TP8

Matplotlib

1. Relier des points

`matplotlib` est un module permettant de créer des graphiques et autres rendus visuels avec python.

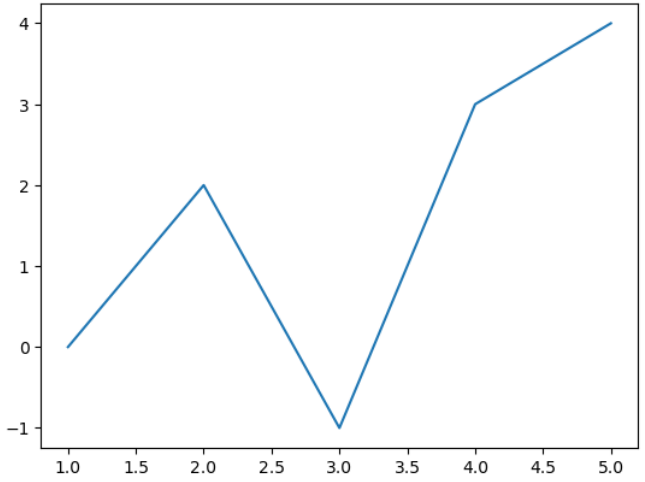
Nous allons utiliser le sous-module `matplotlib.pyplot`. Ce nom étant très long, on va l'importer en défissant un alias.

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

Ce sous-module est maintenant chargé et nommé `plt`. Pour utiliser une fonction de ce module on utilisera donc `plt.fonction()`.

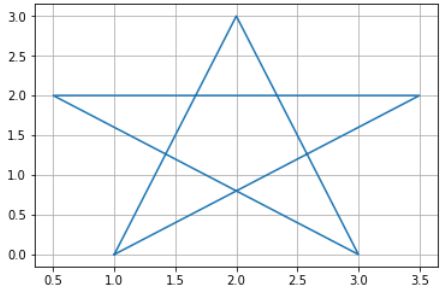
Voici un exemple de graphique construit avec matplotlib.pyplot.

In [2]:
```
abscisses = [1, 2, 3, 4, 5]
ordonnées = [0, 2, -1, 3, 4]
plt.plot(abscisses, ordonnées)
plt.show()
```

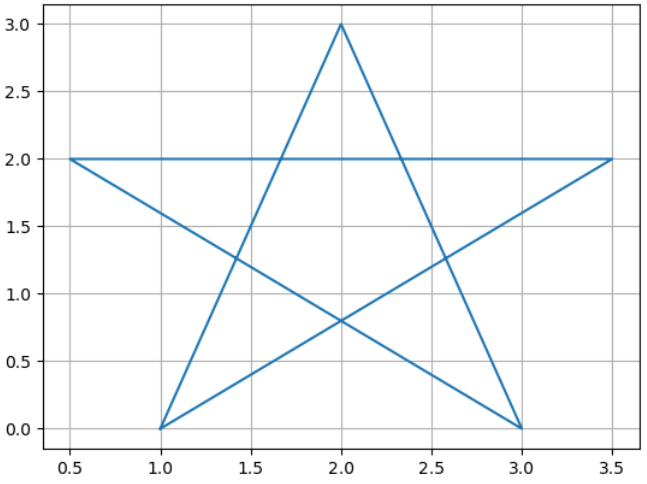


Exercice 1

1. Réaliser cette figure (sans la grille).

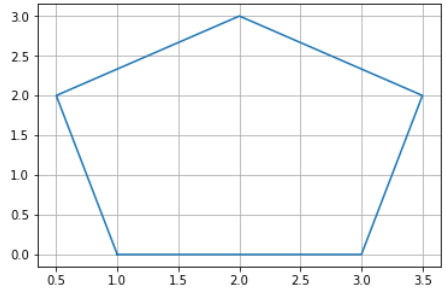


In [3]:
```
x = [1,3.5,0.5,3,2,1]
y = [0,2,2,0,3,0]
plt.plot(x,y)
plt.grid(visible=True)
plt.show()
```

