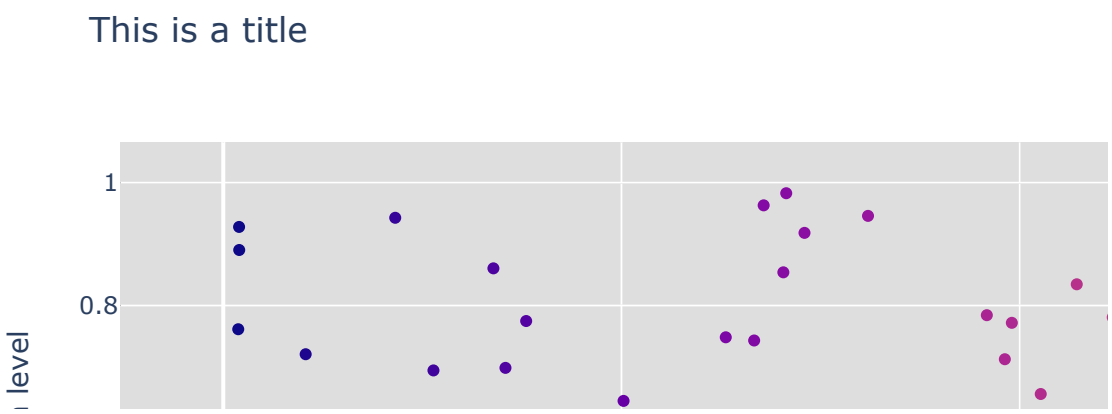

Το plot αυτό είναι διαδραστικό (interactive). Μπορούμε να το σώσουμε σαν html:

In [3]:
```
fig.write_html("file.html")
```

Scatter plot με plotly:

In [12]:

```
import numpy as np

X = np.random.random(100)
Y = np.random.random(100)

fig=px.scatter(
    x=X,
    y=Y,
    labels={'x':"3'UTR length", 'y':'mean expression level'},
    color=X,
    title='This is a title',
    #log_x=True,
)

fig.update_layout({'plot_bgcolor': '#dedede'})
fig.show()
```




Ένα παράδειγμα με "κυκλικά plots". Το χλωροπλαστικό DNA της Arabidopsis thaliana. Για να κατεβάσετε τα δεδομένα για αυτό το plot: πάμε εδώ: ftp://ftp.ensemblgenomes.org/pub/plants/release-49/gff3/arabidopsis_thaliana και κατεβάζετε το αρχείο Arabidopsis_thaliana.TAIR10.49.chromosome.Pt.gff3.gz:

In [15]:

```
# Credit: Τζωρτζίνα Νούλη
# Εργασία πάνω στο μάθημα: προγραμματισμός με τη γλώσσα python
# Μεταπτυχιακό πρόγραμμα βιοπληροφορικής, 2020-2021
# Ιατρική σχολή Πανεπιστημίου Κρήτης.

import plotly.graph_objects as go
import numpy as np
from matplotlib import colors as mcolors
import re
from matplotlib import cm

def colors(rgb=False):
    for mp in ['Accent','tab10_r','Pastel1','Set3',]: #other options 'Paste
        for x in cm.get_cmap(mp).colors:
            yield 'rgb'+str(x) if rgb else x

with open('Arabidopsis_thaliana.TAIR10.49.chromosome.Pt.gff3','r') as f:
    s=f.read().split('\n')
    s=[i for i in s if i and i[0]!='#']
    l=s[0].split()[4]
    length=int(l)/360
    s=[(int(i.split()[3])/length,int(i.split()[4])/length,i.split()[6],re.
    groups=set(i[3][:3].lower() for i in s)
    s={k:[v for v in s if v[3][:3].lower()==k] for k in groups }



color=colors(True)
fig = go.Figure()

for a,w,h in [(1.06,1,'Strand 2'),(0.94,1,'Strand 1')]:
    fig.add_trace(go.Scatterpolar(r=[a] * 20*len(s),theta=np.linspace(1,360
                                  line_color='black', line_width=w,showlege

for a,th,txt in zip([0.8] * (int(l)//1000),np.linspace(0,360,int(l)//1000),
    if not txt%5:
        fig.add_trace(go.Scatterpolar(r=[a],theta=[th],textfont={'family':'

for group in s:
    c=next(color)
    for i in s[group]:
        if i[3]!='Unnamed' and i[4]!='exon':
            fig.add_trace(go.Scatterpolar(r=[1.06*(i[2]=='-')+0.94*(i[2]==

        ### Alternative version of plot with coding regions for mRNA,tRNA,

        #if i[3]=='Unnamed' and i[4]!='CDS':
        #    col={'mRNA':'Burlywood','tRNA':'olive','rRNA':'tomato'}
        #    fig.add_trace(go.Scatterpolar(r=[1.06*(i[2]=='-')+0.94*(i[2]==


fig.update_layout(
  polar=dict(angularaxis=dict(rotation=90,direction="counterclockwise",show

fig.show()
```

# Seaborn

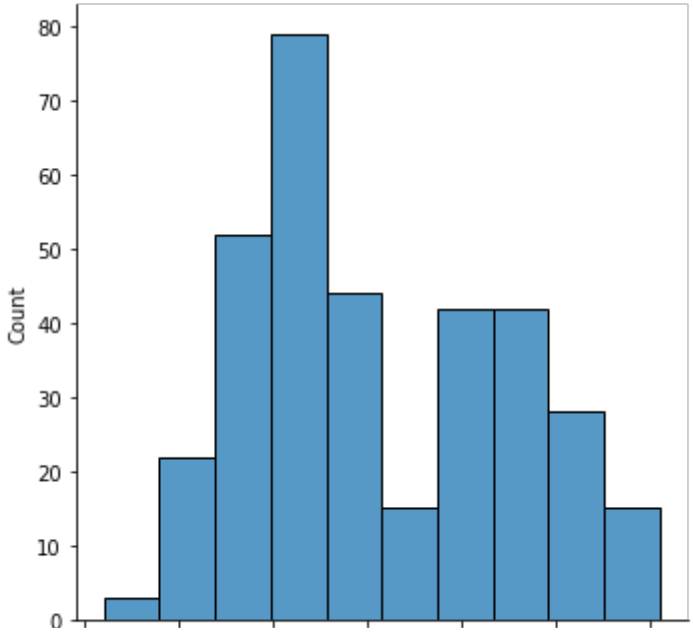
Η seaborn είναι μια βιβλιοθήκη για "statistical graphics". Δηλαδή παρέχει μεθόδους για πλοτάρισμα κατανομών, ιστογραμμάτων, ομαδοποίησης (clustering) κτλ.

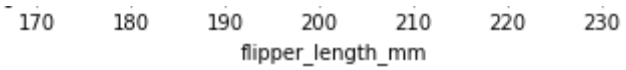
Ένα παράδειγμα από εδώ: https://seaborn.pydata.org/generated/seaborn.displot.html

In [16]:
```
import seaborn as sns
```

In [18]:
```
penguins = sns.load_dataset("penguins")
sns.displot(data=penguins, x="flipper_length_mm")
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x7ffb595c7f40>
```

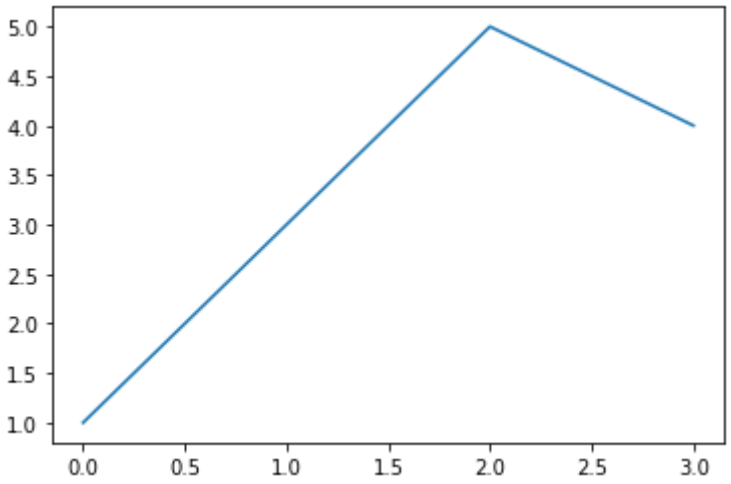





Η seaborn επιτρέπει επίσης την "ωραιοποίηση" των πλοτς της matplotlib. Ας δούμε ένα παράδειγμα:

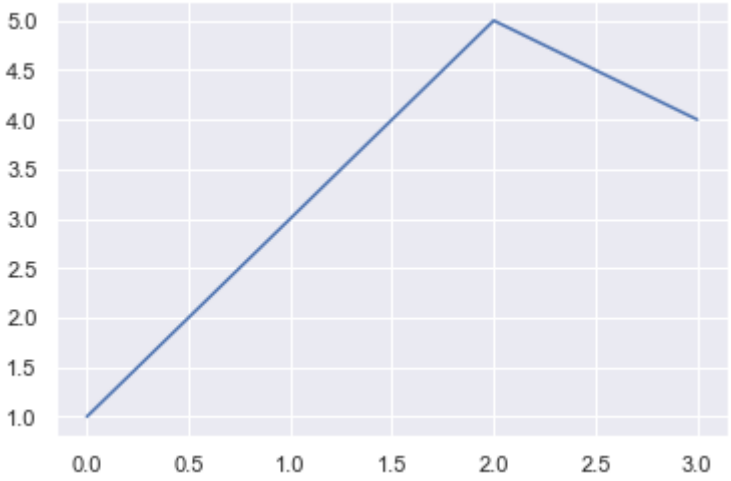
Χωρίς seaborn:

