Μια βελτιωμένη μέθοδος για εκτύπωση strings

Τρόποι για Formatted Εκτύπωση

Μέχρι τώρα έχουμε χρησιμοποιήσει δύο είδη formatted εκτύπωσης:

```
% formatting:        Python v2
>>>> name = 'Fotis'
>>>> "Hello %s"%name
     "Hello Fotis"
```

```
str.format()         Python v3
>>>> "Hello! my name is {} and I am {} years old".format('George',20)
     Hello! My name is George and I am 20 years old
```

Η 2η επιλογή μας δίνει τη δυνατότητα να περάσουμε αναφερθούμε στις μεταβλητές δίνοντας μέσα στα {} τη θέση τους στη λίστα

```
>>>> "Hello! my name is {1} and I am {0} years old".format(20, 'George')
     Hello! My name is George and I am 20 years old
```

Θα μπορούσαμε να περάσουμε στη {} κάποιο όνομα μεταβλητής και να έχουμε δυνατότητα να περάσουμε αντικείμενα και να έχουμε πρόσβαση σε παραμέτρους

```
>>>>person ={'name':'George', 'age':20}
>>>>"Hello! my name is {name} and I am {age} years old".
     format(name=person['name'], age=person['age'])
     Hello! My name is George and I am 20 years old
```

```
>>>>person ={'name':'George', 'age':20}
>>>>"Hello! my name is {name} and I am {age} years old".format(**person)
```

Η επιλογή *f string για formatted*

f" {expr}...{expr}" Python v3.6

Περιέχει το γράμμα *f* ακολουθούμενο από " " και μέσα στα { } περιέχει εκφράσεις που θα αντικατασταθούν κατάλληλα

```
>>>>name='George'
>>>>age = 20
>>>>f" My name is {name} and I am {age} years old"
    My name is George and I am 20 years old
```

```
>>>>f" It takes {2*40} sec to get there" Replace the value 2*40
    It takes 80 sec to get there
```

Θα μπορούσαμε να περάσουμε **κάποια συνάρτηση** ή μια μέθοδο απευθείας:

```
>>>>def to_low(input):
.....  return input.lower()
>>>>name="George Georgiou"
>>>>f" Welcome {to_low(name)}"
    Welcome george georgiou
```

```
>>>>name="George Georgiou"
>>>>f" Welcome {name.lower()}"
    Welcome george georgiou
```

Η επιλογή *f string για formatted*

Θα μπορούσαμε να περάσουμε **κάποια δεδομένα κλάσης**:

```
class Comedian:
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
    def __str__(self):
        return f"{self.first_name} {self.last_name}is {self.age}"
    def __repr__(self):
        return f"{self.first_name} {self.last_name} ! Surprise"
```

Με βάση τα παραπάνω θα μπορούσαμε να καλέσουμε:

```
>>>> newcomedian = Comedian('Max','Chrutso',29)
>>>> f"{newcomedian}"
```

Οι συναρτήσεις **__str__()** και **__repr__()** είναι οι δύο μέθοδοι που λένε πως ένα αντικείμενο αναπαριστάται ως string και μια από τις 2 θα πρέπει να υπάρχει σε μια κλάση

Η μέθοδος **__repr__()** είναι η επίσημη αναπαράσταση και η πλέον καθορισμένη και η προτιμητέα

Η επιλογή *f string για formatted*

Θα μπορούσαμε να περάσουμε string constants αρκεί να χρησιμοποιήσουθμε Εξωτερικά ” “ και εσωτερικά ’ ‘ ή το ανάποδο ή ”” “”

```
>>>> f”{’George Georgiou’}”
     ‘George Georgiou’
```

```
>>>> f‘{”George Georgiou”}’
     ‘George Georgiou’
```

```
>>>> f’’’George Georgiou’’’
     ‘George Georgiou’
```

Τα παραπάνω θα πρέπει να χρησιμοποιηθούν στην περίπτωση των keys σε dictionary

```
>>>>comedian={‘name’:’George Georgio’, ‘age’:29}
>>>>f”The comedian is {comedian[‘name’]}, {comedian[‘age’]} year old”
    The comedian is George Georgiou, 29
```

Η επιλογή *f string για formatted – σύνταξη*

Στοίχιση

Υπάρχουν διάφοροι τρόποι για να συνδυαστεί η f-string format με στοίχιση:

> Δεξιά στοίχιση στο διαθέσιμο χώρο

< Αριστερή στοίχιση στο διαθέσιμο χώρο

^ Στοίχιση στο κέντρο

= Πρόσθεση κενών μετά το πρόσημο για αριθμούς

Data type (dt)

s Format για strings

d Ακέραιοι δεκαδικού συστήματος

e Επιστημονική σήμανση χρησιμοποιώντας το γράμμα e

f floats. Η ακρίβεια είναι 6 δεκαδικά ψηφία

% τοις %. Πολ.ζει τον αριθμό με 100 και βάζει το % στο τέλος

Η σύνταξη: **{expr : Wdt}** όπου W είναι ο ελάχιστος αριθμός ψηφίων/χαρακτήρων και dt (data type). Σύμβολα στοίχισης εισέρχονται μετά το :

Η επιλογή *f string για formatted* - *Παράδειγμα*

```
APPLES = .50
BREAD = 1.50
CHEESE = 2.25
numApples = 3
numBread = 4
numCheese = 2
prcApples = 3 * APPLES
prcBread = 4 * BREAD
prcCheese = 2 * CHEESE
strApples = 'Apples'
strBread = 'Bread'
strCheese = 'Cheese'
total = prcBread + prcBread + prcApples
print(f'{"My Grocery List":^30s}')
print(f'{"="*30}')
print(f'{strApples}\t{numApples:10d}\t\t${prcApples:>5.2f}')
print(f'{strBread}\t{numBread:10d}\t\t${prcBread:>5.2f}')
print(f'{strCheese}\t{numCheese:10d}\t\t${prcCheese:>5.2f}')
print(f'{"Total:":>19s}\t\t${total:>4.2f}')
```

```
                My Grocery List
==============================
Apples                3           $ 1.50
Bread                 4           $ 6.00
Cheese                2           $ 4.50
                   Total:         $13.50
```

Η επιλογή *f string για formatted*

Για μεγάλα νούμερα, χρειάζεται να εισαχθεί υποδιαστολή :

Ο τρόπος που χρησιμοποιείται με ή χωρίς στοίχιση είναι ο ακόλουθος:

```
Number = 1000000
print(f'The number, 1000000, formatted with a , {Number:,.2f}')
```

```
Number = 1000000
print(f'The number, 1000000, formatted with a , and right-aligned in a width of
15 digits {Number:>15,.2f}')
```

Οπτικοποίηση δεδομένων

Γραφήματα - Matplotlib

Matplotlib - https://matplotlib.org/stable/index.html

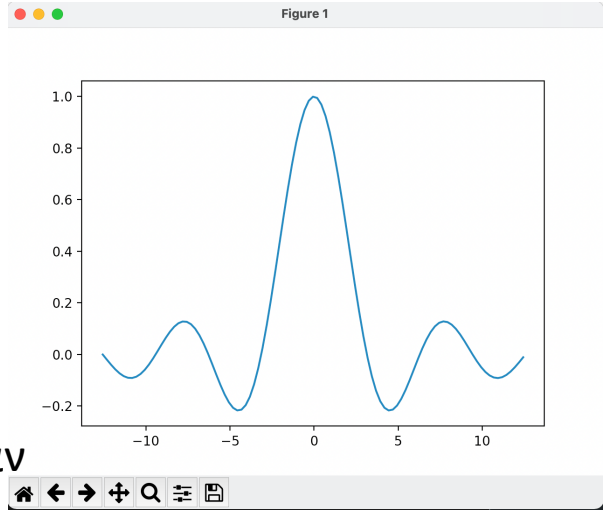
Για οπτικοποίηση δεδομένων θα χρησιμοποιήσουμε τα γραφικά πακέτα τα οποία μας προσφέρονται από την PYTHON

Θα χρησιμοποιήσουμε την βιβλιοθήκη **matplotlib** και το module **pyplot**

Η εντολή που θα πρέπει να δώσουμε είναι: **import matplotlib.pyplot as plt**

Η πλέον συνηθισμένη εντολή για να κάνουμε το γράφημα μιας συνάρτησης είναι να δημιουργήσουμε μια λίστα (array) με τιμές της ανεξάρτητης μεταβλητής και να υπολογίσουμε τις αντίστοιχες τιμές της συνάρτησης

```
>>>>import numpy as np
>>>>import matplotlib.pyplot as plt
>>>>x = np.arange(-4*np.pi, 4*np.pi, 0.25)
>>>>sincx = np.sin(x)/x
>>>>plt.figure()
>>>>plt.plot(x,sincx)
>>>>plt.show()
```



Η μέθοδος figure χρειάζεται για να δημιουργήσουμε έναν κανβά για να σχεδιαστεί το γράφημα

Η μέθοδος plot θα κάνει ένα 2-Δ γράφημα αν δοθεί ένα σετ ζευγών (x,y).

Η μέθοδος show χρειάζεται για να δούμε στο τέλος όλα τα γραφήματα

Η εντολή plot και σχετιζόμενες επιλογές

Η εντολή plot δέχεται και επιπλέον προαιρετικά ορίσματα που βοηθούν να αλλάξουμε το χρώμα, το είδος του συμβόλου (marker) ή της γραμμής της γραφικής

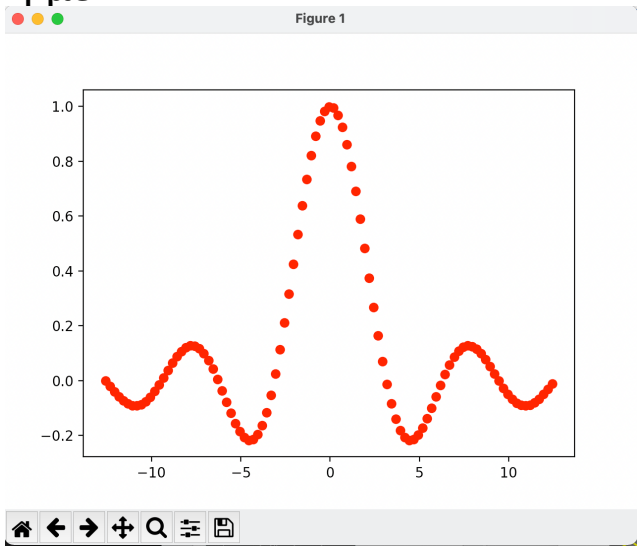
Η σύνταξη είναι: plt.plot(x, y, '**CS**')

C: αναφέρεται στο χρώμα [k=μαύρο, g=πράσινο, b=μπλε, r=κόκκινο, c=κυανό, y=κίτρινο, m=magenta, w=άσπρο]

S: αναφέρεται στον marker [o=κύκλοι, s=τετράγωνο, .=τελεία, v=τρίγωνο κάτω ^=τρίγωνο πάνω, <=τρίγωνο αριστερά, >=τρίγωνο δεξιά D=διαμάντι, d=λεπτό διαμάντι, *=αστεράκι, x=x, +=+]

S: αναφέρεται στο είδος της γραμμής [- = συνεχής, --=διακεκομμένη, -. = -. : = διακεκομμένη με .

```
>>>>plt.clf()    #clear figure
>>>>plt.figure()
>>>>plt.plot(x,sincx,'ro')
>>>>plt.show()
```



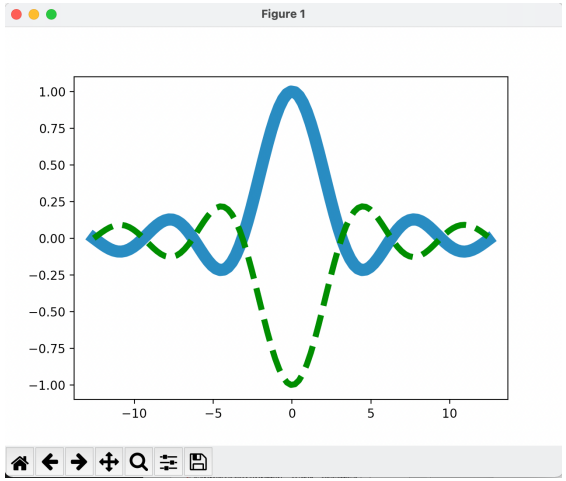
Αποθήκευση γραφήματος σε pdf format γίνεται δίνοντας την εντολή:

plt.savefig("myfigures.pdf")

Η εντολή plot και σχετιζόμενες επιλογές

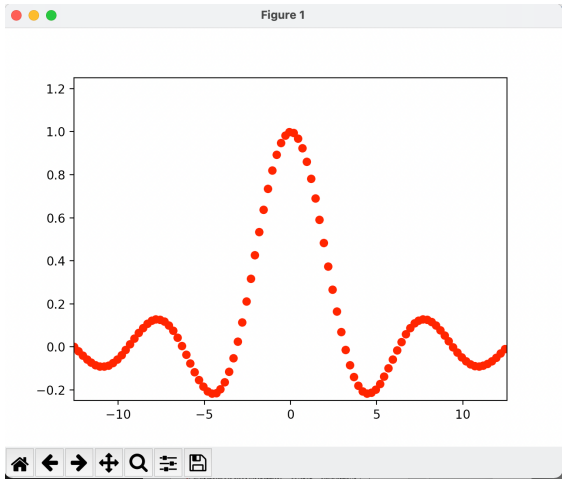
Μπορούμε να έχουμε επιπλέον επιλογές ως προς το πάχος της γραμμής (**lw**) ή το πόσο διαφανής ή συμπαγής (**alpha**) είναι (alpha=1 συμπαγής, 0=διαφανής)

```
>>>>plt.clf()    #clear figure
>>>>plt.figure()
>>>>plt.plot(x,sincx,lw=10,alpha=0.5)
>>>>plt.plot(x,-sincx,'g--',lw=5)
>>>>plt.show()
```



Μπορούμε να δώσουμε το εύρος της κλίμακας του x και y άξονα

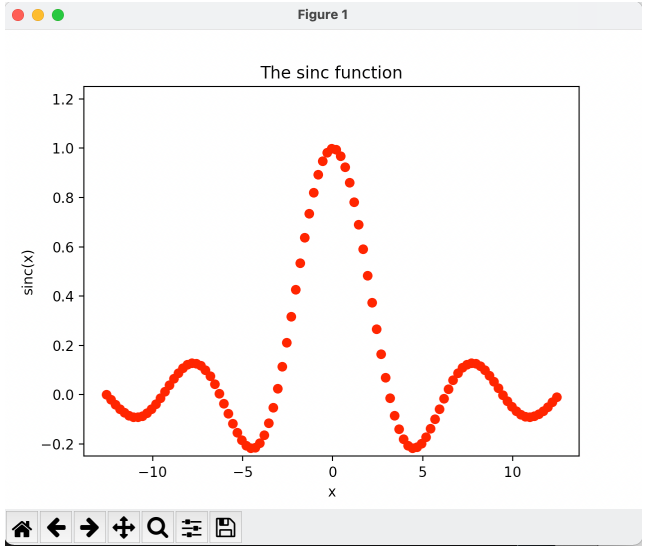
```
>>>>plt.clf()    #clear figure
>>>>plt.figure()
>>>>plt.plot(x,sincx,'ro')
>>>>plt.xlim(-4*np.pi, 4*np.pi)
>>>>plt.ylim(-0.25,1.25)
>>>>plt.show()
```



Η εντολή plot και σχετιζόμενες εντολές

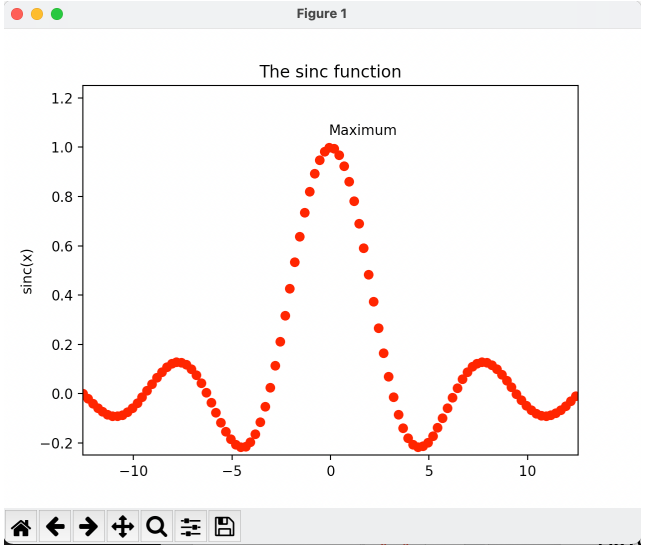
Σε όλα τα γραφήματα θα πρέπει να υπάρχουν κατάλληλα ονοματισμένοι άξονες. Αυτό γίνεται με τις εντολές **title**, **xlabel** και **ylabel** που δίνουν τίτλο και ονόματα στους x, y- άξονες

```
>>>>plt.clf()   #clear figure
>>>>plt.figure()
>>>>plt.plot(x,sincx,'ro')
>>>>plt.xlim(-4*np.pi, 4*np.pi)
>>>>plt.ylim(-0.25,1.25)
>>>>plt.title('The sinc function')
>>>>plt.xlabel('x')
>>>>plt.ylabel('sinc(x)')
>>>>plt.show()
```



Πολλές φορές είναι χρήσιμο να έχουμε text στο γράφημα για να επισημάνουμε κάτι.
Αυτό επιτυγχάνεται με την εντολή **plt.text(x,y,s)** όπου x, y οι συντεταγμένες και s string

```
>>>>plt.text(-0.1, 1.1,'Maximum')
```



Ευπαρουσίαστα γραφήματα

Μπορούμε να αλλάξουμε τον y-άξονα από γραμμικό σε ημιλογαριθμικό άξονα:

plt.semilogy

Αν επιθυμούμε ημιλογαριθμικό τον x-άξονα τότε μπορούμε να γράψουμε:

plt.semilogx

Αν επιθυμούμε λογαριθμικούς και τους δύο άξονες τότε δίνουμε την εντολή: :

Αλλαγή του στυλ της γραμμής:

plt.loglog

Μπορούμε να αλλάξουμε το στυλ της γραμμής αφού σχεδιάσουμε τη γραμμή:

plt.setp (**lines[0]**, **linestyle**='--', **linewidth**=3, **color**='r')

Αλλαγή της πρώτης γραμμής (lines[0]) σε --, με πάχος 3 και χρώματος κόκκινο

Εισαγωγή legend: (περιγραφή μιας γραμμής για παράδειγμα**)**

plt.plot(xvalue, yvalue, **label**="Energy 1")

plt.plot(xvalue, yvalue*yvalue, **label**="Energy 2")

plt.legend()

Εισαγωγή σφαλμάτων:

Μπορούμε να κάνουμε ένα γράφημα με σφάλματα (error bars)

plt.errorbar(xvalue, yvalue, yerr=y_errors, xerr=x_errors)

Πολλαπλά γραφήματα στον ίδιο κανβά

Πολλές περιπτώσεις χρειάζεται να έχουμε πολλά διαφορετικά γραφήματα στον ίδιο κανβά ακόμα και γραφήματα με διαφορετικούς άξονες

Κάθε φορά που δίνουμε την εντολή plt.figure() δημιουργείται ένας νέος κανβάς που έχει ένα νούμερο (1,2,3, κλπ)

Μπορούμε να δώσουμε νούμερο σε κάποιο figure και να προσδιορίσουμε το μέγεθός του:

plt.figure(num=3, figsize=(7,5)) Το μέγεθος είναι (x=7)x (y=5) ίντσες

Με βάση τον αριθμό του figure επιλέγουμε ποιο μπορεί να είναι ενεργό για γράφημα:

Αν θέλουμε να δημιουργήσουμε πολλαπλά γραφήματα μέσα στο ίδιο figure, μπορούμε να χωρίσουμε τον κανβά σε πολλά τμήματα χρησιμοποιώντας την εντολή subplot()

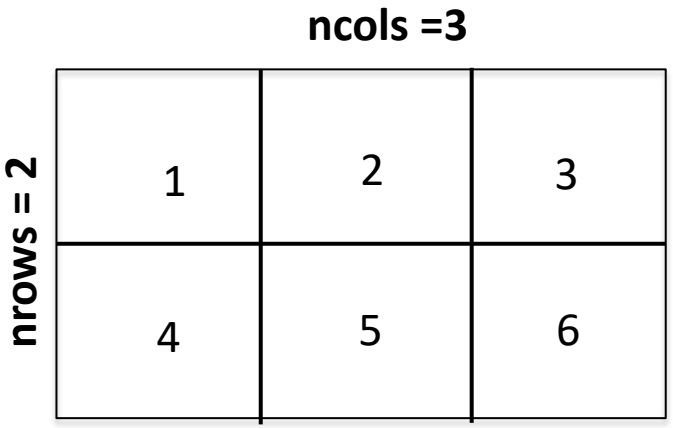
plt.subplot(nrows, ncols, curplot)

Η μέθοδος δέχεται 3 ορίσματα, το 1ο δηλώνει τον αριθμό των γραμμών, το 2ο τον αριθμό των στηλών και το 3ο ποια θέση από τις nrows*ncols είναι ενεργός

```
>>>>plt.subplot(2,3,1)
```

2γραμμές, 3 στήλες

Τα νούμερα μπορούν να δοθούν ως ένα νούμερο πχ. 231 αντί για 2,3,2



Παράδειγμα

```
import numpy as np
import matplotlib.pyplot as plt
func1=lambda x: 4*np.exp(-2*x)
func2=lambda x: 0.5*x**2
x=np.arange(0.,2.1,0.1)
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.plot(x,func1(x),'r-')
plt.plot(x,func2(x),'b--')
plt.text(0.4,2.0,r'f(x)=$4e^{-2x}$')
plt.text(1.5,1.0,r'f(x) = $0.5x^2$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.xlim(0,2)
plt.grid(True)
plt.subplot(1,2,2)
plt.plot(x,func1(x)-func2(x),'b-')
plt.text(0.5,1.5,r'f(x) = $4e^{-2x} - 0.5x{^2}$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.xlim(0,2)
plt.grid(True)
plt.axhline(y=0,color='red',linestyle='--')
plt.tight_layout()
plt.show()
```