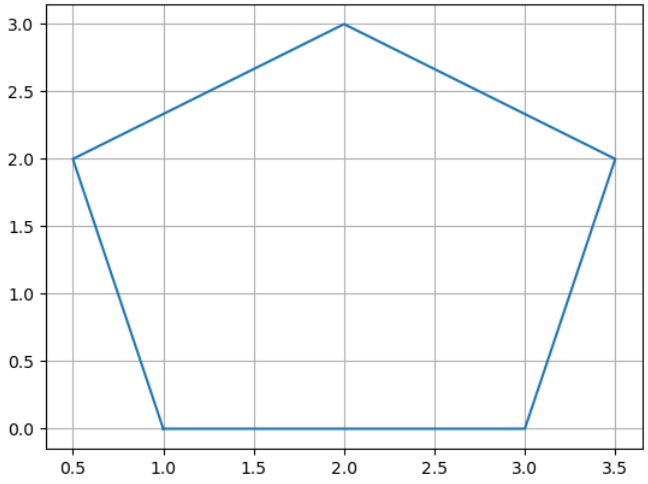


Exercice 1 - suite

2. Réaliser cette figure (sans la grille).



In [4]:
```
x = [1,0.5,2,3.5,3,1]
y = [0,2,3,2,0,0]
plt.plot(x,y)
plt.grid(visible=True)
plt.show()
```



Exercice 2

On souhaite tracer la courbe de la fonction $\exp$ sur l'intervalle $[-5, 2]$.

1. Créer (sans utiliser de module) la liste $x = [-5, -4.9, -4.8, \ldots, 2]$.
2. Créer (sans utiliser de module) la liste $y = [\exp(-5), \exp(-4.9), \exp(-4.8), \ldots, \exp(2)]$.
3. Tracer alors le graphique demandé.

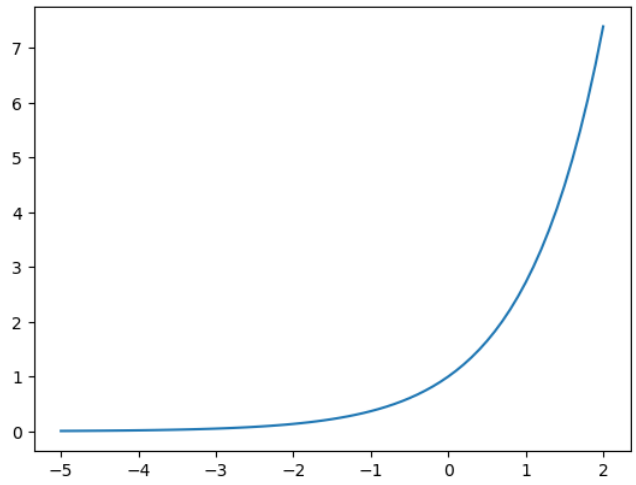
In [10]:
```
x = [-5+k/10 for k in range(71)]
print(x)

# ou alors
t = -5
x = []
while t <= 2:
    x.append(t)
    t = t + 0.1
print(x)
```

```
[-5.0, -4.9, -4.8, -4.7, -4.6, -4.5, -4.4, -4.3, -4.2, -4.1, -4.0, -3.9, -3.8, -3.7, -3.
6, -3.5, -3.4, -3.3, -3.2, -3.1, -3.0, -2.9, -2.8, -2.7, -2.6, -2.5, -2.4, -2.3, -2.2, -
2.1, -2.0, -1.9, -1.7999999999999998, -1.7000000000000002, -1.6, -1.5, -1.4, -1.29999999
99999998, -1.2000000000000002, -1.1, -1.0, -0.9000000000000004, -0.7999999999999998, -0.
7000000000000002, -0.5999999999999996, -0.5, -0.40000000000000036, -0.2999999999999998,
-0.20000000000000018, -0.09999999999999964, 0.0, 0.09999999999999964, 0.2000000000000001
8, 0.2999999999999998, 0.40000000000000036, 0.5, 0.5999999999999996, 0.7000000000000002,
0.7999999999999998, 0.9000000000000004, 1.0, 1.0999999999999996, 1.2000000000000002, 1.2
999999999999998, 1.4000000000000004, 1.5, 1.5999999999999996, 1.7000000000000002, 1.7999
999999999998, 1.9000000000000004, 2.0]
[-5, -4.9, -4.800000000000001, -4.700000000000001, -4.600000000000001, -4.50000000000000
2, -4.400000000000002, -4.3000000000000025, -4.200000000000003, -4.100000000000003, -4.0
000000000000036, -3.9000000000000035, -3.8000000000000034, -3.7000000000000033, -3.6000
0000000000003, -3.500000000000003, -3.400000000000003, -3.300000000000003, -3.2000000000000
03, -3.1000000000000028, -3.0000000000000027, -2.9000000000000026, -2.8000000000000025,
-2.7000000000000024, -2.6000000000000023, -2.500000000000002, -2.400000000000002, -2.300
000000000002, -2.200000000000002, -2.100000000000002, -2.0000000000000018, -1.9000000000
000017, -1.8000000000000016, -1.7000000000000015, -1.6000000000000014, -1.5000000000000
13, -1.4000000000000012, -1.3000000000000012, -1.200000000000001, -1.100000000000001, -
1.0000000000000009, -0.9000000000000009, -0.8000000000000009, -0.700000000000001, -0.600
000000000001, -0.500000000000001, -0.400000000000001, -0.30000000000000104, -0.200000000
00000104, -0.10000000000000103, -1.0269562977782698e-15, 0.09999999999999898, 0.19999999
999999898, 0.299999999999999, 0.399999999999999, 0.499999999999999, 0.599999999999999,
0.699999999999999, 0.7999999999999989, 0.8999999999999989, 0.9999999999999989, 1.0999999
99999999, 1.199999999999999, 1.2999999999999992, 1.3999999999999992, 1.4999999999999993,
1.5999999999999994, 1.6999999999999995, 1.7999999999999996, 1.8999999999999997, 1.999999
9999999998]
```

In [9]:
```
import numpy as np
y = []
for t in x:
    y.append(np.exp(t))
print(y)

# ou alors
y = [0]*len(x)
for k in range(len(x)):
    y[k] = np.exp(x[k])

# ou encore
y = [np.exp(-5+k*0.1) for k in range(71)]
```

```
[0.006737946999085467, 0.007446583070924338, 0.008229747049020021, 0.009095277101695807,
0.010051835744633567, 0.011108996538242287, 0.012277339903068415, 0.013568559012200897,
0.014995576820477665, 0.016572675401761192, 0.018315638888734113, 0.02024191144580432,
0.02237077185616552, 0.02472352647033931, 0.027323722447292472, 0.030197383422318407, 0.
033373269960325976, 0.036883167401239, 0.040762203978366 1, 0.04504920239355768, 0.04978
7068367863806, 0.055023220056407085, 0.06081006262521782, 0.0672055127397496, 0.07427357
821433371, 0.0820849986238986, 0.09071795328941232, 0.10025884372280353, 0.1108031583623
3366, 0.1224564282529817, 0.13533528323661245, 0.14956861922263479, 0.16529888822158628,
0.1826835240527344, 0.2018965179946551, 0.22313016014842954, 0.24659696394160618, 0.2725
3179303401226, 0.30119421191220175, 0.3328710836980792, 0.36787944117144206, 0.406569659
7405987, 0.4493289641172212, 0.4965853037914091, 0.548811636094026, 0.6065306597126328,
0.6703200460356386, 0.7408182206817171, 0.818730753077981, 0.9048374180359586, 0.9999999
99999999, 1.1051709180756466, 1.2214027581601685, 1.3498588075760019, 1.491824697641268
8, 1.6487212707001264, 1.822118800390507, 2.0137527074704744, 2.225540928492465, 2.45960
31111569467, 2.718281828459042, 3.0041660239464303, 3.3201169227365446, 3.66929666761924
1, 4.055199966844672, 4.481689070338062, 4.953032424395112, 5.473947391727196, 6.0496474
64412944, 6.685894442279268, 7.389056098930649]
```

In [11]:
```
plt.plot(x,y)
plt.show()
```



2. On s'aide de Numpy

Utilisons la bibliothèque numpy pour simplifier le tracer de fonctions mathématiques.

Le module numpy défini un nouveau type de données : array (tableau). Ce type de données facilite les calculs termes à termes sur les tableaux ou listes de nombres, comme indiqué sur le document joint.

In [13]:
```
x = np.array([2,3,5])
x**2
```

Out[13]:
```
array([ 4,  9, 25])
```

In [14]:
```
3*x
```

Out[14]:
```
array([ 6,  9, 15])
```

Attention !! Il ne faut pas confondre l'opérateur * du type liste et l'opérateur * du type array.

In [15]:
```
L = [2, 3, 5] # L est une liste
3*L
```

Out[15]:
```
[2, 3, 5, 2, 3, 5, 2, 3, 5]
```

Le module numpy dispose également d'une fonction **linspace** et d'une fonction **arange** permettant de créer des tableaux de nombres régulièrement espacés.

In [16]:
```
np.linspace(1,10,5)
```

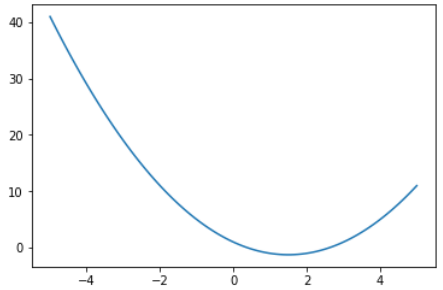
Out[16]:
```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

In [17]:
```
np.arange(1, 5, 0.5) # Attention, la dernière valeur du tableau sera < 5
```

Out[17]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

Exercice 3
Tracer la courbe de la fonction $x \mapsto x^2 - 3x + 1$ sur l'intervalle $[-5, 5]$ en s'aidant de numpy

In [24]:
```
x = np.linspace(-5,5,100)
y = x**2 - 3*x + 1
plt.plot(x,y)
plt.show()
```



Le module numpy dispose également de toutes les fonctions usuelles mathématiques. Cependant, ce ne sont pas exactement les même fonctions que celles du module math : les fonctions usuelles de numpy peuvent être appliquées à des listes ou tableaux.

In [18]:
```
x = np.array([2,3,5])
np.log(x)
```

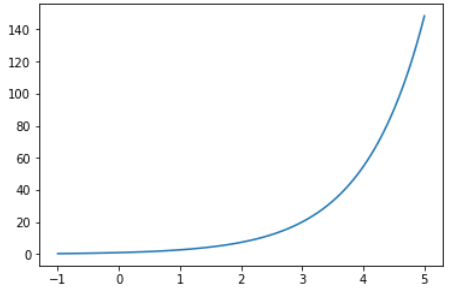
Out[18]:
```
array([0.69314718, 1.09861229, 1.60943791])
```

In [19]:
```
L = [2,3,5]
np.log(L)
```

Out[19]:
```
array([0.69314718, 1.09861229, 1.60943791])
```

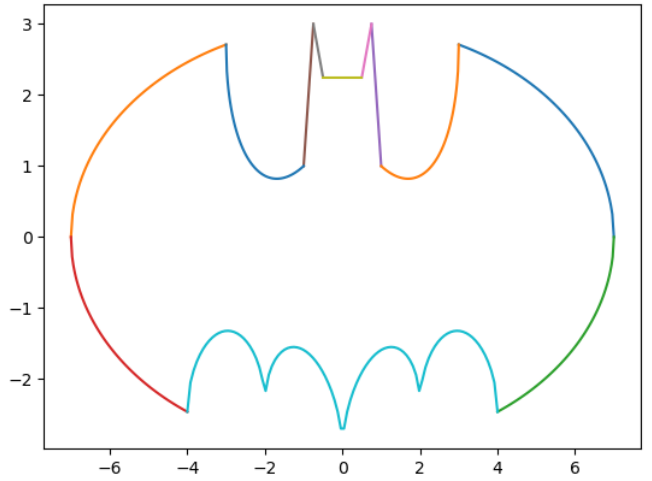
Exercice 4
Tracer la courbe de la fonction $\exp$ sur l'intervalle $[-1, 5]$ en s'aidant de numpy.

In [28]:
```
x = np.linspace(-1,5,100)
y = np.exp(x)
plt.plot(x,y)
plt.show()
```



3. Exercice supplémentaire (voir sur votre feuille)

In [25]:
```
x = np.linspace(3,7,100)
y = 3*np.sqrt(1-x**2/49)
plt.plot(x,y)

x = np.linspace(-7,-3,100)
y = 3*np.sqrt(1-x**2/49)
plt.plot(x,y)

x = np.linspace(4,7,100)
y = -3*np.sqrt(1-x**2/49)
plt.plot(x,y)

x = np.linspace(-7,-4,100)
y = -3*np.sqrt(1-x**2/49)
plt.plot(x,y)

x = np.linspace(0.75,1,100)
y = 9-8*np.abs(x)
plt.plot(x,y)

x = np.linspace(-1,-0.75,100)
y = 9-8*np.abs(x)
plt.plot(x,y)

x = np.linspace(0.5,0.75,100)
y = 0.75+3*np.abs(x)
plt.plot(x,y)

x = np.linspace(-0.75,-0.5,100)
y = 0.75+3*np.abs(x)
plt.plot(x,y)

# Attention, f5 est une fonction constante
x = [-0.5,0.5]     # ou x = np.linspace(-0.5,0.5,100)
y = [2.25, 2.25]   #    y = 2.25 + x*0
# x et y doivent être des listes de même taille.
plt.plot(x,y)

x = np.linspace(-4,4,100)
y = np.abs(x/2)-(3*np.sqrt(33)-7)/112*x**2 + np.sqrt(1-( np.abs( np.abs(x)-2) -1)**2) - 3
plt.plot(x,y)

x = np.linspace(-3,-1,100)
y = 6*np.sqrt(10)/7 + 1.5-0.5*np.abs(x) - 6*np.sqrt(10)/14*np.sqrt(3+2*np.abs(x)-x**2)
plt.plot(x,y)

x = np.linspace(1,3,100)
y = 6*np.sqrt(10)/7 + 1.5-0.5*np.abs(x) - 6*np.sqrt(10)/14*np.sqrt(3+2*np.abs(x)-x**2)
plt.plot(x,y)

plt.show()
```



In [ ]: