Python 繪圖：從函數圖形到比較定理的應用

林貫原

January 8, 2024

當談到在 Python 中繪製數學式的圖形時，我們進入了一個極具啟發性和有趣的領域。本文將引導您進入這個精彩的世界，展示如何使用 Python 的數學計算和視覺化庫，例如 NumPy 和 Matplotlib，來實現這一目標。我們將深入探討如何將數學式轉化為程式碼，並生成漂亮的圖形，從簡單的方程式到更複雜的數學模型都將得以呈現。這不僅是數學愛好者和教育者的理想工具，還可應用於科學研究、數據分析和工程領域。無論您是對數學充滿熱情，還是需要可視化數學數據，本文將為您提供必要的知識和技巧，幫助您開始在 Python 中繪製令人驚嘆的數學式圖形。讓我們一同探索這個令人振奮的領域，展現數學和編程的完美結合。

1 PYTHON 函數繪圖

1.1 sin 函數

本小節將由介紹如何繪製函數圖形開始，首先以 sin 函數為例。我們將詳細介紹相關程式碼的含義，一旦您掌握了基本概念，後面出現的重複程式碼將不再進行詳細介紹。當面對到很複雜的函數時，若想討論他的極限是多少的話，當然可以運用到微積分的方式去算出來，但若不想動手算，看圖形做判斷有時也是一個很快能得到答案的方法之一。三角函數在數學和物理中具有廣泛的應用，因此對於理解它們的性質和特點非常重要。通過觀察和繪製這些函數的圖形，我們能夠直觀地掌握它們的週期性、震盪和極值等特性。這些特性對於分析和解決實際問題，尤其是在科學和工程領域，都具有重要價值。以下為三個有關三角函數 sin 的函數式：

$$f(x) = \frac{sin(x)}{x}, -3\pi \leq x \leq 3\pi \tag{1}$$

$$f(x) = \frac{sin(x^2)}{x}, -3\pi \leq x \leq 3\pi \tag{2}$$

$$f(x) = \frac{sin^2(2x)}{x^2}, -3\pi \leq x \leq 3\pi \tag{3}$$

若想知道這三個函數當 x 趨近於 0 的時候，函數值會是多少的話，可以運用羅必達法則做運算，計算過程如下：

$$\begin{aligned}
\lim_{x\to 0} f(x) &= \lim_{x\to 0} \frac{sin(x)}{x} \\
&= \lim_{x\to 0} \frac{\frac{\mathrm{d}}{\mathrm{d}x} sin(x)}{\frac{\mathrm{d}}{\mathrm{d}x} x} \\
&= \lim_{x\to 0} \frac{cos(x)}{1} \\
&= \lim_{x\to 0} cos(x) \\
&= 1
\end{aligned}$$

$$\begin{aligned}
\lim_{x\to 0} g(x) &= \lim_{x\to 0} \frac{sin(x^2)}{x} \\
&= \lim_{x\to 0} \frac{\frac{\mathrm{d}}{\mathrm{d}x} sin(x^2)}{\frac{\mathrm{d}}{\mathrm{d}x} x} \\
&= \lim_{x\to 0} \frac{cos(x^2)\cdot 2x}{1} \\
&= \lim_{x\to 0} cos(x^2)\cdot 2x \\
&= \frac{1\cdot 0}{1} \\
&= 0
\end{aligned}$$

$$\begin{aligned}
\lim_{x\to 0} h(x) &= \lim_{x\to 0} \frac{sin^2(2x)}{x^2} \\
&= \lim_{x\to 0} \frac{\frac{\mathrm{d}}{\mathrm{d}x} sin^2(2x)}{\frac{\mathrm{d}}{\mathrm{d}x} x^2} \\
&= \lim_{x\to 0} \frac{2sin(2x)\cdot cos(2x)\cdot 2}{2x} \\
&= \lim_{x\to 0} \frac{\frac{\mathrm{d}}{\mathrm{d}x} 4sin(2x)\cdot cos(2x)}{\frac{\mathrm{d}}{\mathrm{d}x} 2x} \\
&= \lim_{x\to 0} \frac{8cos^2(2x) - 8sin^2(2x)\cdot 2}{2} \\
&= \frac{8\cdot 1 - 8\cdot 0}{2} \\
&= \frac{8}{2} \\
&= 4
\end{aligned}$$

但做計算往往對於一些數學能力較不好的人會吃力一些，因此也可以看圖來做判斷，由圖 1 可清楚發現到當 x 趨近於 0 的時候，函數 (1) 會等於 1；由圖 2 可清楚發現到當 x 趨近於 0 的時候，函數 (2) 會等於 0；由圖 3 可清楚發現到當 x 趨近於 0 的時候，函數 (3) 會等於 4。寫成數學式的結果就是：

$$\lim_{x\to 0} f(x) = 1, \lim_{x\to 0} g(x) = 0, \lim_{x\to 0} h(x) = 4$$

至於圖 4 中呈現了三個不同函數 $f(x)$、$g(x)$ 和 $h(x)$ 的圖形，它們在相同的 x 範圍內繪製，有助於比較和理解不同函數的行為。每個函數都具有不同的形狀和特徵，包括峰值、奇點[1]和週期性振蕩。圖中的 x 軸標記使用了 π 的倍數，幫助我們觀察到週期性的性質。

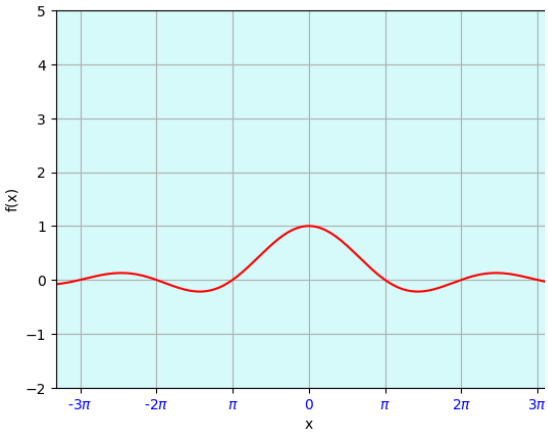


圖 1: $f(x) = \frac{sin(x)}{x}$

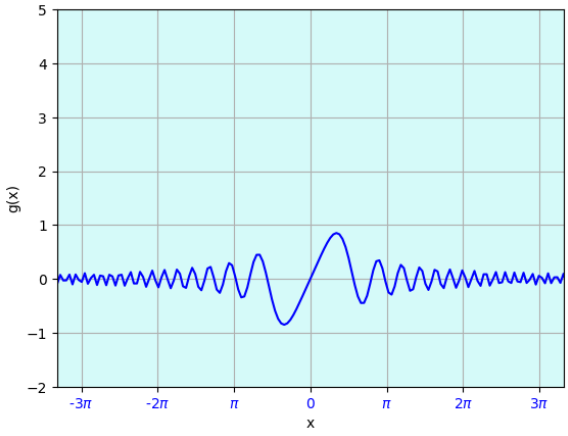


圖 2: $g(x) = \frac{sin(x^2)}{x}$

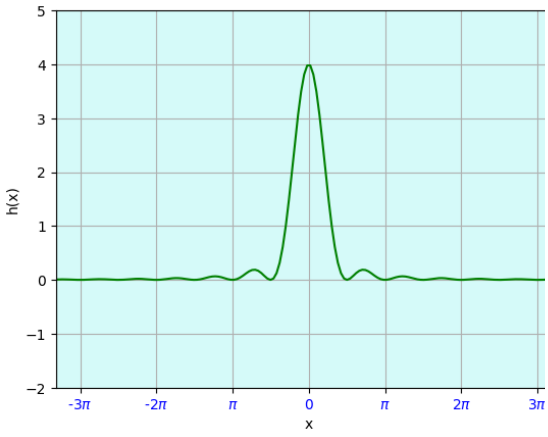


圖 3: $h(x) = \frac{sin^2(2x)}{x^2}$

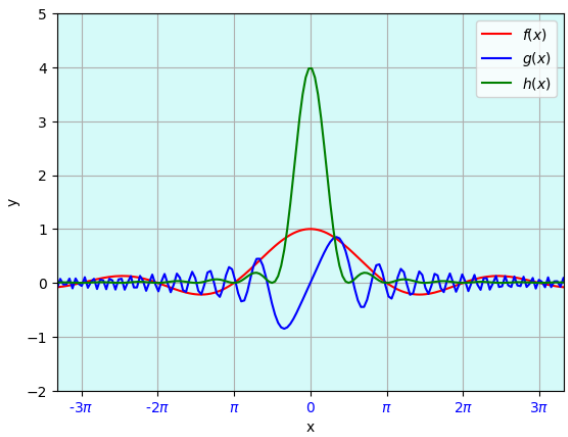


圖 4: 組合 $f(x)$ $g(x)$ $h(x)$

[1]在函數或曲線上的特殊點，其特點是在該點附近，函數的數學性質出現異常或不連續。

以下為圖 1 的程式碼與其介紹：

```
#  導入NumPy庫，並將其別名設為np。NumPy是用於數值計算
   的。
import numpy as np
#  導入Matplotlib的pyplot模組，並將其別名設為plt。
   Matplotlib是用於繪圖的。
import matplotlib.pyplot as plt
#  創建一個名為x的NumPy數組，其中包含了從-3π到3π的200個
   等間隔數值。這將用作x軸的範圍。
x=np.linspace(-3*np.pi,3*np.pi,200)
#  創建一個名為y的函數，要注意sin要改成np.sin
y=np.sin(x)/x
#  創建一個Figure對象fig和一個Axes對象ax。這將是繪圖的主
   要容器。括號內的數字代表子圖數量
fig,ax=plt.subplots()
#  在Axes對象ax上繪製一條曲線，x軸為x，y軸為y，曲線的顏
   色為紅色。
ax.plot(x,y,color="r")
#  設置x軸的刻度位置
ax.set_xticks(np.array([-3,-2,-1,0,1,2,3])*np.pi)
#  設置x軸的刻度標籤，同時設置字體大小為10，標籤顏色為藍
   色。
ax.set_xticklabels(
    ["-3$\pi$","-2$\pi$","$\pi$","0","$\pi$","2$\pi$","3
       $\pi$"],
    fontsize=10,
    color="b",)
#  設定x軸與y軸的標籤
plt.xlabel("x")
plt.ylabel("y")
#  設置背景顏色為淡藍色。
ax.set_facecolor('#D5FAF9')
#  設置y軸的範圍為-2到5
ax.set_ylim([-2,5])
#  啟用網格線
ax.grid(True)
#  顯示繪圖
plt.show()
###要注意，若繪圖的時候想要更改背景顏色，但繪圖時都沒有
   使用到ax，此時就需要一次加入這兩行。
ax=plt.gca()
ax.set_facecolor('#D5FAF9')
#  至於要查詢顏色代碼可以參考此網站
https://www.toodoo.com/db/color.html
```

1.2 指數函數

指數函數 (Exponential Function) 是一種在數學和科學領域中極為重要的特殊函數，其中 e 代表自然對數的底數，其數值約為 2.71828。指數函數廣泛運用於多個領域，包括概率論和統計學，用於描述時間間隔和概率分佈；微積分，作為導數和積分的關鍵函數；複利計算，特別在金融領域；物理學，描述放射性衰變、波動現象和電路行為；以及工程學和計算機科學，用於建模和解決控制系統、數據分析和模擬等各種問題。指數函數的廣泛應用使其成為自然科學和工程領域中不可或缺的數學工具。以下為一個有關指數的函數：

$$f(x) = \frac{e^{\alpha x}}{e^{\alpha x} + 1} \tag{4}$$

若要探討當 x 趨近於無限大或負無限大的值，當然可以運用前一節所展示的羅必達法則做運算，在此就不再多做介紹，我們一樣運用判別圖形的技巧做說明。圖 5 為當 $\alpha = 1$ 的時候函數 (4) 的圖形，由圖中可清楚發現到 $f(x)$ 為 S 型曲線，函數值變化較明顯的區間大概在 $-2 < x < 2$，兩端較趨緩，紅色虛線為漸近線，當 x 趨近於無限大的時候，$f(x)$ 會等於 1，而當 x 趨近於負無限大的時候，$f(x)$ 會等於 0。寫成數學式的結果就是：

$$\lim_{x\to\infty} f(x) = 1, \lim_{x\to-\infty} f(x) = 0$$

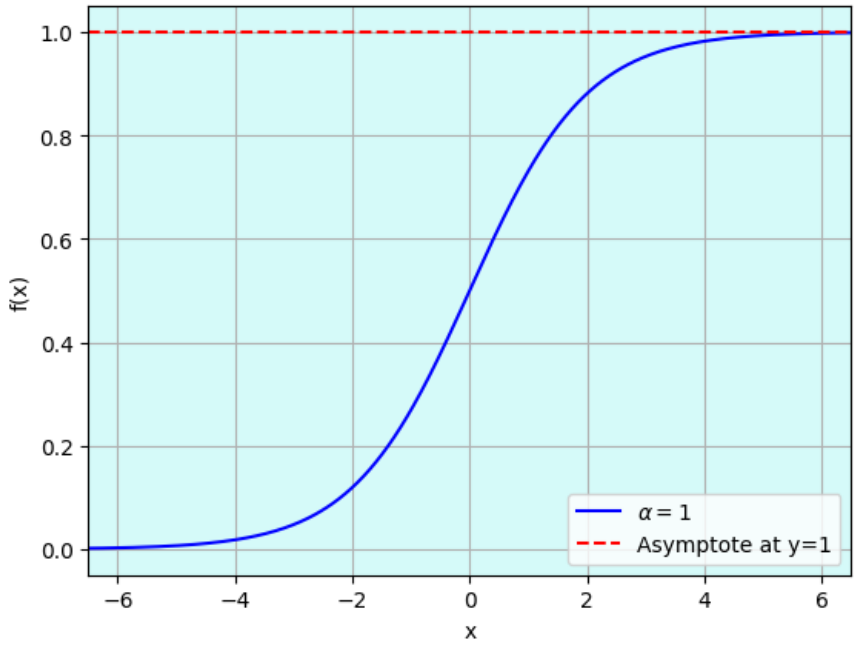


圖 5: $f(x) = \frac{e^{\alpha x}}{e^{\alpha x}+1}, \alpha = 1$

圖 6 呈現了 $\alpha = 1, 2, ..., 10$ 的函數 (4) 圖形，可從圖中發現以下幾項特點：

1. 每個曲線對應不同的 α 值。函數 (4) 中的 α 參數影響 S 型曲線的陡峭度，隨著 α 值的增加，S 型曲線會顯得越陡峭。
2. 當 x 增加時，該函數開始緩慢上升，並且隨著 x 趨近正無窮時，逐漸趨近 1。它們也在 x 趨近負無窮時逐漸趨近 0。
3. 紅色虛線代表 $y = 1$ 處的漸近線，這個漸近線對於所有的 S 型曲線都是相同的。

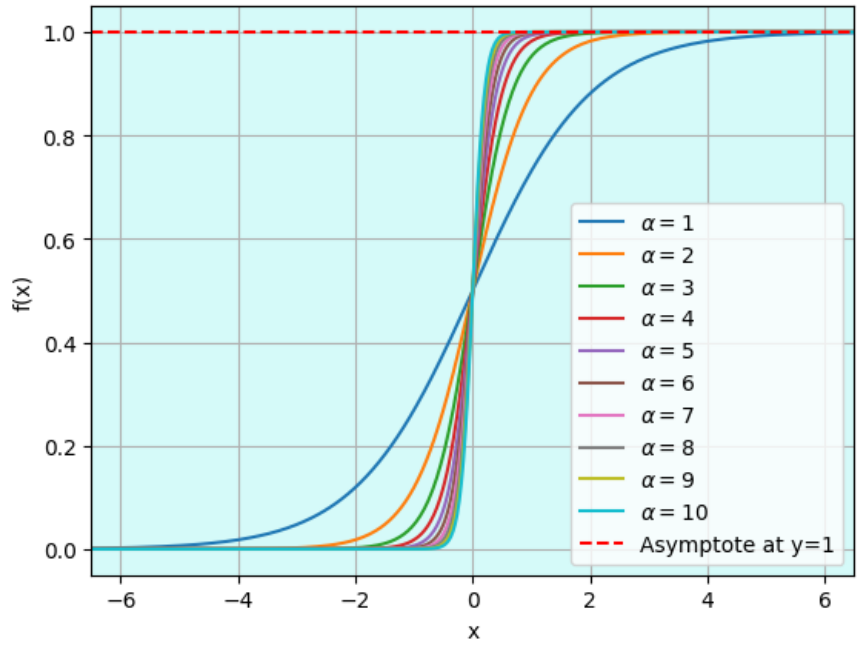


圖 6: $f(x) = \frac{e^{\alpha x}}{e^{\alpha x}+1}$

此節較特殊的程式碼：

```
#  定義函數
def sigmoid(x,alpha):
    return(np.exp(alpha*x))/(np.exp(alpha*x)+1)
#  設定alpha值範圍
alphas=range(1,11)
#  繪製不同alpha值的函數
for alpha in alphas:
    y=sigmoid(x,alpha)
    plt.plot(x,y,label=f'$\\alpha ={alpha}$')
#  繪製y=1漸近線並以虛線表示
plt.axhline(1,color='red',linestyle='--',label='
   Asymptote at y=1')
```

1.3 指數與 sin 聯合函數

以下為一個指數與 sin 的聯合函數：

$$f(x) = e^{\frac{-x}{10}} sin(x) \tag{5}$$

圖 7 為函數 (5) 的圖形，圖中呈現的是一個波形，由兩個不同的函數組成：指數衰減和正弦波，紅點為每個波的相對極大值，綠點為每個波的相對極小值。這個波形主要特點在於振蕩行為，振幅隨著 x 值增加逐漸減小。振幅的減小是由指數衰減項所引起的，該項隨著 x 值的增加而快速減小。同時，正弦波成分讓波形呈現週期性的變化，使其在 x 軸周圍進行振盪。整個波形的振幅變化可以清晰地在 y 軸範圍內觀察到。可參考圖 8 來觀察函數 (5) 的變化。

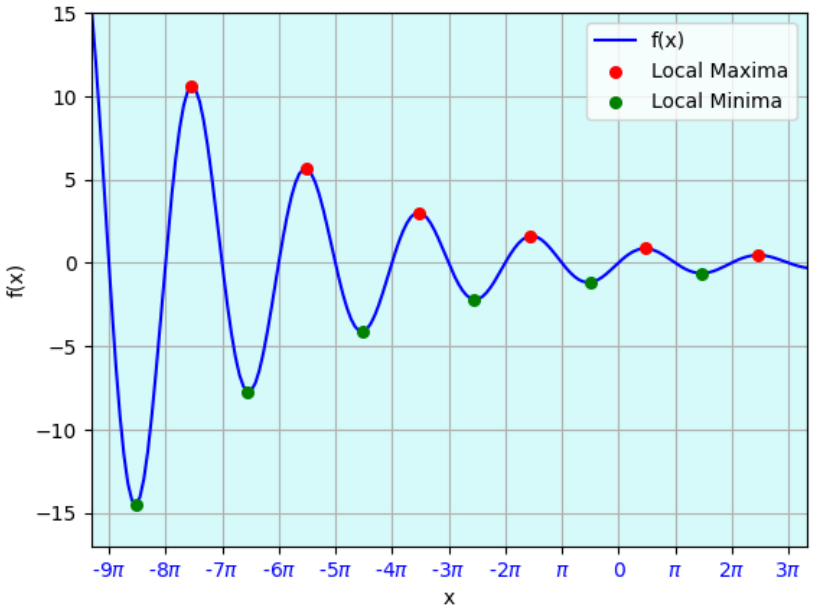
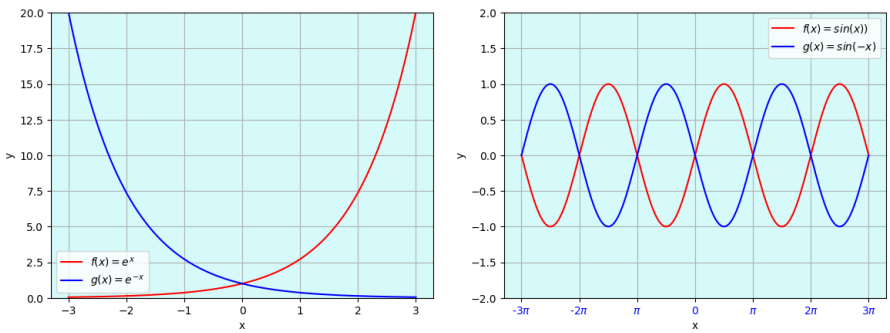


圖 7: $f(x) = e^{\frac{-x}{10}} sin(x)$



(a) exponential (b) sine

圖 8: 指數與 sin 函數

圖 9 將這些代表相對極大值的紅點與代表相對極小值的綠點用曲線做連接，由此可更輕易判斷 $f(x)$ 隨著 x 值得增加，震盪幅度逐漸縮小的趨勢。

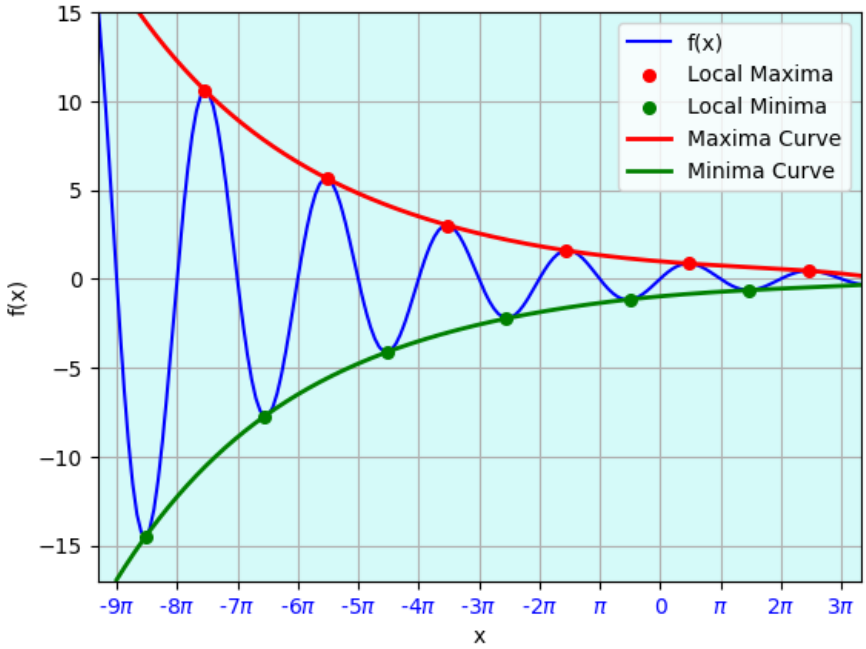


圖 9: 震盪幅度逐漸縮小

此節較特殊的程式碼：

```
#  這行代碼導入了SciPy函式庫中的argrelextrema函數，用於
   找到一個數組中的極值。
from scipy.signal import argrelextrema
#  這行代碼使用argrelextrema函數找到了函數y中的相對極大
   值。它使用了np.greater作為比較函數，以確定哪些值被視為
   極大值。
local_maxima_indices = argrelextrema(y, np.greater)
#  這兩行將找到的相對極大值的x和y值提取出來，分別存儲在
   local_maxima_x和local_maxima_y中。
local_maxima_x = x[local_maxima_indices]
local_maxima_y = y[local_maxima_indices]
#  這行代碼使用散點圖將相對極大值的位置標示在圖上。指定
   了顏色、標籤以及點的大小（s=30）。zorder參數用於指定圖
   層順序，確保這些點在圖形的最上層。
ax.scatter(local_maxima_x, local_maxima_y, color='r',
   label='Local Maxima', s=30, zorder=2)
#  跟剛剛步驟很類似，現在使用np.less尋找相對極小值
local_minima_indices = argrelextrema(y, np.less)
#  連接紅點繪畫曲線
ax.plot(x_curve, y_maxima_curve, color='r', label='
   Maxima Curve', linestyle='-', linewidth=2)
```

1.4 分數函數

當 x 在分母中出現的函數在數學中有一些特點，特別是當分母趨近於 0 的時候，此時該函數特點通常與極限和連續性有關。以下為一個有關 x 在分母的分數函數：

$$f(x) = \frac{1}{x-2} \tag{6}$$

圖 10 呈現函數 (6) 的圖形，圖中呈現的是一個雙曲線，紅色虛線表示 $x = 2$ 時的垂直漸近線，發現當 x 從右邊趨近 2 時趨近正無窮大，並在 x 從左邊趨近 2 時趨近負無窮大。圖 10 故意避免在 $x = 2$ 處繪製函數圖形，因為這一點上函數是未定義的，分母是不可為 0 的。

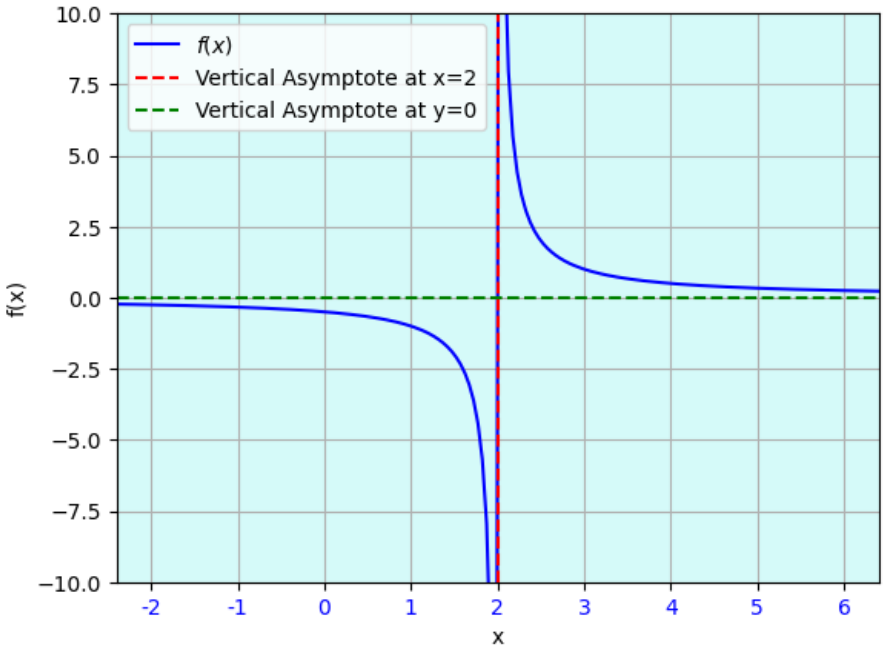


圖 10: $f(x) = \frac{1}{x-2}$ with Vertical Asymptote at $x = 2$

此節較特殊的程式碼：

```
#  這行程式碼使用NumPy函式setdiff1d創建一個新的x值數組
   x_no_2，該數組排除了x=2的值。這是為了避免分母為零，因
   為在x=2時，分母等於零，這會導致無定義的情況。
x_no_2=np.setdiff1d(x,[2])
y_no_2=1/(x_no_2-2)
#  這行程式碼添加了一條垂直的虛線，表示垂直漸近線在x=2的
   位置。這條線的顏色是紅色，並且使用虛線樣式。同時，它的
   標籤說明了這是在 x=2 的位置的垂直漸近線。
ax.axvline(x=2,color='red',linestyle='--', label='
   Vertical Asymptote at x=2')
```

1.5 反函數

反函數是一個與原始函數相關的特殊函數，它們彼此互為逆操作。這表示當你應用反函數於原函數的輸出時，你會得到原函數的輸入。反函數的圖形與原函數的圖形有關於 $y = x$ 線對稱。然而，並非所有函數都有反函數，原函數必須是一對一的且映射到整個定義域。反函數的符號通常是原函數的名稱後面加上上標”-1”，例如，如果原函數是 $f(x)$，反函數為 $f^{-1)}(x)$。以下為一個函數與其反函數：

$$f(x) = x^3 + 2 \tag{7}$$

$$f^{-1}(x) = \sqrt[3]{x-2} \tag{8}$$

圖 11 呈現函數 (7) 和其反函數 (8)，原始函數由藍色曲線表示，它的反函數以綠色呈現，金黃色的對稱線用來顯示其對稱性。圖表上標有兩個紅色的點，是該函數及其反函數的反曲點，分別為在 (0,2) 及 (2,0)。以下為給定原函數要算出其反函數的計算過程：

$$\begin{aligned} y = x^3 + 2 \Rightarrow y - 2 &= x^3 \\ \Rightarrow x &= \sqrt[3]{y-2} \\ \therefore f^{-1}(x) &= \sqrt[3]{y-2} \end{aligned}$$

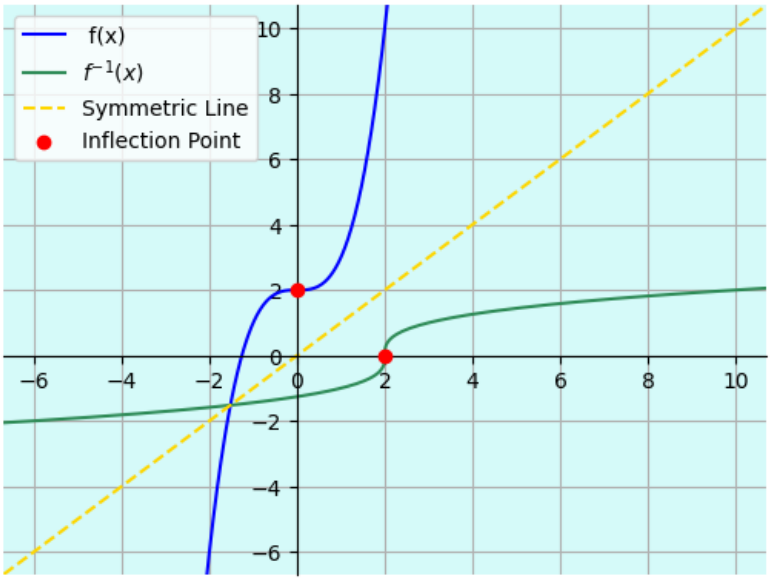


圖 11: $f(x) = x^3 + 2$ 與其反函數

此節較特殊的程式碼：

```
#  定義函數f(x)
def f(x):
    return x**3 + 2
#  產生x範圍
x1 = np.linspace(-2, 2, 400)
x2 = np.linspace(-6, 10, 400)
#  計算x1中每個值的f(x)值，並將結果存儲在y1中。
y1 = f(x1)
#  計算x1中每個值的反函數值，將結果存儲在y_inv中。
y_inv = x1**3 + 2
#  繪製反函數 f^(-1)(x)
plt.plot(y_inv, x1, label="$f^{-1}(x)$", color="#2E8B57"
    )
#  獲取當前的坐標軸對象。
ax=plt.gca()
#  設置了左側坐標軸的位置。通過('data',0)，這意味著左側
    坐標軸將穿過x=0這一點。
ax.spines['left'].set_position(('data',0))
#  設置了底部坐標軸的位置，是在x軸的數值0位置。這確保底
    部坐標軸穿過y=0這一點。
ax.spines['bottom'].set_position(('data',0))
#  設置了頂部坐標軸的可見性，將頂部的坐標軸設置為不可
    見，也就是隱藏它。
ax.spines['top'].set_visible(False) # take off top line
#  設置了右側坐標軸的可見性，將右側的坐標軸設置為不可
    見，也就是隱藏它。
ax.spines['right'].set_visible(False)
```

1.6 高斯分配

高斯分配，又稱正態分配、常態分配，是統計學中最常見的概率分布之一，常用來描述許多自然現象中的隨機變數。它的特點是數據點在平均值周圍呈現出典型的鐘形曲線分布。這個分布由兩個關鍵參數，平均數 (μ) 和方差 (σ^2)，所定義，這兩個參數分別決定了曲線的中心位置和形狀。高斯分布符合中央極限定理，適用於多個領域，並具有經典的 68-95-99.7 法則，描述數據分佈在不同標準差範圍內的比例。這種分布在統計分析、數據建模和自然科學中起著關鍵作用。以下為一個高斯分配的函數：

$$\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-3)^2}{2}} \tag{9}$$

仔細觀察函數 (9)，可發現其服從 $N(3,1)$，只要是常態分配就擁有 68-95-99.7 法則，又被稱為" 三個標準差法則"，這個法則告訴我們，在正態分佈的數據中，

大約 68% 的數據點分佈在均值（平均值）的一個標準差範圍內，約 95% 分佈在兩個標準差範圍內，而約 99.7% 分佈在三個標準差範圍內。這反映了正態分佈的特點，其中數據點相對於平均數的分散程度由標準差來衡量。具體來說，均值是正態分佈的對稱中心，且分佈在均值兩側是對稱的。標準差用於表示數據點相對於均值的離散程度，標準差小意味著數據點更接近均值，而標準差大表示數據點更分散。儘管該法則通常適用於近似正態分佈的數據，但在實際應用中，它也可作為一種便捷的方式，用來了解數據集的分佈特點，無需詳細的統計分析，但僅限於常態分配使用。以下為其用數學式作表示：

$$\Pr(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 0.682689492137086$$

$$\Pr(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 0.954499736103642$$

$$\Pr(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 0.997300203936740$$

圖 12 可看到三者比例，淺綠色區域位於 $\mu \pm \sigma$ 中，代表該區間大約佔了 68.2%；再加上黃色區域的話，其位於 $\mu \pm 2\sigma$ 中，代表該區間大約佔了 95.4%；若再加上紅色區域，其位於 $\mu \pm 3\sigma$ 中，代表該區間大約佔了 99.7%。

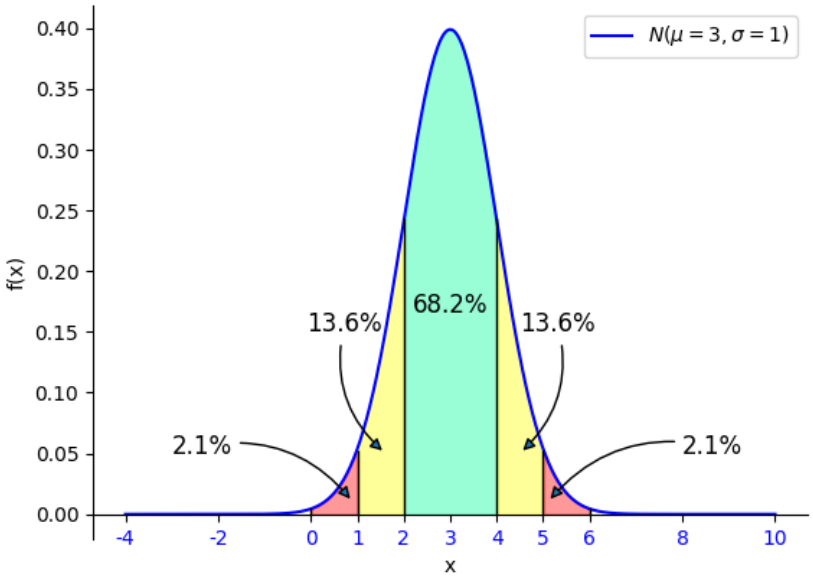


圖 12: $N(3, 1)$

此節較特殊的程式碼：

```
#  用來填充圖形區塊顏色
plt.fill_between(x, 0, y, where=(x >= mu + 2 * sigma) &
    (x <= mu + 3 * sigma), color='red', alpha=0.4)
#  箭頭拉出的標記
plt.axvline(x=mu - sigma, color='black', linestyle='-',
    linewidth=1, ymin=0.045,ymax=0.6)
```

1.7 四次函數

四次函數是一種多項式函數，它的最高次方項是 4。這種函數的特點在於它的曲線形狀可能有多個反曲點，導致出現相對極大值或相對極小值。這些反曲點可以使函數圖形有多個起伏和轉折。四次函數的對稱性取決於係數的選擇，它可以是關於 y 軸對稱的，也可以是無對稱性的。這種函數可能有多個零點，取決於係數的值，最多可以有 4 個實根。此外，四次函數的奇偶性也與係數相關，如果只包含偶次冪的係數，則為偶函數，如果只包含奇次冪的係數，則為奇函數。總的來說，四次函數的圖形可能是多彎曲的，且具有各種特點，這些特點取決於方程中的係數。

$$f(x) = 3x^3 - x^4 \tag{10}$$

圖 13 呈現函數 (10) 圖形主要變化的區間。首先，圖形形狀反映了該函數的曲線特性，有助於理解其行為。其次，圖上的 x 軸交點表示函數的零點位置。此外，相對極值可在圖上的峰值和谷底處識別，代表斜率變化和相對極大或極小值。

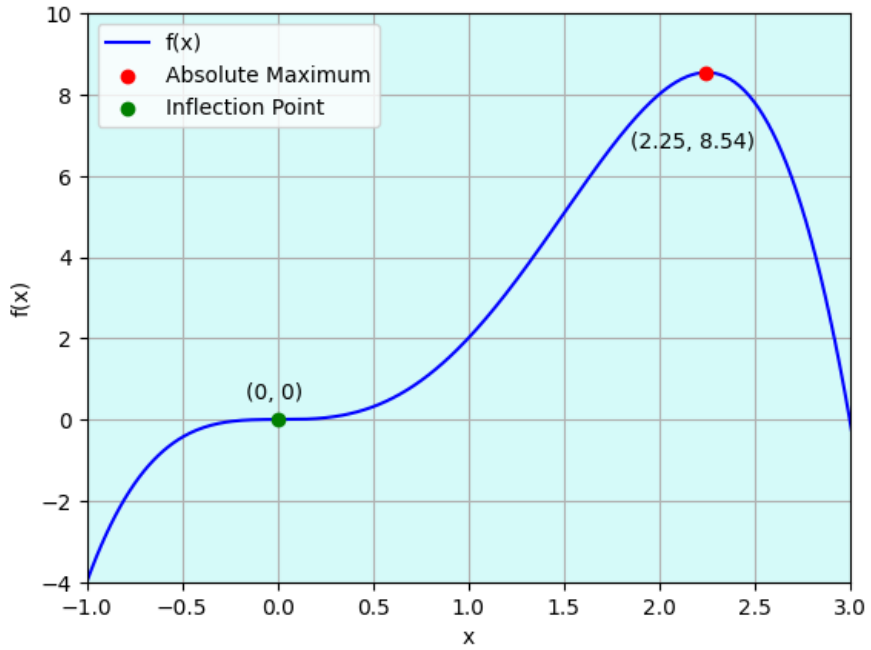


圖 13: $f(x) = 3x^3 - x^4$

```
#  用此方法先設置的是想要呈現的位置，再來才是要表達的內
   容
plt.text(1.85,6.7,(2.25,8.54),fontsize=10)
```

1.8 ln 函數

自然對數函數 (ln) 是一種以數學常數"e" 為底的對數函數，常表示為 $ln(x)$。它在數學和科學領域中扮演著重要的角色，尤其在處理指數和對數相關問題時非常有價值。自然對數函數的主要性質包括其導數等於 $1/x$，以及它可用於描述複利計算、成長、衰減和其他自然現象。此函數在微積分、統計學和工程學等學科中廣泛應用，並以其簡潔且強大的數學性質而聞名。

$$f(x) = \frac{ln(x)}{x^2} \tag{11}$$

圖 14 呈現函數 (11) 的圖形，於 (1.65,0.18) 有絕對極大值。前面小節有提到，分數函數的分母是不可為零的，不然會無意義，因此 x 不可為零，分子的部分因為 $ln(1) = 0$，所以當 $x = 0$ 的時候，其函數值為 0，而看到紅色虛線所表示的漸近線，當 x 趨近無限大的時候，其函數值會趨近於 0，至於 0 的右極限會趨近負無限大，寫成數學式則為：

$$\lim_{x\to\infty} \frac{ln(x)}{x^2} = 0, \lim_{x\to 0^+} \frac{ln(x)}{x^2} = -\infty$$

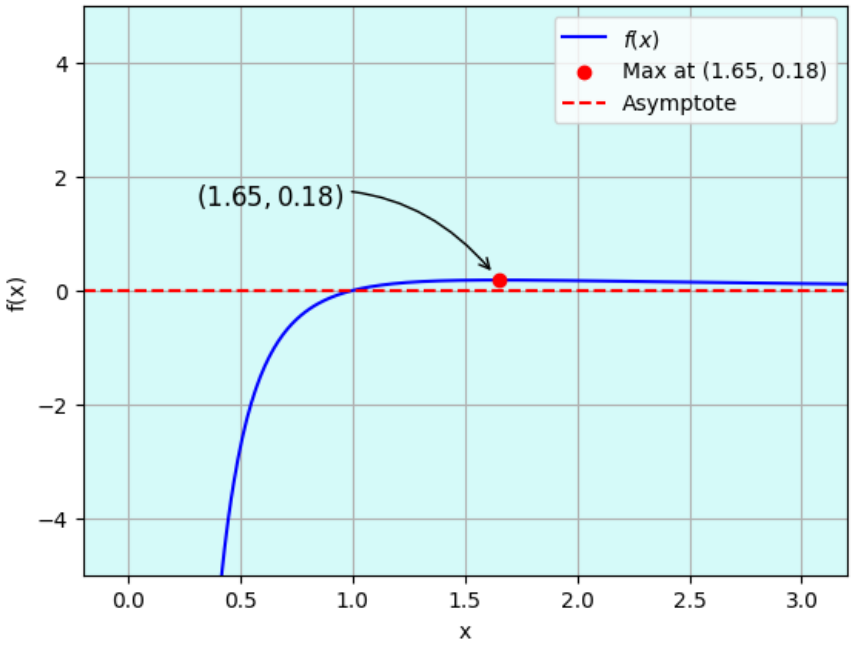


圖 14: $f(x) = \frac{ln(x)}{x^2}$

圖 15 特地將 x 座標設置在 0 到 3 之間，y 座標設置在 0 到 0.3 之間，以顯示函數 (11) 具有絕對極大值。

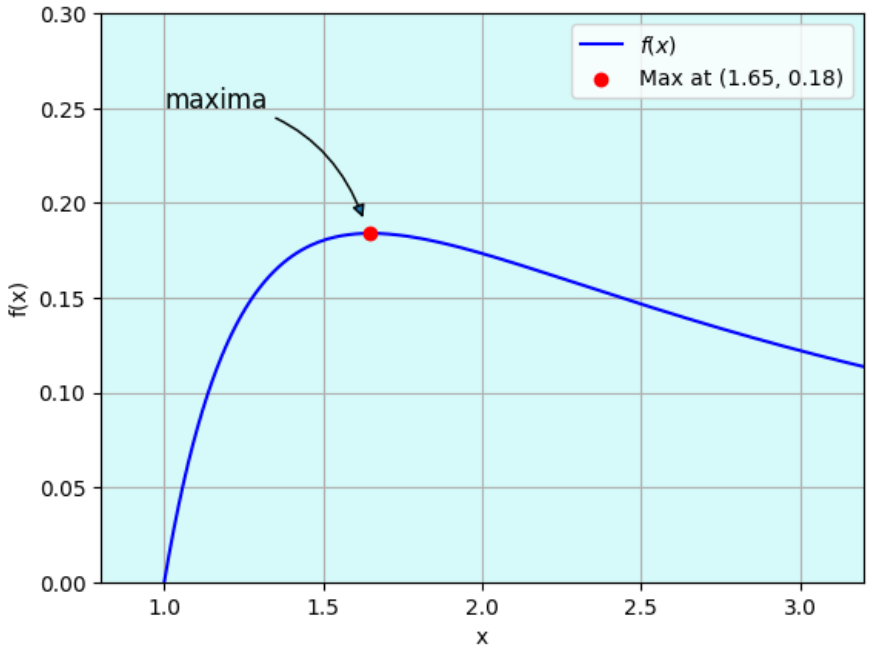


圖 15: $f(x) = \frac{ln(x)}{x^2}$

此節較特殊的程式碼：

```
#  找到最大值的座標
max_x = x[np.argmax(y)]
max_y = np.max(y)
#  在圖中點出絕對極大值
plt.scatter(max_x, max_y, color='black', label=f'Max at
    ({max_x:.2f}, {max_y:.2f})', zorder=5)
```

1.9 離散函數

離散函數其定義域是離散的，通常包括整數或有限的數值集合，而不是連續的實數集合。這些函數僅返回特定的離散數值，並以點值方式表示。它們通常是無限的，並以由離散數據點組成的圖形呈現，而不是平滑的曲線。離散函數在離散數據處理、計算機科學、數位信號處理和離散數學等領域中廣泛應用，因為這些領域需要處理非連續的數據，並使用離散函數作為有效的數學工具。以下為一離散函數：

$$f(x) = \begin{cases} 1, & 1 \leq x < 3 \\ 2, & 3 \leq x < 5 \\ 3, & 5 \leq x < 7 \end{cases} \tag{12}$$

圖 16 呈現函數 (12) 的離散圖形，會發現它由三組不同顏色的線段組成，分別為藍色、紅色和綠色。每組線段代表一種線性函數，具有特定的斜率和間隔。這些線段皆為左實右虛的，表示當函數值為 1 時，$x = 1$ 但 $x \neq 3$; 當函數值為 2 時，$x = 3$ 但 $x \neq 5$; 當函數值為 3 時，$x = 5$ 但 $x \neq 7$。

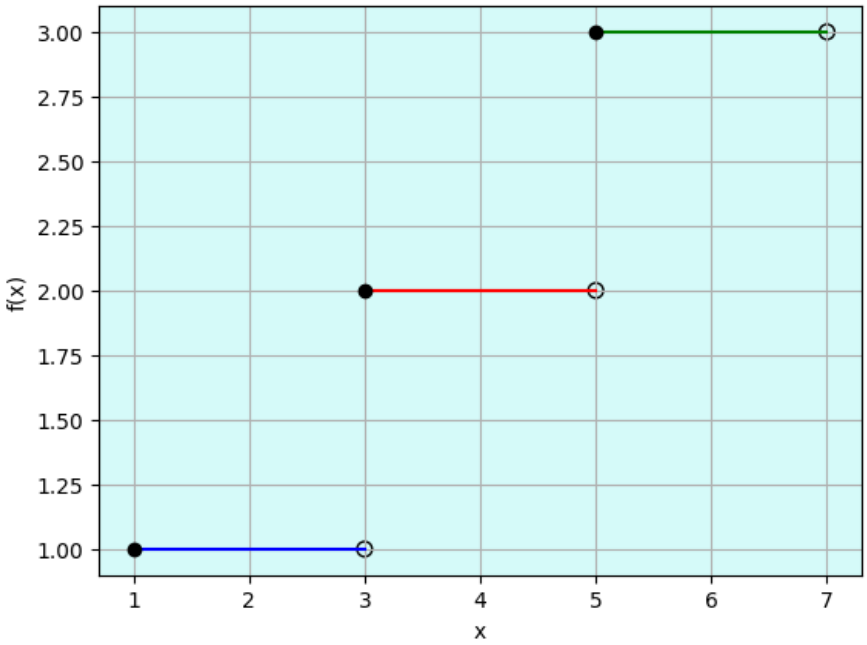


圖 16: $f(x) = \begin{cases} 1, & 1 \leq x < 3 \\ 2, & 3 \leq x < 5 \\ 3, & 5 \leq x < 7 \end{cases}$

此節較特殊的程式碼：

```
#  此段繪製平行線
plt.plot([1,3],[1,1],color='b')
plt.plot([3,5],[2,2],color='r')
plt.plot([5,7],[3,3],color='g')
#  繪製黑點於自己想要的地方
plt.scatter([1,3,5],[1,2,3], color='black', marker='o',
    zorder=5)
#  用facecolors控制其為空心的點
plt.scatter([3,5,7],[1,2,3], color='black', marker='o',
    facecolors='none' ,s=50)
```

1.10 圓形函數

式 (17) 為一圓形方程式：

$$x^2 + y^2 = 1 \tag{13}$$

於 Python 中有許多方式能夠畫圓，以下會介紹其中三種方式：

1. 將變數 x 與 y 表示為另一個變數 θ 的函數,$x = \sin\theta, y = \cos\theta,\ \ 0 \leq \theta \leq 2\pi$，圖 17 為函數 (13) 的圖形，其為半徑為 1 的圖。

   此圖較特殊的程式碼：

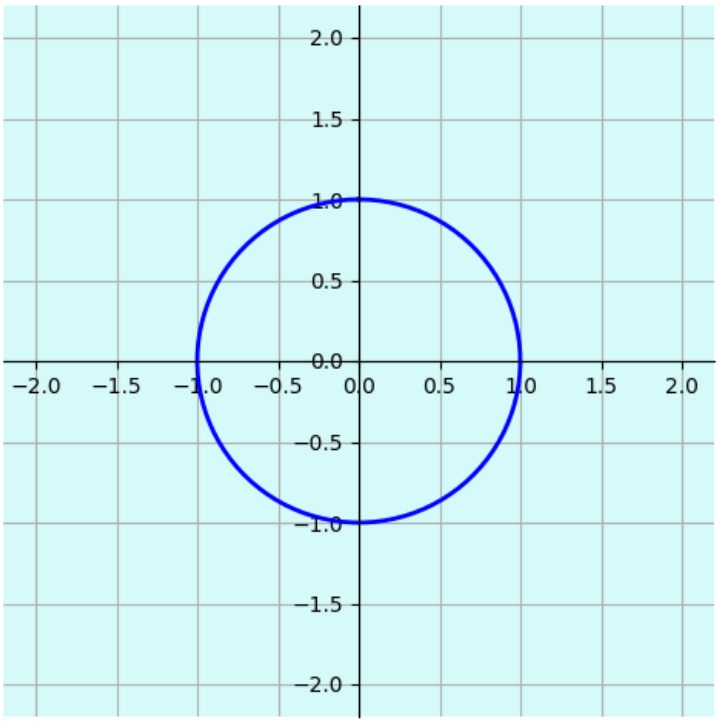


圖 17: $x^2 + y^2 = 1$

```
#  產生一組\theta值
theta=np.linspace(0,2*np.pi,100)
#  根據x=sin(\theta)和y=cos(\theta) 計算x和y座標
x=np.sin(theta)
y=np.cos(theta)
```

2. 將圓形方程式當作隱函數 (implicit function) 處裡。

   此圖較特殊的程式碼：

```
#  定義隱函數
def implicit_circle(x, y):
    return x**2+y**2-1
#  產生x和y座標的數據點
x=np.linspace(-1.5,1.5,400)
y=np.linspace(-1.5,1.5,400)
X,Y=np.meshgrid(x,y)
Z=implicit_circle(X,Y)
#  繪製圓形方程式的圖形
```

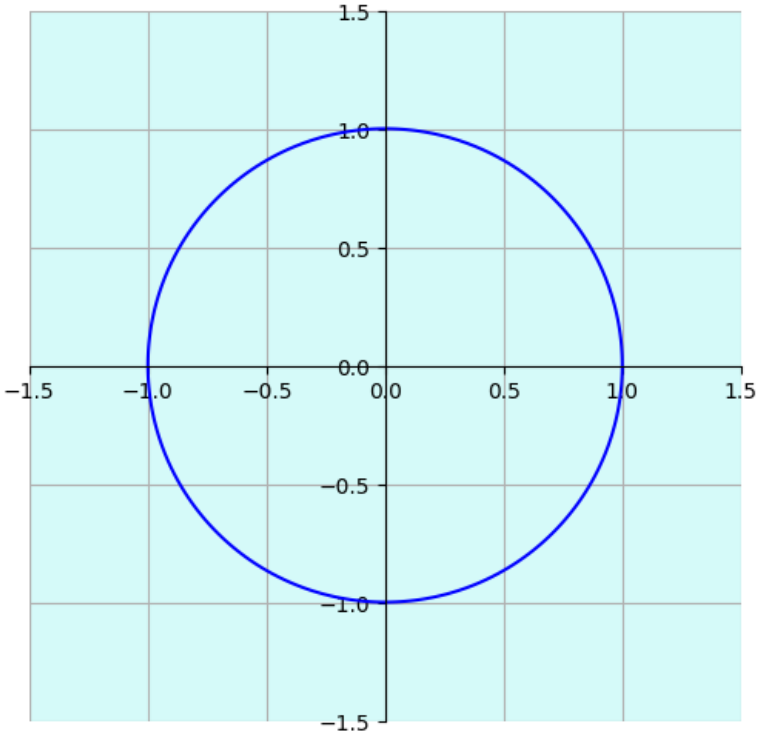


圖 18: $x^2 + y^2 = 1$

```
plt.figure(figsize=(6, 6))
#  levels=[0]表示畫出等於0的等高線
plt.contour(X,Y,Z,levels=[0],colors='b')
#  讓x和y軸的比例相同，確保圓形正確顯示
plt.gca().set_aspect('equal',adjustable='box')
```

3. 運用 matplotlib 套件中的 circle 指令。圓的通用公式為：

$$x^2 + y^2 = r^2$$

其中 r 為圓的半徑，此部分我加入了另外兩個半徑為 1.5 與 0.5 的圓並展示了對其中心上色，兩者的方程式為以下：

$$x^2 + y^2 = 1.5^2$$

$$x^2 + y^2 = 0.5^2$$

此圖較特殊的程式碼：

```
#  創建一個圓形物件。它位於原點(0,0)，半徑為1，線寬
   度為2，顏色為藍色 ，並設置不填充內部(fill=False)。
circle = plt.Circle((0, 0), 1,linewidth=2, color='b'
   , fill=False)
```

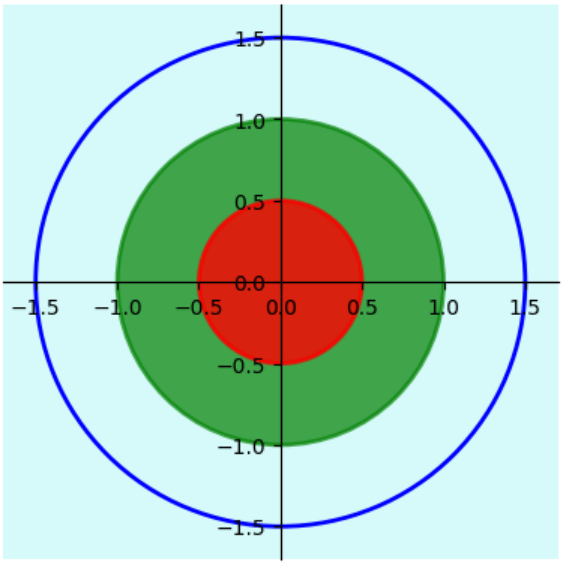


圖 19: Three Circles

```
#  把圓形添加到軸上
ax.add_artist(circle)
#  設置軸等比例縮放以確保圓看起來像一個完整的圓形。
ax.set_aspect('equal', adjustable='box')
#  將左邊及下方的軸位置設置為中心
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('center')
#  隱藏了右側和上側的軸，使圖形只保留左側和底部的
   軸，從而更突出圓的位置。
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
```

1.11 方形函數

正方形是一個具有四個相等邊和對角線相等的簡單而優雅的幾何形狀。其對稱性和均勻性使之廣泛應用於數學、藝術和工程領域。在數學中，它具有獨特的性質，面積等於邊長的平方，周長是四倍的邊長。在藝術中，正方形代表著平衡和秩序，經常被畫家和設計師用來創作圖案。在工程中，它是穩定結構的理想基礎。總之，正方形是一個簡約且多用途的形狀，具有廣泛的意義，並在不同領域中得到廣泛應用。圖 20 為一個邊長為一的正方形。

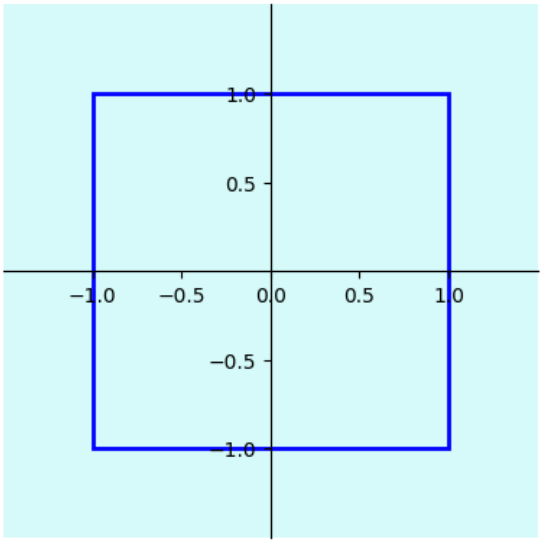


圖 20: A square of side 1

此節較特殊的程式碼：

```
#  設定正方形的中心座標為(0,0)。
square_center=(0,0)
#  設定正方形的邊長為1。
side_length=1
#  使用patches.Rectangle創建一個矩形圖形，表示正方形
square=patches.Rectangle(
    (square_center[0]-side_length/2,square_center[1]-
       side_length/2),side_length,side_length,linewidth
       =2,edgecolor='b',facecolor='none')
```

2 比較定理

比較定理 (Comparison Theorem) 是微積分中的一個重要定理，通常用於分析和比較兩個函數或級數的性質。這個定理有多個不同的變種，包括比較收斂定理和比較發散定理，它們用於研究數列、函數和積分的性質。

1. 比較收斂定理：當我們想要確定一個級數是否收斂（即總和有一個有限極限），可以使用比較定理。如果我們可以找到一個已知的收斂級數，並證明要評估的級數的每一項都小於對應的已知收斂級數的每一項，那麼我們可以得出要評估的級數也是收斂的。

2. 比較發散定理：用於確定一個級數是否發散，需找到一已知發散級數，並證明要評估的級數每項均大於對應已知發散級數的對應項。

比較定理在處理複雜級數時非常有用，因為它允許我們使用已知的結果來推斷未知級數的性質。這在計算學術、工程和物理學等領域中非常有用。以下為一個例題：設 $S_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

1. 驗證 $\lim_{n\to\infty} S_n$ 會發散。
2. 令 γ_n 表示陰影面積總和，如圖 22 所示。證明 $\gamma_n = S_n - \ln(n+1)$。
3. 驗證 $\frac{1}{2}(1 - \frac{1}{n+1}) < \gamma_n < 1$。(提示：為了顯示三者不相等，可以透過將每個陰影區域與較大的矩形和較小的三角形進行比較來完成。)

< 證明 >

1. 若要探討 S_n 的發散 (diverge) 可以嘗試改變 n，從小變大，直到很大很大，如 $n = 100, 1000, \cdots, 10^6, \cdots$ 並計算 S_n，觀察其數值的變化。

$$\sum_{k=1}^{10} \frac{1}{k} \approx 2.9290, \sum_{k=1}^{20} \frac{1}{k} \approx 3.5977, \sum_{k=1}^{30} \frac{1}{k} \approx 3.9950, \sum_{k=1}^{40} \frac{1}{k} \approx 4.2785$$

$$\sum_{k=1}^{100} \frac{1}{k} \approx 5.1874, \sum_{k=1}^{1000} \frac{1}{k} \approx 7.4855, \sum_{k=1}^{10000} \frac{1}{k} \approx 9.7876, \sum_{k=1}^{100000} \frac{1}{k} \approx 12.0901$$

由此可發現當 n 越大，S_n 的數值就越大，也可參考圖 21，可顯示 $\lim_{n\to\infty} S_n$ 是發散的。

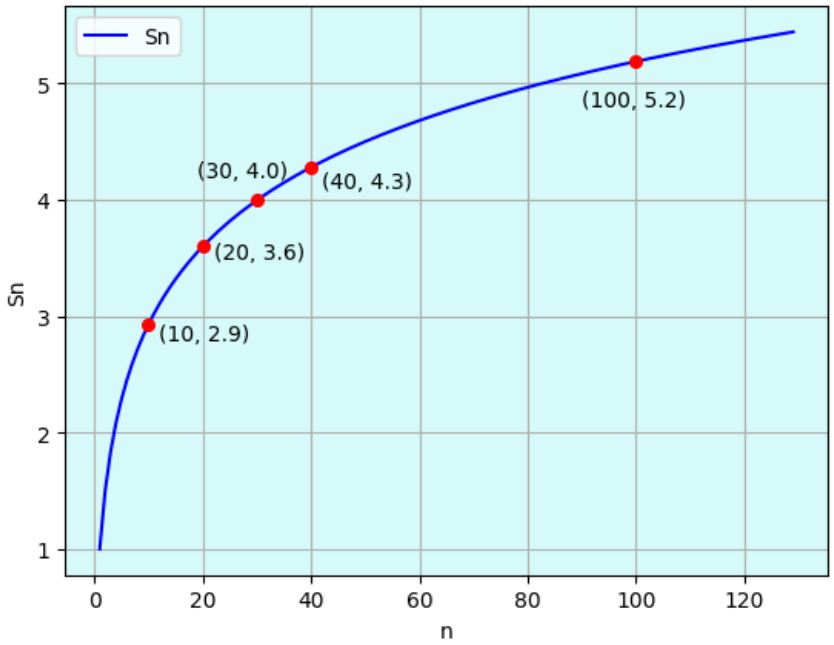


圖 21: S_n 的發散特性

2. γ_n 所代表的是圖 22 中橘色區塊的面積和，因次先運用積分計算每個區塊的面積後，再將這些面積做加總即可。證明過程如下：

$$
\begin{aligned}
\gamma_n &= \sum_{i=1}^{n} \int_{i}^{i+1} \int_{\frac{1}{x}}^{\frac{1}{i}} dydx \\
&= \sum_{i=1}^{n} \int_{i}^{i+1} \frac{1}{i} - \frac{1}{x} dydx \\
&= \sum_{i=1}^{n} \left(\frac{x}{i} - ln(x) \Big|_{i}^{i+1} \right) \\
&= \sum_{i=1}^{n} \left(\frac{i+1}{i} - ln(i+1) - \frac{i}{i} + ln(i) \right) \\
&= \sum_{i=1}^{n} \left(\frac{1}{i} - ln\left(\frac{i+1}{i} \right) \right) \\
&= \sum_{i=1}^{n} \left(\frac{1}{i} \right) - \sum_{i=1}^{n} ln\left(\frac{i+1}{i} \right) \\
&= S_n - \left[ln\left(\frac{2}{1} \right) + ln\left(\frac{3}{2} \right) + ln\left(\frac{4}{3} \right) + \cdots + ln\left(\frac{n+1}{n} \right) \right] \\
&= S_n - ln\left(\frac{2}{1} \cdot \frac{3}{2} \cdot \frac{4}{3} \cdot \cdots \cdot \frac{n+1}{n} \right) \\
&= S_n - ln\left(n+1 \right)
\end{aligned}
$$

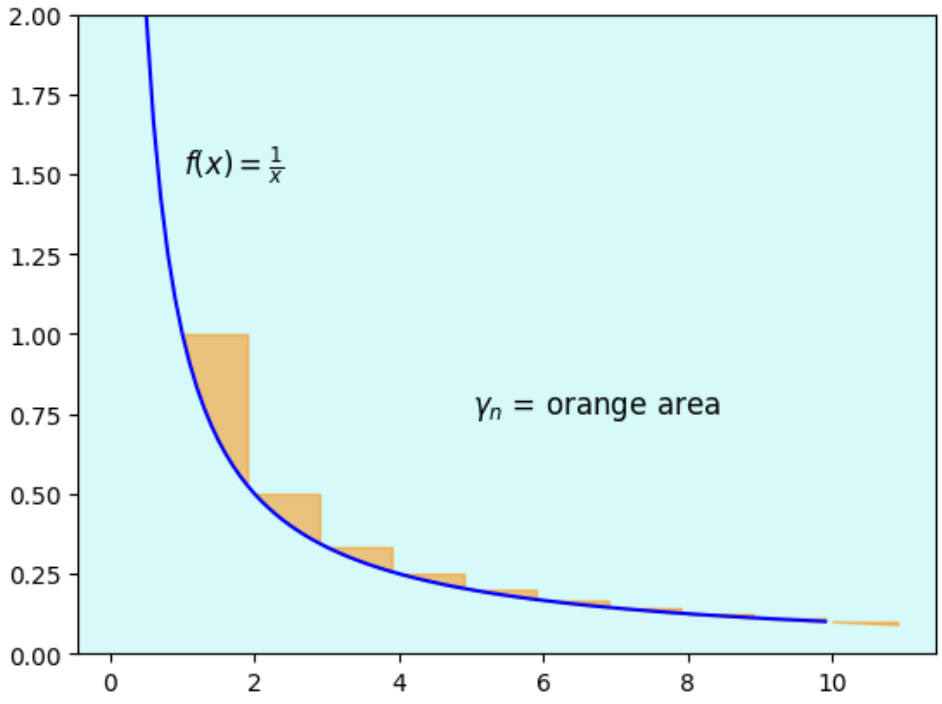
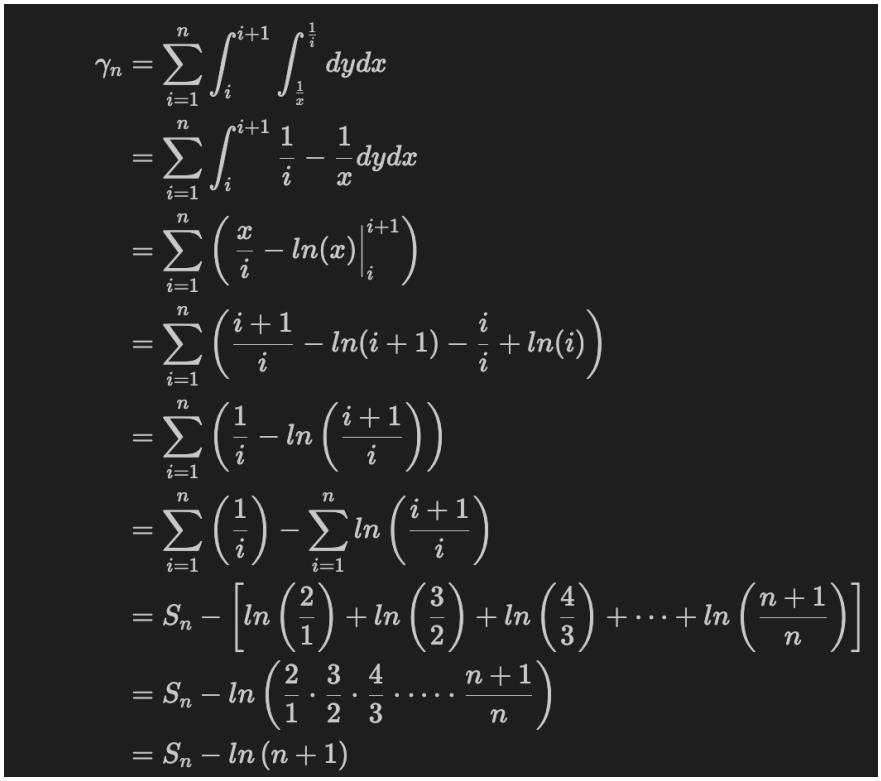


圖 22: γ_n = sum of the orange area



$$
\begin{aligned}
\gamma_n &= \sum_{i=1}^{n} \int_{i}^{i+1} \int_{\frac{1}{x}}^{\frac{1}{i}} dydx \\
&= \sum_{i=1}^{n} \int_{i}^{i+1} \frac{1}{i} - \frac{1}{x} dydx \\
&= \sum_{i=1}^{n} \left(\frac{x}{i} - ln(x) \Big|_{i}^{i+1} \right) \\
&= \sum_{i=1}^{n} \left(\frac{i+1}{i} - ln(i+1) - \frac{i}{i} + ln(i) \right) \\
&= \sum_{i=1}^{n} \left(\frac{1}{i} - ln\left(\frac{i+1}{i} \right) \right) \\
&= \sum_{i=1}^{n} \left(\frac{1}{i} \right) - \sum_{i=1}^{n} ln\left(\frac{i+1}{i} \right) \\
&= S_n - \left[ln\left(\frac{2}{1} \right) + ln\left(\frac{3}{2} \right) + ln\left(\frac{4}{3} \right) + \cdots + ln\left(\frac{n+1}{n} \right) \right] \\
&= S_n - ln\left(\frac{2}{1} \cdot \frac{3}{2} \cdot \frac{4}{3} \cdot \cdots \cdot \frac{n+1}{n} \right) \\
&= S_n - ln\left(n+1 \right)
\end{aligned}
$$

圖 23: 於 markdown 紀錄運算過程

此小題較特殊的程式碼：

```
#  開始一個循環，從0到9，這將用於繪製填充區域。
for i in range(0, 10):
#  創建一個NumPy數組x，它包含一系列連續數，從”1+i”到
    ”2+i”，間隔為0.1。
     x = np.arange(1 + i, 2 + i, 0.1)
#  創建一個NumPy數組y，其中每個元素是對應x數組中元素
    的倒數。
     y = 1/x
#  創建一個長度為10的NumPy數組g，其中的每個元素都是
    (1/i+1))的值
     g = [1/(i + 1)] * 10
#  繪製一個填充區域，這個區域的x值由x數組確定，y值由
    y數組確定，填充的底部由g數組確定。填充區域的顏色設
    置為（橙色）並設置透明度為0.5。
     ax.fill_between(x, y, g, color='#FF8C00', alpha
         =0.5)
#  設置標註於圖中想要的位置。
ax.text(1.0,1.5,'$f(x)=\\frac{1}{x}$',fontsize='12')
ax.text(5,0.75,'$\gamma_n$ = orange area',fontsize='
    12')
```

3. 圖 24 分別畫出了三個函數，紅色線代表的是 $y = 1$，藍色線代表的是 $y = \gamma_n$，綠色線代表的則是 $y = \frac{1}{2}(1 - \frac{1}{n+1})$。由圖中可清楚看到，不管 n 是多少，y 值的顏色順序皆為紅色、藍色、綠色，因此可驗證 $\frac{1}{2}(1 - \frac{1}{n+1}) < \gamma_n < 1$ 的結果。

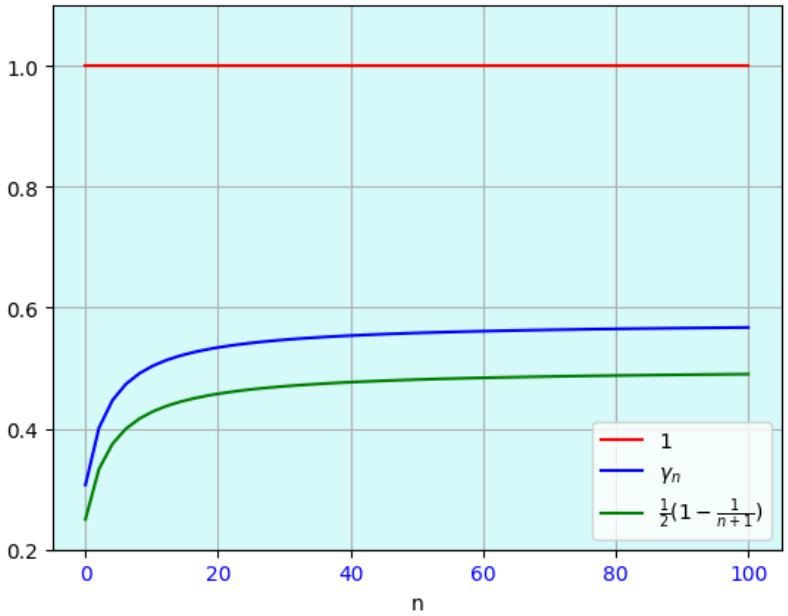


圖 24: $\frac{1}{2}(1 - \frac{1}{n+1}) < \gamma_n < 1$

圖 25 表示了為何此題會需要證明 γ_n 會介在兩個數中間。

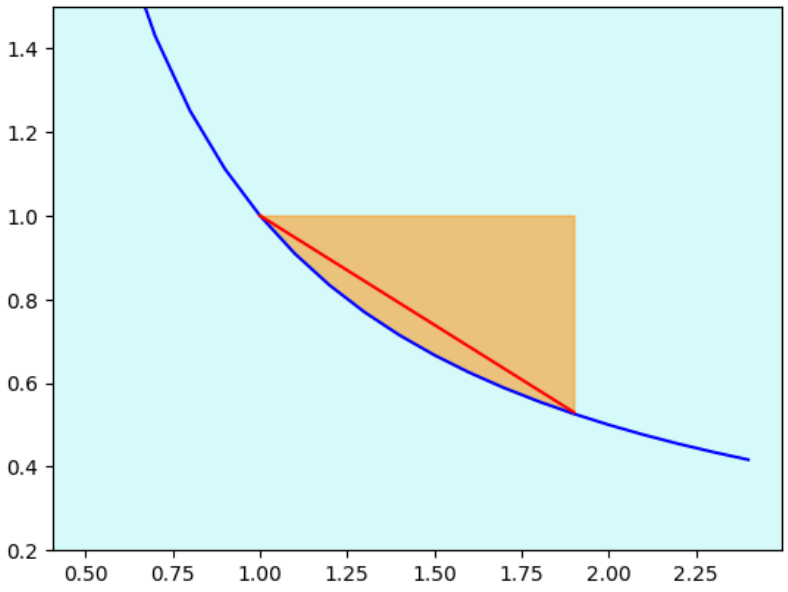


圖 25: 驗證 $\frac{1}{2}(1 - \frac{1}{n+1}) < \gamma_n < 1$

3 Ferris Wheel

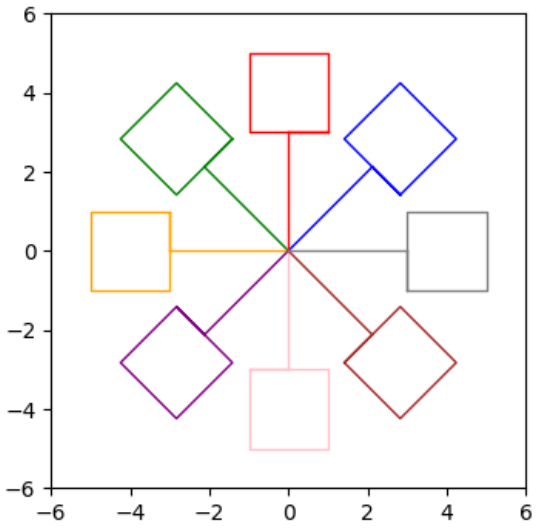


圖 26: 簡易摩天輪

圖 26 的程式碼：

```
#  設定多邊形的邊數，這裡是一個八邊形。
n=8
#  設定多邊形的邊長為2單位。
side_length=2
#  設定多邊形的中心座標，這裡是(4,0)。
center=(4,0)
#  計算每個多邊形的旋轉角度，將完整的圓周角度（360度）分
    成多邊形的邊數。np.deg2rad 函數用於將度數轉換為弧度。
rotation_angle = np.deg2rad(360/n)
#  定義一個正方形的x和y座標，它是多邊形的基本形狀。
square_x=[center[0]-side_length/2,center[0]+side_length
    /2,center[0]+side_length/2,center[0]-side_length/2,
    center[0]-side_length/2,center[0]-side_length/2]
square_y=[center[1]-side_length/2,center[1]-side_length
    /2,center[1]+side_length/2,center[1]+side_length/2,
    center[1]-side_length/2,center[1]]
#  將x和y座標組合成一個NumPy數組，代表多邊形的基本形狀。
square=np.array([square_x,square_y])
#  定義了一系列不同顏色的名稱，用於繪製多邊形的不同部
    分。
colors=['blue','red','green','orange','purple','pink','
    brown','gray']
#  創建一個 Matplotlib 圖形和軸的物件，指定圖形的大小為
    4x4 單位。
fig,ax=plt.subplots(figsize=(4,4))
#  設定 x 和 y 軸的等比例，確保多邊形不會變形。
ax.set_aspect(1)
```

```
#  開始一個迴圈，迴圈的次數等於多邊形的邊數。
for_in range(n):
#  計算旋轉矩陣，用於將多邊形旋轉到正確的位置。
    rotation_matrix=np.array([[np.cos(rotation_angle),-
       np.sin(rotation_angle)],[np.sin(rotation_angle),
       np.cos(rotation_angle)]])
#  應用旋轉矩陣來旋轉多邊形。
    square=np.dot(rotation_matrix ,square)
#  選擇多邊形的顏色，這裡使用了一系列預定義的顏色。
    colort=colors[_%len(colors)]
#  繪製旋轉後的多邊形。
    ax.plot(square[0],square[1],lw=1,color=colort)
#  繪製多邊形的一條邊，以模擬摩天輪的結構。
    ax.plot([square[0][5],0],[square[1][5],0],lw=1,color
       =colort)
#  保存圖形為一個圖像文件，這裡保存為 'ferris'，並指定
   DPI（每英寸像素數）和 bbox（用於確保圖形不被裁切）。
plt.savefig(fname='ferris',dpi=300,bbox_inches='tight')
```

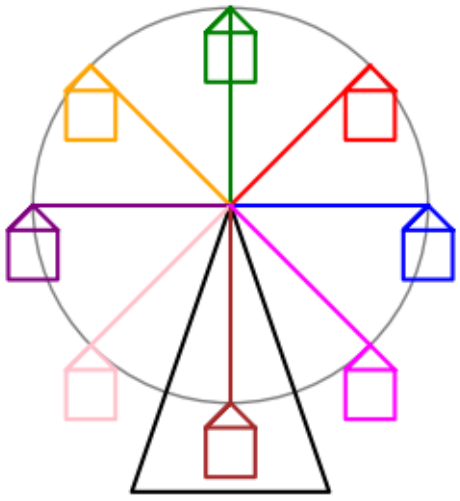


圖 27: 真實摩天輪

圖 27 的程式碼：

```
#  用於隱藏X軸與y軸的標尺
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
#  創建一個圓，位於坐標(0, 0)，半徑由radius變數指定，顏
   色為灰色，但不填充。
circle1=plt.Circle((0,0),radius ,color='gray',fill=False)
#  將上述創建的圓添加到圖的坐標軸ax中。
ax.add_patch(circle1)
```

```
#   這行程式碼創建一個名為tri的Numpy陣列，表示兩個三角形
    的坐標，用於繪製圖中的三角形。
tri=np.array([[0,-2,2,0],[0,-5.8,-5.8,0]])
#   這行程式碼根據square_x和square_y中的坐標繪製一個多邊
    形，顏色由colort指定。
      ax.plot(square_x,square_y,color=colort)
#   這行程式碼繪製一條線段，從center點到坐標(0, 0)，顏色
    由colort指定。
      ax.plot([center[0],0],[center[1],0],color=colort)
#   創建一個名為rotation_matrix的2x2矩陣，表示一個旋轉變
    換矩陣，用於旋轉坐標點。
      rotation_matrix=np.array([[np.cos(rotation_angle),-
         np.sin(rotation_angle)],[np.sin(rotation_angle),
         np.cos(rotation_angle)]])
#   這行程式碼使用矩陣乘法將旋轉矩陣rotation_matrix應用於
    center點，以實現坐標點的旋轉。center點現在將成為旋轉後
    的新坐標。
      center=np.dot(rotation_matrix,center)
```

4 結語

這篇文章介紹如何使用 Python 的數學計算和視覺化庫，例如 NumPy 和 Matplotlib，來實現繪製數學式的圖形。文章從簡單的方程式到更複雜的數學模型，深入探討了如何將數學式轉化為程式碼，並生成漂亮的圖形。這篇文章不僅適用於數學愛好者和教育工作者，還可應用於科學研究、數據分析和工程領域。我們學習了如何使用 Matplotlib 庫來繪製各種類型的函數圖形，例如線性函數、二次函數、三角函數、指數函數和對數函數等。

這篇文章還介紹了比較定理，這是一種用於研究數列、函數和積分性質的定理。比較定理有多個不同的變種，包括比較收斂定理和比較發散定理。比較收斂定理可以用於確定一個級數是否收斂，即總和有一個有限極限。如果我們可以找到一個已知的收斂級數，並證明要評估的級數的每一項都小於對應的已知收斂級數的每一項，那麼我們可以得出要評估的級數也是收斂的。

最後，這篇文章強調了數學和程式設計的完美結合。無論您是數學愛好者、教育工作者、科學研究者、數據分析師還是工程師，Python 的數學計算和視覺化庫都可以幫助您更好地理解和展示數學概念。