Le Bestiaire de l'Algomathique

Tracer...

1. Représentation d'une fonction

- module numpy
- np.ma.masked_array
- plot

1.1 Du bon goût...

Écrire et tester le programme suivant.

```
1  #-*- coding : utf8 -*-
2  # python 3
3
4  from matplotlib import pyplot as plt
5  import numpy as np
6
7  x = np.linspace(-10,10,500)
8  y1 = np.cos(x)
9  y2 = 1/24 * x**4 - 1/2 * x**2 + 1
10 plt.plot(x, y1, 'r')
11 plt.plot(x, y2, 'b-.', linewidth=2)
12 plt.plot(x, np.ma.masked_array(np.sqrt(y1)), 'g-')
13 plt.ylim(-5, 5)
14 plt.grid()
15 plt.legend(('cos', 'poly', 'sqrt'), loc='upper right')
16 plt.show()
```

1. Comprendre le rôle de `np.ma.masked_array`.
2. Tester la représentation graphique d'autres fonctions.

1.2 ... au mauvais

```python
#-*- coding : utf8 -*-
# python 3

from matplotlib import pyplot as plt
import numpy as np

x = np.linspace(-10,10,500)
y1 = np.cos(x)
y2 = 1/24 * x**4 - 1/2 * x**2 + 1

# personnalisation de la fenêtre bien laide
# pour montrer différentes possibilités...
# voir : https://matplotlib.org/users/customizing.html
with plt.rc_context({'axes.edgecolor':'orange',
    'xtick.color':'red', 'ytick.color':'green',
    'figure.facecolor':'#F5BC88',
    'axes.facecolor':'#C1CCED',
    'grid.color':'#0E2E92', 'grid.linewidth':'3',
    'legend.facecolor':'yellow'}):
    repere1 = plt.subplot(121)
    repere1.set_xlim(1.1*min(x), 1.1*max(x))
    repere1.xaxis.set_ticks_position('bottom')
    repere1.spines['bottom'].set_position(('data', 0))
    repere1.tick_params(axis='x', labelsize=14)
    repere1.set_ylim(-5, 5)
    repere1.yaxis.set_ticks_position('left')
    repere1.spines['left'].set_position(('data', 0))
    repere1.grid()
    repere1.plot(x, y1, 'r')
    repere1.plot(x, y2, 'b-.', linewidth=2)
    repere1.legend(('cos', 'poly'), loc='upper right')
```

```
32
33 # paramètres par défaut
34 repere2 = plt.subplot(122)
35 repere2.grid()
36 repere2.plot(x, y1, 'r')
37 repere2.plot(x, np.ma.masked_array(np.sqrt(y1)), 'g-')
38 repere2.legend(('cos', 'sqrt'), loc='upper right')
39
40 plt.show()
```

2. Tableaux de fils

Le programme suivant trace des segments...

```
1  #-*- coding : utf8 -*-
2  # python 3
3
4  from matplotlib import pyplot as plt
5  import numpy as np
6
7  x = np.linspace(0, 10, 5)
8  # liste de 5 nombres, donc (5-1) intervalles
9  # sur [0;10[
10 y = 10 * np.ones(5)
11 plt.plot([x, y] , [y, 10 - x])
12 plt.axis("equal")
13 plt.show()
```

2.1 Conjecture - démonstration

1. À partir du programme, écrire une fonction qui permet de tracer un nombre de segments défini par l'utilisateur.
2. L'ensemble des segments fait apparaître une courbe : émettre une conjecture quand à sa nature et préciser certaines caractéristiques.
3. On se place dans un repère orthonormé $\left(O;\vec{\imath};\vec{\jmath}\right)$; $\mathscr{D}$ une droite d'équation $y = mx + p$.
   Démontrer que la distance de O à la droite $\mathscr{D}$ est donnée par :
   $d(O,\mathscr{D}) = \dfrac{|p|}{\sqrt{1+m^2}}$
   Aide : si la droite $\mathscr{D}$ coupe l'axe des ordonnées en A et l'axe des abscisses en B, on peut calculer l'aire du triangle OAB de deux façons différentes.
4. Confirmer ou infirmer la conjecture précédente.

2.2 Jolis tableaux

1. Compléter le programme pour faire afficher les segments dans les quatre coins du carré de diagonale $(-10;-10)$ $(10;10)$. Le nombre de segments doit être donné par l'utilisateur.
2. Modifier le programme afin qu'il fasse afficher un as de carreau. Le nombre de segments doit être donné par l'utilisateur.