In [2]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([0,1,2,3], [1,3,5,4], '-')
plt.show()
```




Με seaborn:

In [3]:
```
import seaborn as sns
sns.set_theme()

fig, ax = plt.subplots()
ax.plot([0,1,2,3], [1,3,5,4], '-')
plt.show()
```



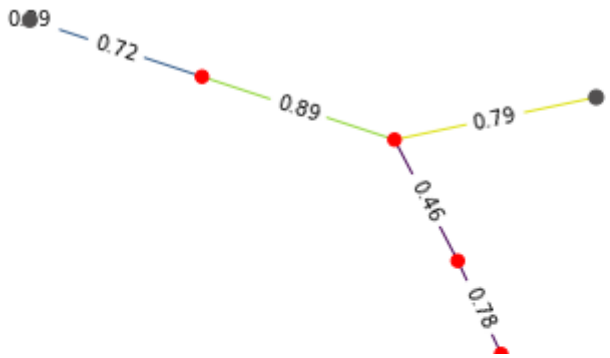

Διαβάζουμε εδώ: http://seaborn.pydata.org/tutorial/aesthetics.html πως μπορούμε να επιλέξουμε διαφορετικά "στυλ" για το πλοτ μας από τη seaborn: http://seaborn.pydata.org/tutorial/aesthetics.html

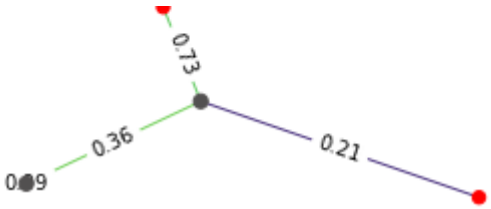
## Networkx

Η networkx είναι μία βιβλιοθήκη για την επεξεργασία αλλά και την οπτικοποίηση γράφων. Αφού την εγκαταστήσουμε (pip install...) μπορούμε να φτιάξουμε γράφους ως εξής:

In [37]:
```
import random

import networkx as nx

G = nx.Graph()

labels = []

# 10 τυχαίες ακμές
for x in range(10):
    n1 = random.randint(1,10) # Ξεκινάμε από έναν τυχαίο κόμβο
    n2 = random.randint(1,10) # Καταλίγουμε σε έναν τυχαίο κόμβο
    c = random.random() # Τυχαίο χρώμα
    G.add_edge(n1, n2,
               color=c,
               weight= '{0:.2f}'.format(random.random()), # Τυχαίο βάρος σ
              )
    labels.append('d')

pos = nx.spring_layout(G) # Υπολογίζουμε τις θέσεις στον 2διάστατο χόρο τω

node_size = {x:x/2 for x in range(1,11)} # Υπολογίζουμε (αυθαίρετα) το μέγ

# Τα κάνουμε plot
nx.draw(
    G,
    pos,
    with_labels=False,
    node_size=[50 if i%2==0 else 40 for i in G.nodes()],
    edge_color=nx.get_edge_attributes(G,'color').values(),
    node_color=['#525050' if i%2==0 else '#FF0000' for i in G.nodes()],
)
labels = nx.get_edge_attributes(G,'weight')  # Βάζουμε τα βάρη πάνω στις α
nx.draw_networkx_edge_labels(G,pos,edge_labels=labels) # πλοτάρουμε τα βάρ

#plt.figure(3,figsize=(15,10))
```

Out[37]:
```
{(9, 6): Text(-0.3246399734880249, 0.8393607649129553, '0.72'),
 (9, 1): Text(-0.13822768754437625, 0.6171603138000668, '0.89'),
 (6, 6): Text(-0.41265536497435334, 0.9446971915222568, '0.39'),
 (1, 3): Text(-0.00752602109411178, 0.2757609811871292, '0.46'),
 (1, 8): Text(0.06323129245690765, 0.5789424219216375, '0.79'),
 (3, 7): Text(0.04699044479343656, -0.12170415825792096, '0.78'),
 (4, 2): Text(0.018412541670912178, -0.796599028576672, '0.36'),
 (4, 5): Text(0.2498097374056697, -0.8219724442684326, '0.21'),
 (4, 7): Text(0.08840511218292141, -0.469289439065243, '0.73'),
 (2, 2): Text(-0.0707830023359778, -0.9492531686164785, '0.99')}
```






# GeoPandas

Η GeoPandas είναι μία επέκταση της pandas η οποία επιτρέπει την οπτικοποίηση γεωγραφικής πληροφορίας.

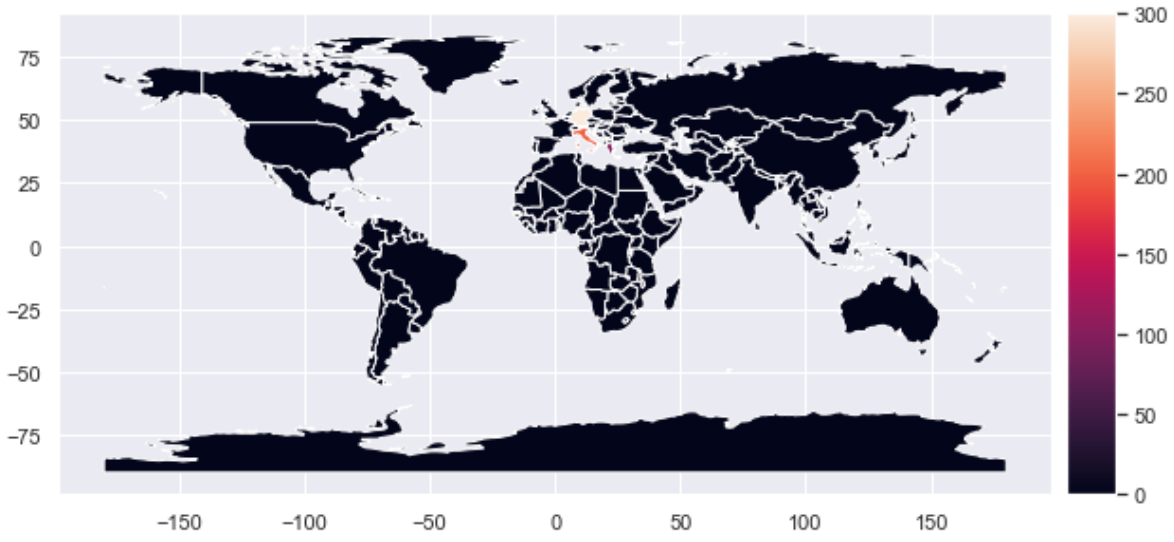
Δίνουμε ένα παράδειγμα:

In [4]:
```
import geopandas
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowr
```

In [8]:
```
def new_function(row):
    name = row['name']
    if name == 'Greece':
        return 100
    if name == 'Italy':
        return 200
    if name == 'Germany':
        return 300

    return 0

world['new_column'] = world.apply(new_function, axis=1)
```

In [9]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

fig, ax = plt.subplots(1, 1, figsize=(10, 8))
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.1)
world.plot(column='new_column', ax=ax, legend=True, cax=cax)
```

Out[9]: <AxesSubplot:>




# plotly GEO interactive

Ομοίως μπορούμε να φτιάξουμε διαδραστικά πλοτς με γεωγραφική πληροφορία μέσω της plotly:

Για το ακόλουθο παράδειγμα χρειαζόμαστε το αρχείο: https://www.dropbox.com/s/brqaopz8g2vs0ox/covid_fasta.gz?dl=1

In [11]:
```
# Credit: Τζωρτζίνα Νούλη
# Εργασία πάνω στο μάθημα: προγραμματισμός με τη γλώσσα python
# Μεταπτυχιακό πρόγραμμα βιοπληροφορικής, 2020-2021
# Ιατρική σχολή Πανεπιστημίου Κρήτης.

import re
import gzip
import numpy as np
from collections import defaultdict
import geopandas
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib import cm
from matplotlib.colors import ListedColormap, LinearSegmentedColormap
import plotly.express as px

def plot_geo(file,interactive=False):
    with gzip.open(file, 'rt') as f:

        d=defaultdict(int)
        for i in f:
            s=re.findall('>hCoV-19/(.+?)/.+',i)
            if s:
                d[s[0]]+=1
    world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_
    world['Covid_Count'] = world.apply(lambda row:d[row['name']], axis=1)
    if not interactive:
        fig, ax = plt.subplots(1, 1, figsize=(14, 12))
        divider = make_axes_locatable(ax)
        cax = divider.append_axes("right", size="4%", pad=0.2)
        ax=world.plot(column='Covid_Count',ax=ax,legend_kwds={'label': 'Nu
        world.boundary.plot(figsize=(14,12),edgecolor='darkgrey',ax=ax)
    else:
        fig = px.choropleth_mapbox(world, geojson=world.geometry, location
                           color_continuous_scale="inferno_r",hover_name='
                           mapbox_style="carto-positron",
                           zoom=3, center = {"lat": 37.0902, "lon": -95.71
                           opacity=0.5)

        fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
        fig.show()


file = 'covid_fasta.gz'
#plot_geo(file)

plot_geo(file,interactive=True)
```

In [ ]: