VISUALISATION DE DONNÉES À L'AIDE DE PYTHON

Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 Installation de Python

3 C'est quoi Jupyter ?

4 Installation d'Anaconda sur Windows

5 Installation d'Anaconda sur Mac

6 Installation d'Anaconda sur Ubuntu

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

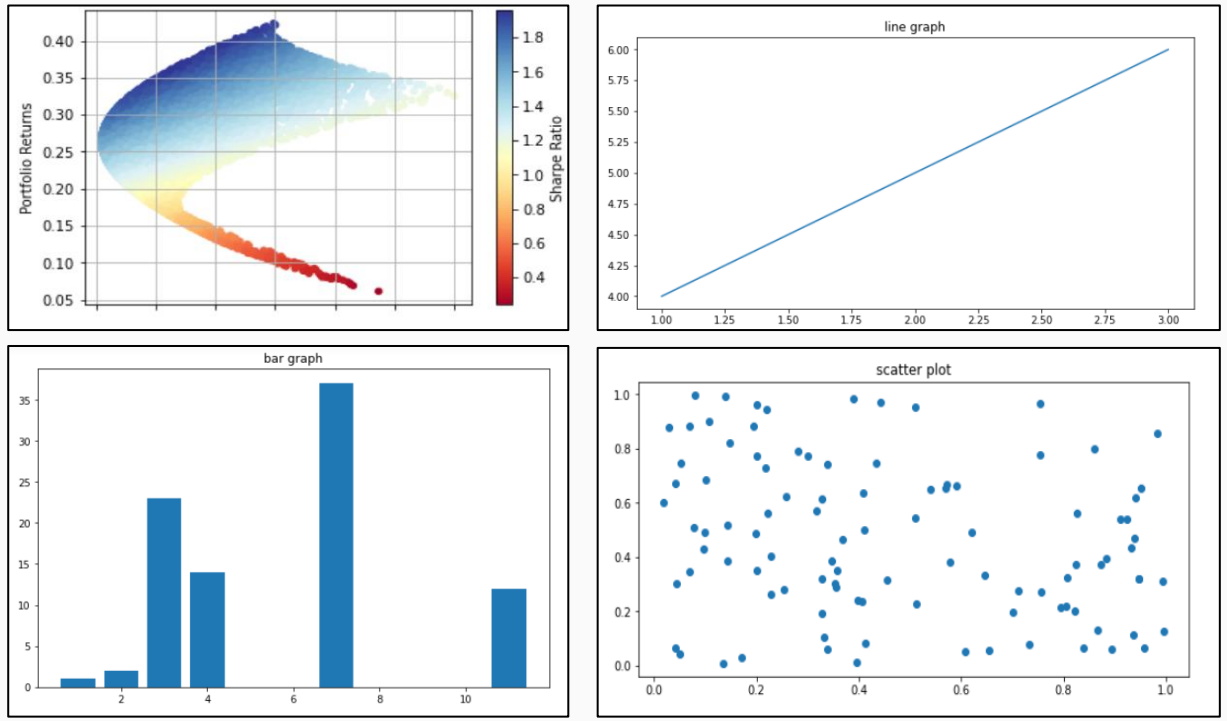
11 Variables

12 Opérateurs Arithmétiques

Meta Brains

À propos de Matplotlib

- Matplotlib est la bibliothèque python la plus populaire pour tracer différents types de graphiques.
- Le module Pyplot de la Matplotlib permet de la faire fonctionner comme Matlab.



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 C'est quoi Jupyter ?

4 Installation d'Anaconda sur Windows

5 Installation d'Anaconda sur Mac

6 Installation d'Anaconda sur Ubuntu

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

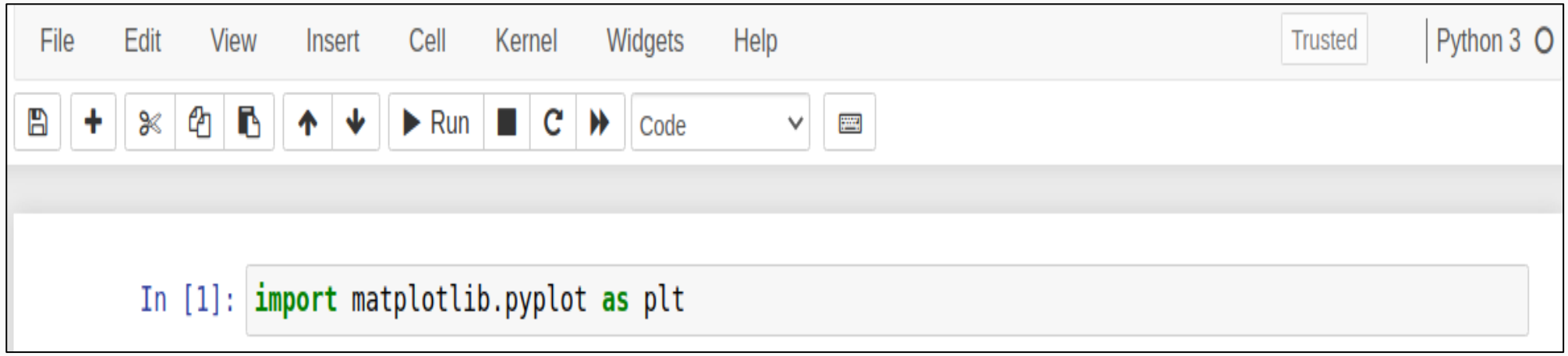
11 Variables

12 Opérateurs Arithmétiques

Meta Brains

Importation de Matplotlib

- Pour importer matplotlib.pyplot, tapez 'import matplotlib.pyplot' dans Jupyter Notebook et exécutez la cellule.
- L'abréviation courante utilisée pour matplotlib.pyplot est plt.



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 Installation d'Anaconda sur Windows

5 Installation d'Anaconda sur Mac

6 Installation d'Anaconda sur Ubuntu

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

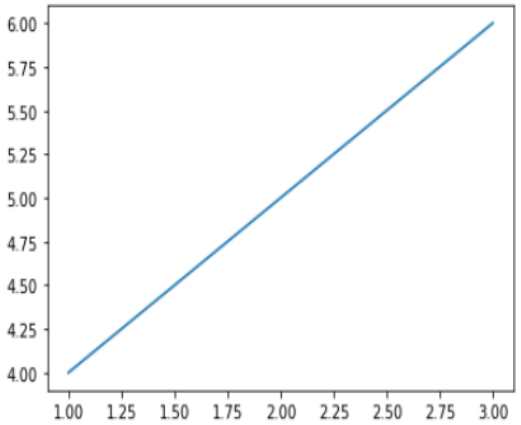
Meta Brains

Tracer des graphiques linéaires (1/2)

- Nous pouvons tracer des graphiques linéaires avec matplotlib en utilisant la fonction .plot().
  - Le premier argument de la fonction .plot() spécifie l'axe des x.
  - Le deuxième argument de la fonction .plot() spécifie l'axe des y.

```
[2]: Axe_x = [1,2,3]
     Axe_y = [4,5,6]
     plt.plot(Axe_x, Axe_y)

[2]: [<matplotlib.lines.Line2D at 0x1c929292e50>]
```



Meta Brains

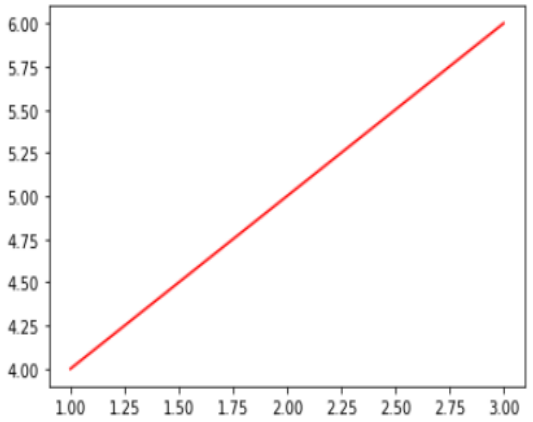
Tracer des graphiques linéaires (2/2)

Changement de couleur

- Nous pouvons également changer la couleur de la ligne en fournissant la couleur comme troisième argument dans la fonction plot().
- Une liste des abréviations des couleurs peut être trouvée à l'adresse suivante : https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html

```
[3]: Axe_x = [1,2,3]
     Axe_y = [4,5,6]
     plt.plot(Axe_x, Axe_y, 'r')

[3]: [<matplotlib.lines.Line2D at 0x1c92938fac0>]
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 **Titre, étiquettes et légende**

5 Installation d'Anaconda sur Mac

6 Installation d'Anaconda sur Ubuntu

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

Meta Brains

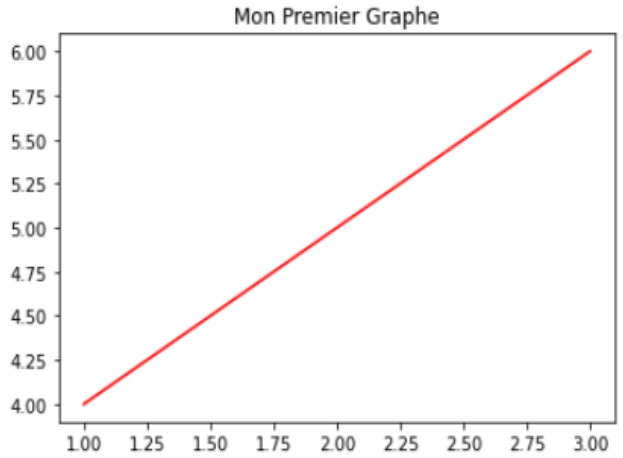
Titre

- Pour définir le titre du graphique, utilisez la fonction .title().

```
[4]: Axe_x = [1,2,3]
     Axe_y = [4,5,6]

     plt.title('Mon Premier Graphe')
     plt.plot(Axe_x, Axe_y, 'r')
```

```
[4]: [<matplotlib.lines.Line2D at 0x1c9294017f0>]
```



Meta Brains

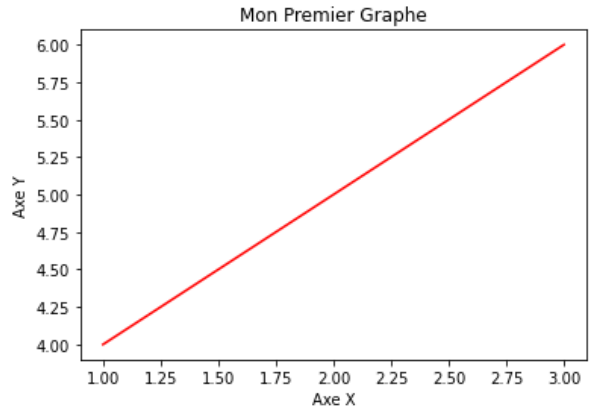
Étiquettes

- Pour attribuer des étiquettes aux axes x et y, utilisez respectivement .xlabel() et .ylabel().

```
[5]: Axe_x = [1,2,3]
     Axe_y = [4,5,6]

     plt.title('Mon Premier Graphe')
     plt.xlabel('Axe X')
     plt.ylabel('Axe Y')
     plt.plot(Axe_x, Axe_y, 'r')
```

```
[5]: [<matplotlib.lines.Line2D at 0x1c92946c310>]
```



Meta Brains

Légende (1/2)

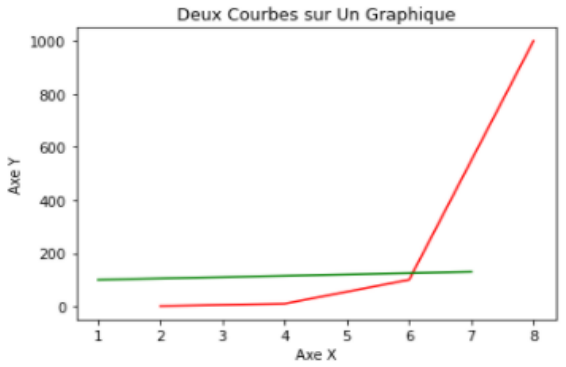
- Nous pouvons tracer plusieurs courbes sur le même graphique simplement en les traçant une par une comme dans l'exemple ci-dessous.

```
[6]: Axe_x1 = [2,4,6,8]
     Axe_y1 = [1,10,100,1000]
     Axe_x2 = [1,3,5,7]
     Axe_y2 = [100,110,120,130]

     plt.title('Deux Courbes sur Un Graphique')
     plt.xlabel('Axe X')
     plt.ylabel('Axe Y')

     plt.plot(Axe_x1, Axe_y1, 'r')
     plt.plot(Axe_x2, Axe_y2, 'g')

[6]: [<matplotlib.lines.Line2D at 0x1c9294d62e0>]
```



Meta Brains

Légende (2/2)

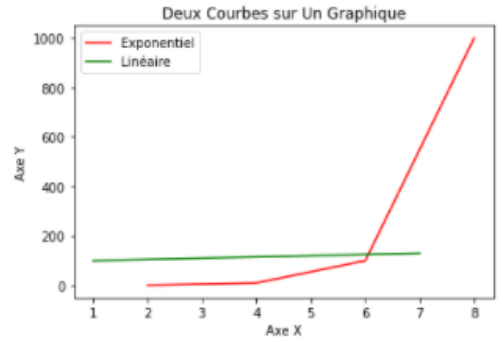
- Si vous tracez plus d'un graphe, il est bon d'ajouter une légende à votre figure en utilisant la fonction .legend().

```
[8]: Axe_x1 = [2,4,6,8]
     Axe_y1 = [1,10,100,1000]
     Axe_x2 = [1,3,5,7]
     Axe_y2 = [100,110,120,130]

     plt.title('Deux Courbes sur Un Graphique')
     plt.xlabel('Axe X')
     plt.ylabel('Axe Y')

     plt.plot(Axe_x1, Axe_y1, 'r')
     plt.plot(Axe_x2, Axe_y2, 'g')
     plt.legend(['Exponentiel','Linéaire'])

[8]: <matplotlib.legend.Legend at 0x1c9295862b0>
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 **Titre, étiquettes et légende**

5 **Tracer des histogrammes**

6 Installation d'Anaconda sur Ubuntu

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

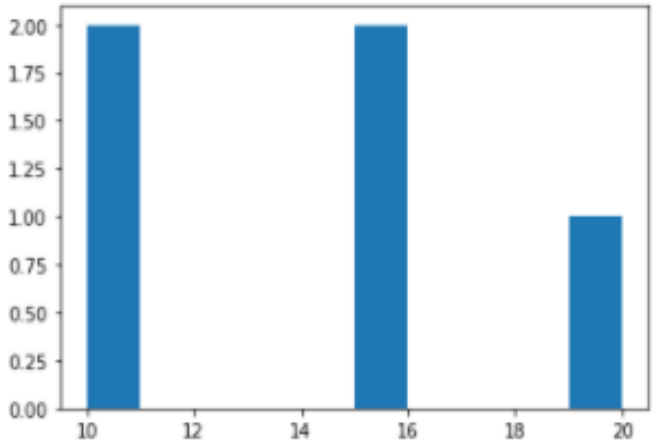
Meta Brains

Tracer des histogrammes (1/3)

- Nous pouvons également tracer des histogrammes en utilisant la fonction .hist().
- Un histogramme est généralement utilisé pour tracer la fréquence qui aide à identifier la distribution des données.

```
[9]: Valeurs = [10, 15, 20, 10, 15]
     plt.hist(Valeurs)
```

```
[9]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
      array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
      <BarContainer object of 10 artists>)
```



Meta Brains

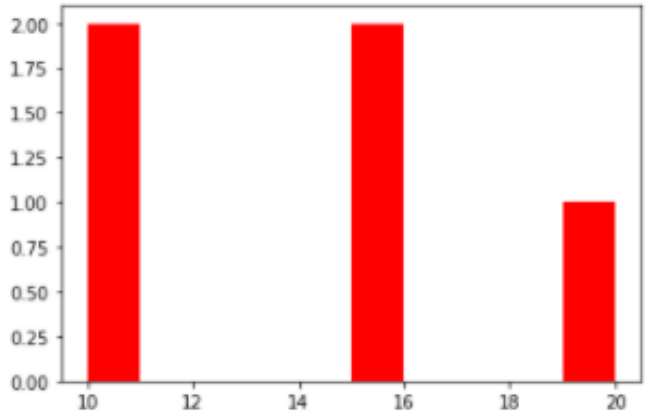
Tracer des histogrammes (2/3)

Changement de couleur

- Nous pouvons également changer la couleur des barres en utilisant le paramètre 'color' dans la fonction hist().

```
[10]: Valeurs = [10, 15, 20, 10, 15]
      plt.hist(Valeurs, color='r')
```

```
[10]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
       array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
       <BarContainer object of 10 artists>)
```



Meta Brains

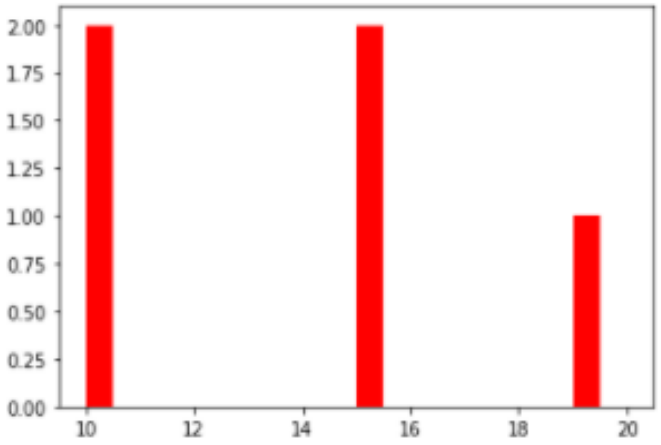
Tracer des histogrammes (3/3)

Modification de la largeur

- Nous pouvons également modifier la largeur des barres en utilisant le paramètre 'width' dans la fonction hist().

```
[11]: Valeurs = [10, 15, 20, 10, 15]
      plt.hist(Valeurs, color='r', width=0.5)
```

```
[11]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
       array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
       <BarContainer object of 10 artists>)
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 **Titre, étiquettes et légende**

5 **Tracer des histogrammes**

6 **Tracer des diagrammes à barres**

7 Comment implémenter Python dans Jupyter

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

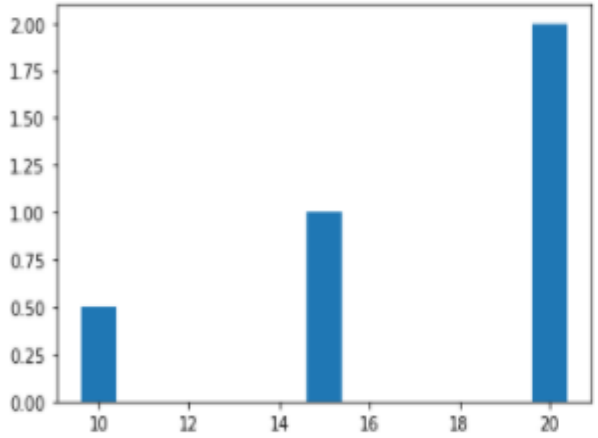
Meta Brains

Tracer des diagrammes à barres (1/2)

- Pour tracer un graphique à barres, utilisez la fonction .bar() de matplotlib.pyplot.
  - Le premier argument de la fonction bar() est l'étiquette x.
  - Le deuxième argument de la fonction .bar() est la hauteur de chaque barre, qui peut être une liste de valeurs ou une valeur unique.

```
[12]: Valeurs = [10, 15, 20]
      plt.bar(Valeurs,[0.5, 1, 2])

[12]: <BarContainer object of 3 artists>
```



Meta Brains

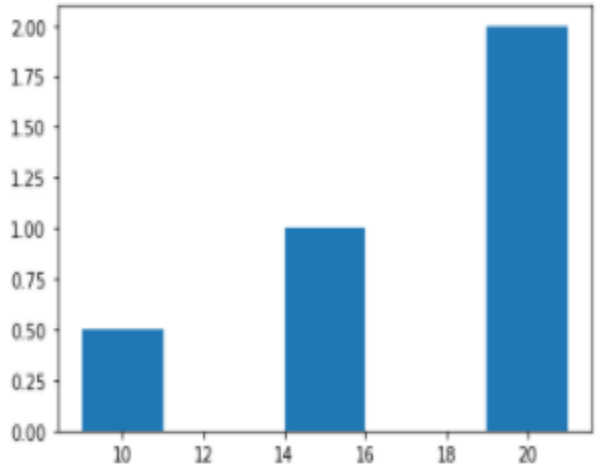
Tracer des diagrammes à barres (2/2)

Modification de la largeur

- Nous pouvons également modifier la largeur des barres dans le graphique à barres en utilisant le paramètre "width".

```
[13]: Valeurs = [10, 15, 20]
      plt.bar(Valeurs,[0.5, 1, 2], width = 2)
```

```
[13]: <BarContainer object of 3 artists>
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 **Titre, étiquettes et légende**

5 **Tracer des histogrammes**

6 **Tracer des diagrammes à barres**

7 **Tracer des diagrammes circulaires**

8 Gestion des répertoires dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

Meta Brains

Tracer des diagrammes circulaires (1/4)

- La fonction .pie() est utilisée pour créer un graphique circulaire dans matplotlib.pyplot.

```
[14]: Valeurs = [20, 20, 35, 25]
      plt.pie(Valeurs)
```

```
[14]: ([<matplotlib.patches.Wedge at 0x1c92964c7f0>,
         <matplotlib.patches.Wedge at 0x1c92949e8e0>,
         <matplotlib.patches.Wedge at 0x1c929793df0>,
         <matplotlib.patches.Wedge at 0x1c9297519a0>],
        [Text(0.8899186877588753, 0.6465637858537406, ''),
         Text(-0.3399187231970732, 1.046162158377023, ''),
         Text(-0.9801071672559598, -0.49938956806635265, ''),
         Text(0.7778174593052022, -0.7778174593052025, '')])
```

Meta Brains

Tracer des diagrammes circulaires (2/4)

Étiquettes

- Pour ajouter des étiquettes dans votre diagramme circulaire, utilisez le paramètre "labels" qui prend une liste d'étiquettes.

```
[15]: Valeurs = [20, 20, 35, 25]
      plt.pie(Valeurs, labels=['a', 'b', 'c', 'd'])
```

```
[15]: ([<matplotlib.patches.Wedge at 0x1c929862c10>,
         <matplotlib.patches.Wedge at 0x1c929871130>,
         <matplotlib.patches.Wedge at 0x1c9298715b0>,
         <matplotlib.patches.Wedge at 0x1c929871a30>],
        [Text(0.8899186877588753, 0.6465637858537406, 'a'),
         Text(-0.3399187231970732, 1.046162158377023, 'b'),
         Text(-0.9801071672559598, -0.49938956806635265, 'c'),
         Text(0.7778174593052022, -0.7778174593052025, 'd')])
```

Meta Brains

Tracer des diagrammes circulaires (3/4)

Explode

- Si vous voulez qu'un ou plusieurs coins de votre camembert ressortent, vous pouvez utiliser le paramètre "explode".
  - Fournissez une liste contenant la distance de chaque coin par rapport au centre au paramètre "explode".

```
[16]: Valeurs = [20, 20, 35, 25]
      plt.pie(Valeurs, labels=['a', 'b', 'c', 'd'], explode=[0, 0.2, 0, 0])
```

```
[16]: ([<matplotlib.patches.Wedge at 0x1c9298b5460>,
         <matplotlib.patches.Wedge at 0x1c9298b5940>,
         <matplotlib.patches.Wedge at 0x1c9298b5dc0>,
         <matplotlib.patches.Wedge at 0x1c9298c3280>],
        [Text(0.8899186877588753, 0.6465637858537406, 'a'),
         Text(-0.4017221274147229, 1.2363734599001182, 'b'),
         Text(-0.9801071672559598, -0.49938956806635265, 'c'),
         Text(0.7778174593052022, -0.7778174593052025, 'd')])
```

Meta Brains

Tracer des diagrammes circulaires (4/4)

Les couleurs

- Nous pouvons également modifier les couleurs des cales en fournissant une liste de couleurs au paramètre "colors".

```
[17]: Valeurs = [20, 20, 35, 25]
      plt.pie(Valeurs, labels=['a', 'b', 'c', 'd'], colors=['r', 'g', 'b', 'y'])
```

```
[17]: ([<matplotlib.patches.Wedge at 0x1c9298f7bb0>,
         <matplotlib.patches.Wedge at 0x1c9299050d0>,
         <matplotlib.patches.Wedge at 0x1c929905550>,
         <matplotlib.patches.Wedge at 0x1c9299059d0>],
        [Text(0.8899186877588753, 0.6465637858537406, 'a'),
         Text(-0.3399187231970732, 1.046162158377023, 'b'),
         Text(-0.9801071672559598, -0.49938956806635265, 'c'),
         Text(0.7778174593052022, -0.7778174593052025, 'd')])
```

Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 **À propos de Matplotlib**

2 **Importation de Matplotlib**

3 **Tracer des graphiques linéaires**

4 **Titre, étiquettes et légende**

5 **Tracer des histogrammes**

6 **Tracer des diagrammes à barres**

7 **Tracer des diagrammes circulaires**

8 **Tracer des diagrammes de dispersion**

9 Entrées/sorties

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

Meta Brains

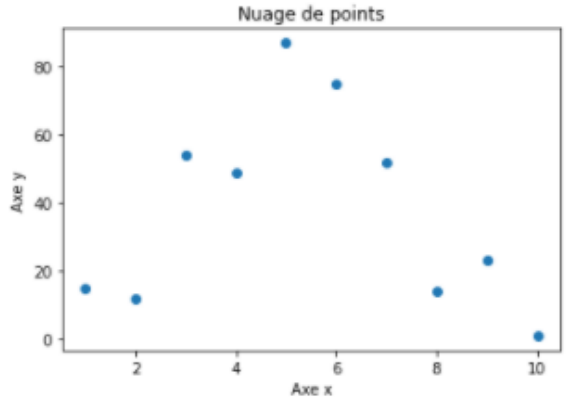
Tracer des diagrammes de dispersion (1/5)

- Nous pouvons également tracer des diagrammes de dispersion en utilisant la fonction .scatter() dans matplotlib.pyplot.
- Elle prend deux listes comme arguments ;
  - La première liste spécifie les valeurs de l'axe des x.
  - La deuxième liste spécifie les valeurs de l'axe des y.

```
[18]: Axe_x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      Axe_y = [15, 12, 54, 49, 87, 75, 52, 14, 23, 1]

      plt.title('Nuage de points')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.scatter(Axe_x, Axe_y)
```

```
[18]: <matplotlib.collections.PathCollection at 0x1c92992f550>
```



Meta Brains

Tracer des diagrammes de dispersion (2/5)

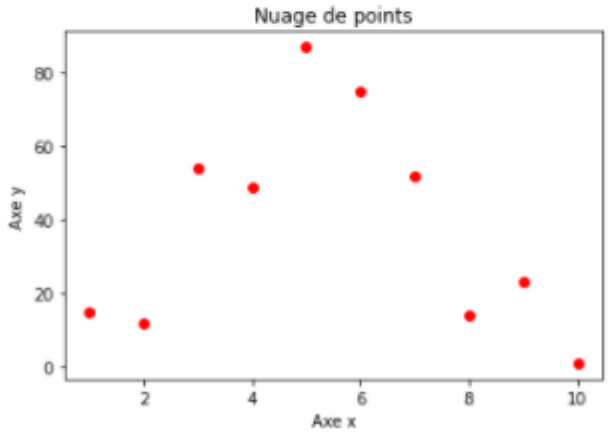
Les couleurs

- Nous pouvons également changer la couleur des points en utilisant le paramètre 'color'.

```
[19]: Axe_x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      Axe_y = [15, 12, 54, 49, 87, 75, 52, 14, 23, 1]

      plt.title('Nuage de points')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.scatter(Axe_x, Axe_y, color='r')
```

```
[19]: <matplotlib.collections.PathCollection at 0x1c92998a3d0>
```



Meta Brains

Tracer des diagrammes de dispersion (3/5)

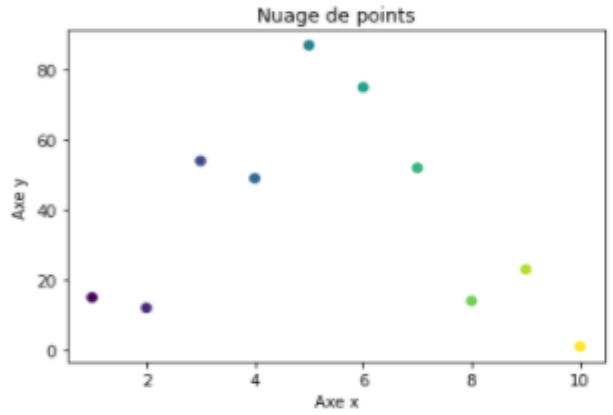
Carte des couleurs

- Si vous souhaitez donner à chaque point une couleur différente, vous pouvez fournir une liste de couleurs ou d'entiers au paramètre 'c'.

```
[20]: Axe_x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      Axe_y = [15, 12, 54, 49, 87, 75, 52, 14, 23, 1]

      plt.title('Nuage de points')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.scatter(Axe_x, Axe_y, c=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
[20]: <matplotlib.collections.PathCollection at 0x1c9299cee20>
```



Meta Brains

Tracer des diagrammes de dispersion (4/5)

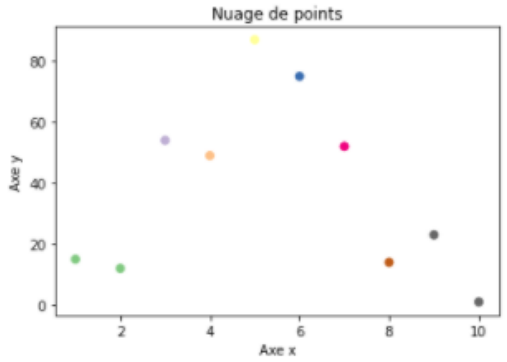
Carte des couleurs

- Vous pouvez également définir la carte des couleurs à l'aide du paramètre 'cmap'.
- Pour cela, vous devrez fournir une liste d'entiers qui seront mis en correspondance avec les couleurs.
- Nous avons utilisé la carte de couleurs 'Accent', qui est l'une des nombreuses cartes de couleurs intégrées dans Matplotlib.

```
[21]: Axe_x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      Axe_y = [15, 12, 54, 49, 87, 75, 52, 14, 23, 1]

      plt.title('Nuage de points')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.scatter(Axe_x, Axe_y, c=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], cmap='Accent')
```

```
[21]: <matplotlib.collections.PathCollection at 0x1c92aa11040>
```



Meta Brains

Tracer des diagrammes de dispersion (5/5)

Carte des couleurs

- Pour voir les cartes de couleurs disponibles dans matplotlib, importez le module 'cm' de matplotlib.
- Donnez la commande 'dir(cm)' et exécutez la cellule.
- Une liste de tous les colormaps disponibles sera affichée.

```
[22]: from matplotlib import cm

[23]: dir(cm)

[23]: ['Accent',
       'Accent_r',
       'Blues',
       'Blues_r',
       'BrBG',
       'BrBG_r',
       'BuGn',
       'BuGn_r',
       'BuPu',
       'BuPu_r',
       'CMRmap',
       'CMRmap_r',
       'Dark2',
       'Dark2_r',
       'GnBu',
       'GnBu_r',
       'Greens',
       'Greens_r',
       'Greys',
       'Greys_r',
       'LUTSIZE',
       'MutableMapping',
```

Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 À propos de Matplotlib

2 Importation de Matplotlib

3 Tracer des graphiques linéaires

4 Titre, étiquettes et légende

5 Tracer des histogrammes

6 Tracer des diagrammes à barres

7 Tracer des diagrammes circulaires

8 Tracer des diagrammes de dispersion

9 Tracer des graphiques logarithmiques

10 Travailler avec différents types de données

11 Variables

12 Opérateurs Arithmétiques

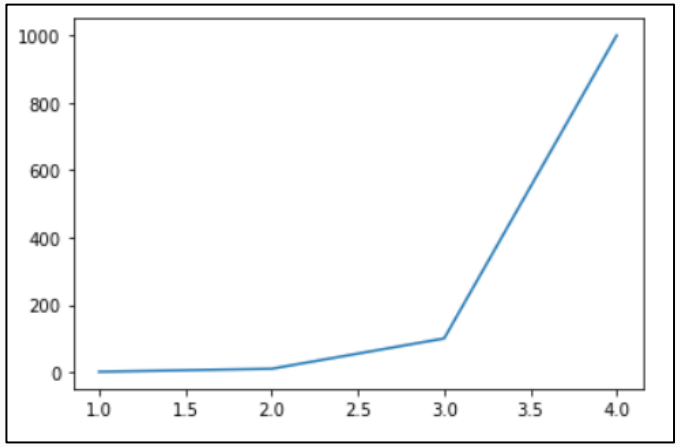
Meta Brains

Tracer des graphiques logarithmiques

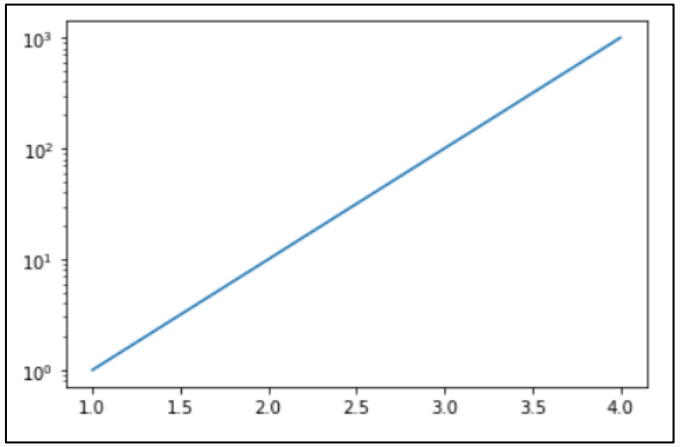
- Nous pouvons également tracer sur l'échelle logarithmique de l'axe des y en utilisant la fonction .yscale() et en passant "log" comme argument.
- L'échelle logarithmique est généralement utilisée si un ou plusieurs points de données sont beaucoup plus grands que la majorité des données, ce qui donne un graphique biaisé.

```
[24]: Axe_x = [1, 2, 3, 4]
      Axe_y = [1, 10, 100, 1000]

      plt.yscale('log')
      plt.plot(Axe_x, Axe_y)
```



Échelle linéaire



Échelle logarithmique

Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 À propos de Matplotlib

2 Importation de Matplotlib

3 Tracer des graphiques linéaires

4 Titre, étiquettes et légende

5 Tracer des histogrammes

6 Tracer des diagrammes à barres

7 Tracer des diagrammes circulaires

8 Tracer des diagrammes de dispersion

9 Tracer des graphiques logarithmiques

10 Tracer des graphiques polaires

11 Variables

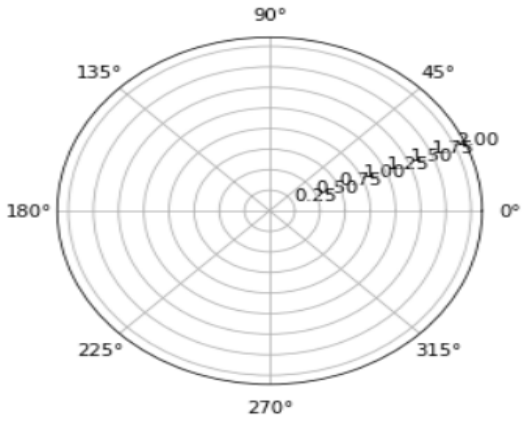
12 Opérateurs Arithmétiques

Meta Brains

Tracer des graphiques polaires

- Matplotlib nous permet également de tracer un graphique polaire.
- Un point dans un tracé polaire est représenté par (r, thêta) ;
  - r est la distance par rapport à l'origine.
  - thêta est l'angle selon lequel r est mesuré.
- Utilisez la fonction .polar() pour tracer un graphe polaire.

```
In [78]: theta = np.arange(0, (2*np.pi), 0.01) # générer un tableau de flotteurs uniformément espacés
         r = 2
         for radian in theta:
             plt.polar(radian, r)
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

Visualisation de données à l'aide de Python

1 À propos de Matplotlib

2 Importation de Matplotlib

3 Tracer des graphiques linéaires

4 Titre, étiquettes et légende

5 Tracer des histogrammes

6 Tracer des diagrammes à barres

7 Tracer des diagrammes circulaires

8 Tracer des diagrammes de dispersion

9 Tracer des graphiques logarithmiques

10 Tracer des graphiques polaires

11 Dates de Manipulation

12 Opérateurs Arithmétiques

Meta Brains

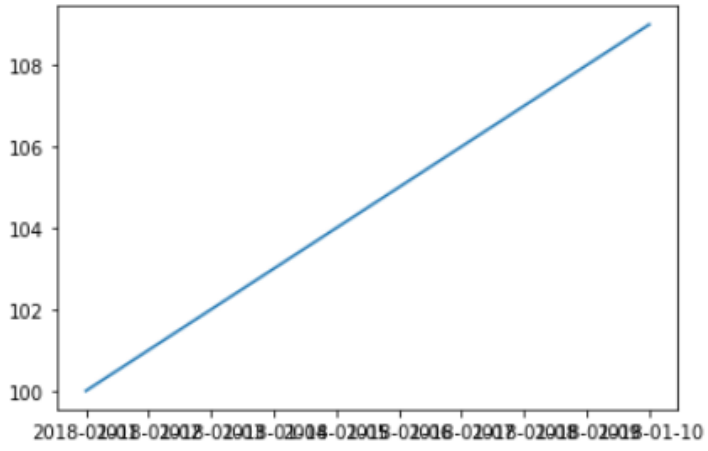
Dates de Manipulation (1/2)

- Parfois, il y a trop de valeurs sur l'axe des x et il devient difficile de les distinguer dans le graphique.
- Cela se produit également lorsque vous avez des dates sur l'axe des x et qu'elles se chevauchent, comme le montre la figure.

```
In [94]: dates = ['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04', '2018-01-05',
                  '2018-01-06', '2018-01-07', '2018-01-08', '2018-01-09', '2018-01-10']
         values = [100, 101, 102, 103, 104, 105, 106, 107, 108, 109]

         plt.plot(dates, values)

Out[94]: [<matplotlib.lines.Line2D at 0x7efef13c0d60>]
```



Meta Brains

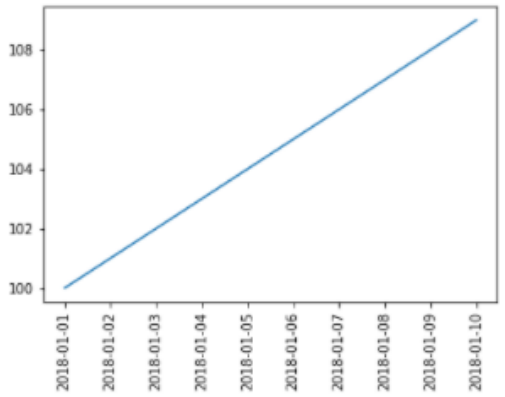
Dates de Manipulation (2/2)

- Nous pouvons éviter cela en modifiant l'orientation des valeurs sur l'axe des x à l'aide de la fonction .xticks().
  - La fonction .xticks() possède un paramètre de rotation qui peut être utilisé pour faire pivoter les valeurs sur l'axe des x d'un angle approprié.

```
[29]: Dates = ['2018-01-01','2018-01-02','2018-01-03','2018-01-04','2018-01-05',
              '2018-01-06','2018-01-07','2018-01-08','2018-01-09','2018-01-10',]
      Valeurs = [100, 101,102,103,104,105,106,107,108,109]

      plt.xticks(rotation=90)
      plt.plot(Dates, Valeurs)

[29]: [<matplotlib.lines.Line2D at 0x1c92ad37370>]
```



Meta Brains

Remise à niveau rapide sur Python

Data Science avec Python

Structures de données Pandas

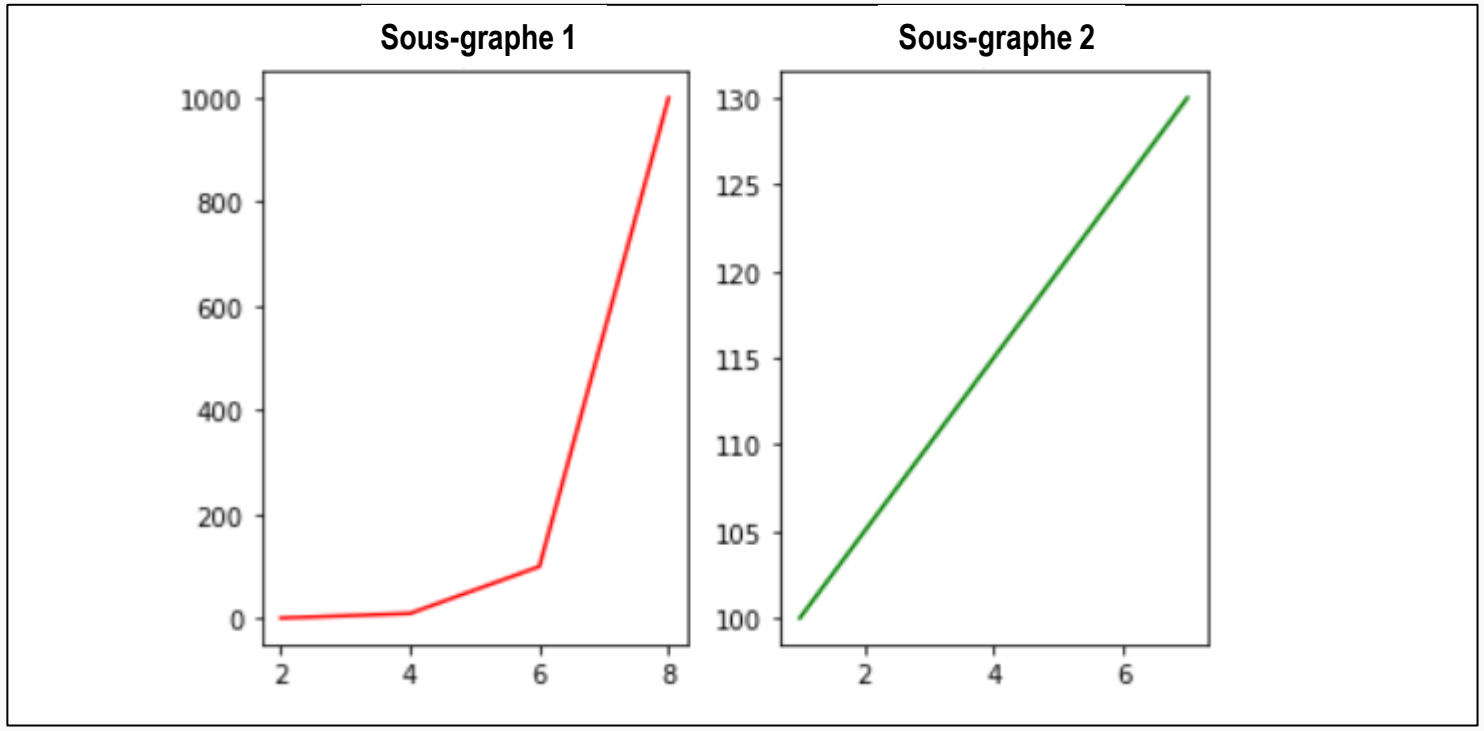
Visualisation de données à l'aide de Python

1. À propos de Matplotlib
2. Importation de Matplotlib
3. Tracer des graphiques linéaires
4. Titre, étiquettes et légende
5. Tracer des histogrammes
6. Tracer des diagrammes à barres
7. Tracer des diagrammes circulaires
8. Tracer des diagrammes de dispersion
9. Tracer des graphiques logarithmiques
10. Tracer des graphiques polaires
11. Dates de Manipulation
12. Création de plusieurs tracés dans une figure

Meta Brains

Création de plusieurs sous-graphes dans une figure(1/3)

- Nous avons vu comment nous pouvions tracer plusieurs parcelles sur le même graphique, mais parfois nous aimerions avoir des graphiques différents pour chaque parcelle.
- Ce que nous voulons en fait, c'est avoir plusieurs sous-graphes dans la même figure, comme le montre la figure donnée.



Meta Brains

Création de plusieurs sous-graphes dans une figure(2/3)

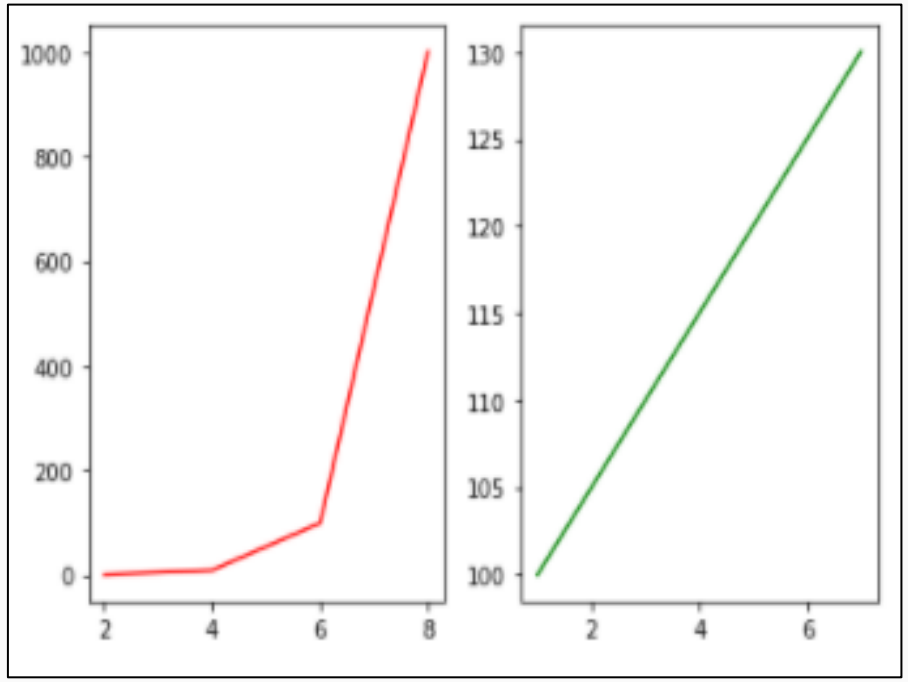
- Pour créer des sous-graphes, nous utilisons la fonction .subplot() avant de tracer chaque graphique.
- La fonction .subplot() prend 3 paramètres ;
  - Le premier paramètre est le nombre de lignes que vous voulez avoir dans votre figure.
  - Le deuxième paramètre est le nombre de colonnes que vous voulez avoir dans votre figure.
  - Le troisième paramètre est l'id/position du graphique dans la figure.

```
[30]: Axe_x1 = [2, 4, 6, 8]
      Axe_y1 = [1, 10, 100, 1000]
      Axe_x2 = [1, 3, 5, 7]
      Axe_y2 = [100, 110, 120, 130]

      plt.subplot(1, 2, 1)
      plt.plot(Axe_x1, Axe_y1, 'r')

      plt.subplot(1, 2, 2)
      plt.plot(Axe_x2, Axe_y2, 'g')

      plt.tight_layout()
```



Meta Brains

Création de plusieurs sous-graphes dans une figure (3/3)

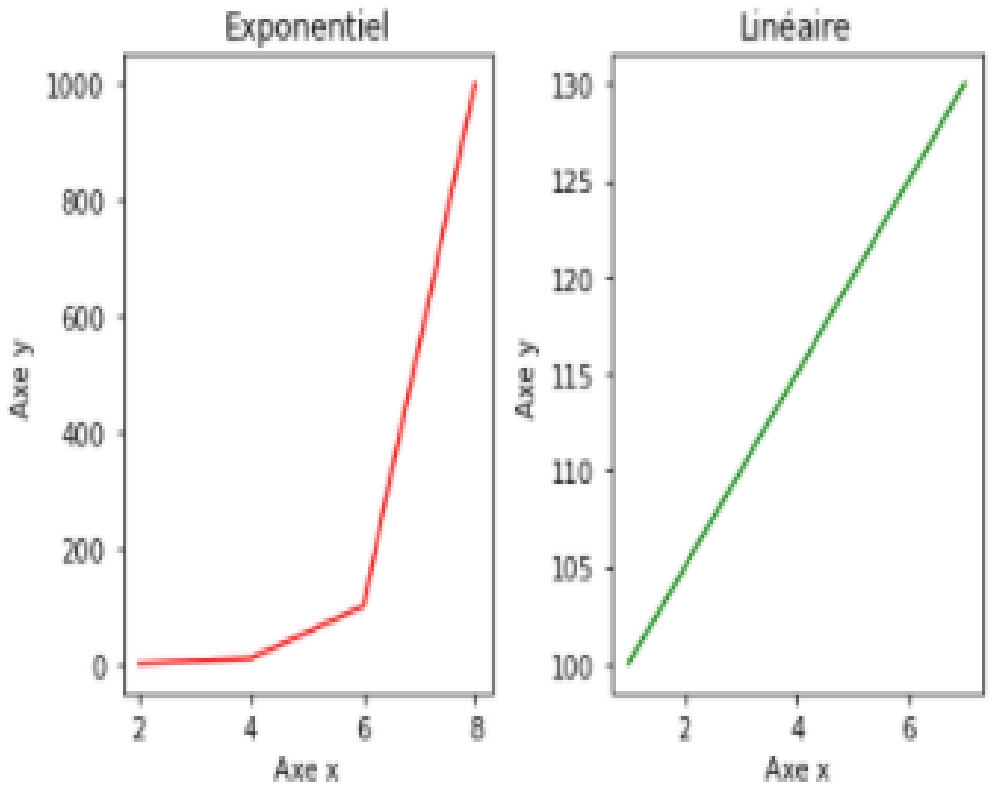
- Nous pouvons avoir des étiquettes x et y distinctes ainsi qu'un titre pour chaque sous-graphe de la figure.

```
[32]: Axe_x1 = [2, 4, 6, 8]
      Axe_y1 = [1, 10, 100, 1000]
      Axe_x2 = [1, 3, 5, 7]
      Axe_y2 = [100, 110, 120, 130]

      plt.subplot(1, 2, 1)
      plt.title('Exponentiel')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.plot(Axe_x1, Axe_y1, 'r')

      plt.subplot(1, 2, 2)
      plt.title('Linéaire')
      plt.xlabel('Axe x')
      plt.ylabel('Axe y')
      plt.plot(Axe_x2, Axe_y2, 'g')

      plt.tight_layout()
```



Meta Brains

Ressources

- https://www.w3schools.com/python/matplotlib_intro.asp

Meta Brains