

舞動數學 - Manim 動畫入門

Ching-Yu Yang

Department of Mathematics
National Taiwan Normal University

January 19, 2022

- 1 前言
- 2 安裝嘗鮮篇
- 3 2D 幾何物件篇
- 4 文字與數學式篇
- 5 方程式圖形篇
- 6 動畫類別篇
- 7 附錄

舞動數學 Animated Math

(網站廣告)

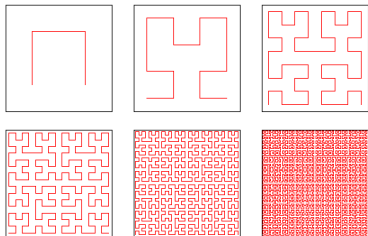
贊助單位：教育部數學領域教學研究中心、臺灣師範大學數學系

為何需要動畫

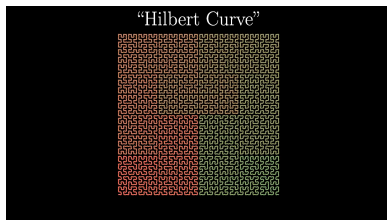
俗語說：

文不如表，表不如圖，圖不如動畫。

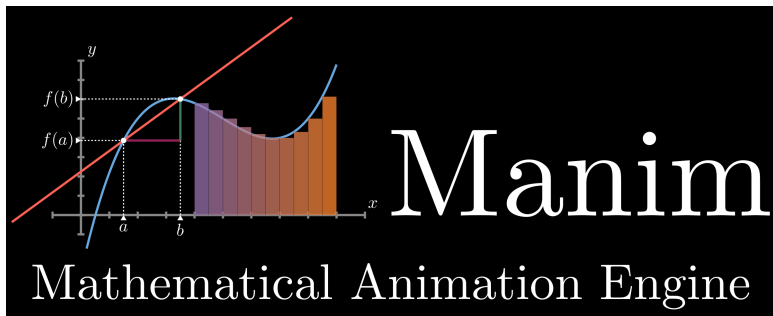
舉例：希爾伯特曲線 (Hilbert's Curve)



靜態圖表 (取自 [wiki](#))



動畫展示 (取自 [3b1b](#) 影片)



Manim 是由史丹福大學的數學研究生 Grant Sanderson 根據 Python 語言發展出來的電腦圖形套件。

先來看一段 youtube 上介紹 Manim 的影片：

<https://www.youtube.com/watch?v=ENMyFGmq5OA>

優點：

- 本身有很好的二維及三維物件定義。
- 對數學或物理方法實現過程的視覺化效果相當優異。
- 可以利用 Python 眾多程式庫及特色來做數據處理。
- 產生的視覺化動畫質量很高，適合展示或教學。

缺點：

- 依賴程式設計的能力較高。
- 無法通過已有數據，快速產生曲面或數據曲線。
- 無法做即時互動效果。



安裝嘗鮮篇

Manim 環境設置

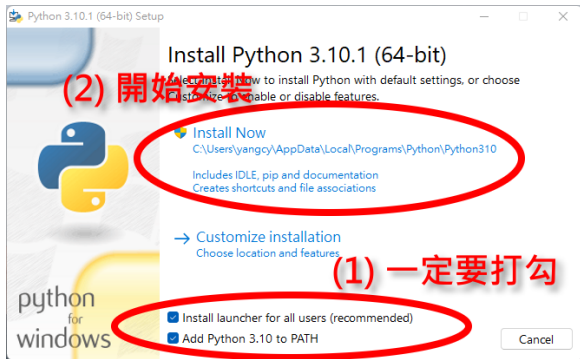
- Python 程式語言 (3.6 以上版本)
- \LaTeX 數學式編譯程式 (Win: MikTeX / Mac: MacTeX)
- ffmpeg 動畫檔案轉換程式
- Manim 程式庫套件 (manimgl)
- SoX SoundExchange 聲音檔案轉換程式

Python 程式語言安裝

步驟 (1)：下載安裝程式 (64 位元)

- Windows 7 - [官方網頁](#) [分流一](#) [分流二](#)
- Windows 10/11 - [官方網頁](#) [分流一](#) [分流二](#)

步驟 (2)：進行安裝



MikTeX + ffmpeg 安裝

MikTeX 安裝較為複雜，為節省時間，使用自製免安裝包。

步驟 (1)：下載 **懶人包** 壓縮檔 (Latex-x64.zip)。

步驟 (2)：解壓縮壓縮檔至任何可寫入目錄 (例如：桌面)。

步驟 (3)：以「**系統管理員身份**」執行 Latex-x64 子目錄內的 `install.cmd` 批次檔。

Q：如何以系統管理員身份執行 `install.cmd`？

A：滑鼠游標移到 `install.cmd` 檔案上按滑鼠右鍵，選擇「**以系統管理員身份執行**」。

結論：批次檔執行完，會把 MikTeX 下載解壓到 C 目錄下，並把執行檔目錄加入系統 Path 變數定義中。同時會將 `ffmpeg.exe` 放置在 Windows 目錄內。

Manim 的三種版本

Manim 至今有以下三個版本：

(1) **ManimGL**：由原作者及 3Blue1Brown 開發的版本。

官網 / [GitHub](#) / [YouTube 3Blue1Brown 頻道](#)

引入套件：`from manimlib import *`

(本研習使用此版本)

(2) **ManimCE**：由社群主動維護的版本，使用者及社群討論較多。

[社群網站](#)

引入套件：`from manim import *`

(3) **ManimCairo**：原作者開發的原始舊版本，已從 pip 移除。

步驟 (1)：開啟「命令提示字元」視窗

方法 1：在搜尋列裡查尋 cmd 指令，或直接「執行」cmd 指令。

方法 2：在「所有程式」中尋找「命令提示字元」程式。

步驟 (2)：在「命令提示字元」視窗裡輸入套件安裝指令

```
pip3 install manimgl
```

Q：如何知道是否安裝完成 Manim？

A：在「命令提示字元」視窗裡輸入以下指令。

```
manimgl
```

如果螢幕右上方出現**黑色視窗**，命令列變彩色即是完成。

manimgl 是 Manim 的編譯程式，可以當成直譯式環境使用。

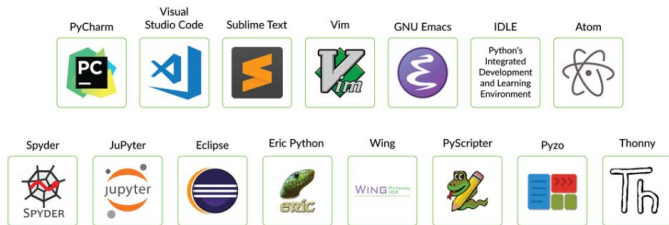
試著輸入以下指令，感受動態呈現。

```
self.add(Circle())
self.wait()
S = Square()
self.play>ShowCreation(S))
self.play(S.move_to,(2,2,0))
T = Triangle().move_to((-2,-2,0))
self.play(Transform(S,T))
M = Tex("f(x)=x^2+\\sin x").move_to((-2,2,0))
self.play(Write(M))
self.play(Rotate(M,2*PI))
self.play(Write(Text("Hello World!"))))
```

(練習時間)

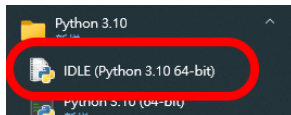
Manim 的編譯式模式

你可以使用個人喜好的編輯器 (IDE) 來編寫 Manim。



這裡我們使用 Python 自帶的 Python IDLE 編輯器。

程式集 \Rightarrow Python 3.10 \Rightarrow IDLE (Python 3.10 64-bit)



編譯第一個 Manim 動畫

IDLE \Rightarrow File \Rightarrow Open (選擇 `example1.py` 檔案) \Rightarrow F5 編譯程式

```
from manimlib import *

class Example1(Scene):
    def construct(self):
        self.add(Circle())
        self.wait()
        S = Square()
        self.play>ShowCreation(S))
        self.play(S.move_to,(2,2,0))
        T = Triangle().move_to((-2,-2,0))
        self.play(Transform(S,T))
        M = Tex("f(x)=x^2+\\sin x").move_to((-2,2,0))
        self.play(Write(M))
        self.play(Rotate(M,2*PI))
        self.play(Write(Text("Hello World!")))

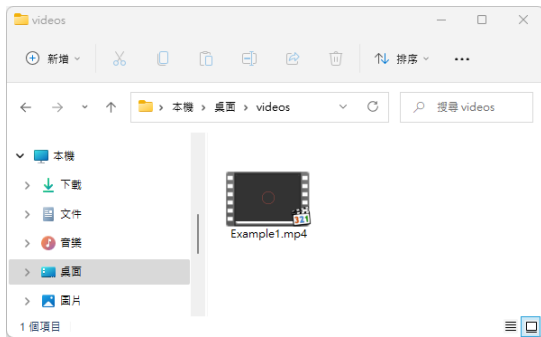
if __name__ == "__main__":
    myfile = os.path.basename(__file__)
    os.system("manimgl -om "+myfile)
```

編譯完成的動畫檔案在那裡？

正確編譯完產生的動畫檔，會放在程式檔所在目錄下的 **videos** 子目錄下，主檔名與 **class 類別宣告名稱** 相同，副檔名為 **mp4**。

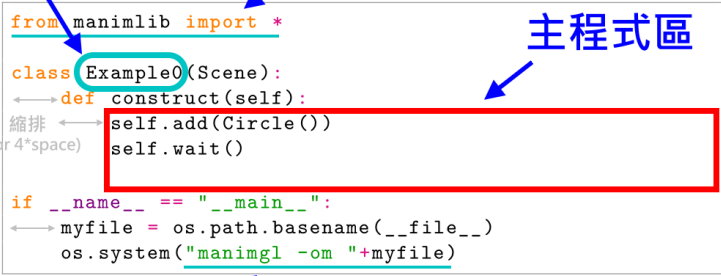
例如：如果 `example1.py` 放在桌面 (Desktop)，那麼編譯完的 `Example1.mp4` 則放在 `Desktop\videos` 目錄下。

Desktop/
example1.py
videos/
Example1.mp4



標準程式碼解說

以 `example0.py` 標準程式碼來解說。



The diagram shows a code editor window with the following Python code:

```
from manimlib import *  
  
class Example0(Scene):  
    def construct(self):  
        self.add(Circle())  
        self.wait()  
  
if __name__ == "__main__":  
    myfile = os.path.basename(__file__)  
    os.system("manimgl -om "+myfile)
```

Annotations with arrows pointing to specific parts of the code:

- 與輸出影片主檔名相同** (Same as output video filename): Points to the `Example0` class name.
- 引入 manim 套件庫** (Import manim library): Points to the `from manimlib import *` line.
- 主程式區** (Main program area): Points to the `if __name__ == "__main__":` block.
- 縮排 (1*tab or 4*space)** (Indentation): Points to the `def construct(self):` line.
- 編譯指令 (720p輸出影片,編譯完直接開啟影片)** (Compilation command): Points to the `os.system("manimgl -om "+myfile)` line.

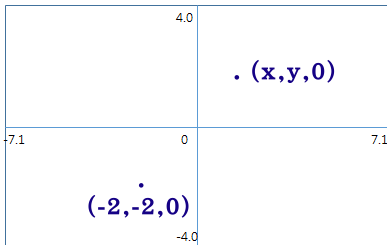
備註：欲修改編譯指令，可以參考 [官方網頁](#) 說明。



2D 幾何物件篇

Manim 的內建座標系統

本篇使用的是二維座標系， x 軸 -7.1 至 7.1 ， y 軸 -4.0 至 4.0 。



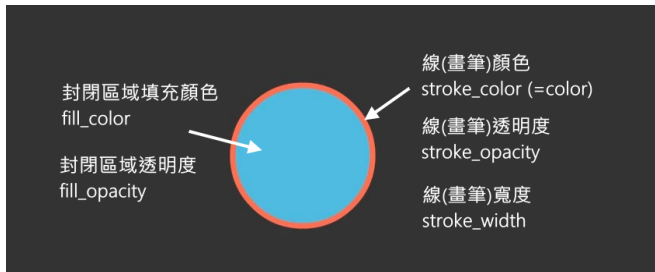
以 `np.array((x,y,z))` 陣列型態表現座標，2D 場景裡 z 座標設為 0 。

幾個可以用在方向性的特別座標替代常數：(所有常數請參考[官方網頁說明](#))

```
ORIGIN=np.array((0,0,0))
UP=np.array((0,1,0))
DOWN=np.array((0,-1,0))
RIGHT=np.array((1,0,0))
LEFT=np.array((-1,0,0))
IN=np.array((0,0,-1))
OUT=np.array((0,0,1))
UL=UP+LEFT
```

```
UR=UP+RIGHT
DL=DOWN+LEFT
DR=DOWN+RIGHT
TOP=FRAME_Y_RADIUS*UP
BOTTOM=FRAME_Y_RADIUS*DOWN
LEFT_SIDE=FRAME_X_RADIUS*LEFT
RIGHT_SIDE=FRAME_X_RADIUS*RIGHT
```


Manim 的 2D 幾何物件 - 以圓為例 (2)



可封閉物件 (曲線) 額外參數：

- `fill_color = stroke_color` (封閉區域填充顏色)
- `fill_opacity = 0` (封閉區域透明度 / 1 不透明 / 0 全透明)

上面圓的程式碼：

```
self.add(Circle(color=RED,stroke_width=8,fill_color=
              BLUE,fill_opacity=1))
self.wait()
```

Manim 場景 (動畫) 添加物件的兩種方式

`self.add(mobject|Method, ...)`

靜態 添加物件或方法 (動作) 在場景上，需要有時間差才能看到物件，所以通常會搭配 `self.wait()` 等待一秒，讓畫面更新。

`self.play(ShowCreation|Write|Method..., ...)`

動態 應用方法 (動作) 在場景上，通常過場 (播放) 時間為 1 秒。

可使用的額外參數：

- `run_time = 1` (播放時間長度, 單位秒)
- `rate_func = smooth` (播放速率參數應用函數，詳見[官網原始碼](#))

`ShowCreation(mobject)` 動態生成 2D/3D 物件的方法。

例如：在 3 秒內動態生成一個半徑 2，厚度 20pts 的綠色圓。

```
C = Circle(radius=2,stroke_width=20,color=GREEN)
self.play(ShowCreation(C),run_time=3)
```

Manim 移動物件的一種方法 `move_to`

`Mobject.move_to(point_or_mobject)`

將物件 `Mobject` 移動到指定點座標或物件處。

例如：添加一個圓心在 $(2,2,0)$ 的紅圓。

```
self.add(Circle().move_to((2,2,0)))  
self.wait()
```

想動態呈現移動過程，需用 `self.play` 搭配 `ApplyMethod` 方法。

`ApplyMethod(mobject.method, ...)` (應用方法)




例如：紅圓在原點處等待 2 秒後，動態移動到 $(2,2,0)$ 處



```
C = Circle()  
self.add(C)  
self.wait(2)  
self.play(ApplyMethod(C.move_to, (2,2,0)))
```

`self.play` 搭配 `ApplyMethod()` 方法，可省略 `ApplyMethod()` 簡略式寫法

```
self.play(C.move_to, (2,2,0))
```

Manim 2D 幾何物件 (1)

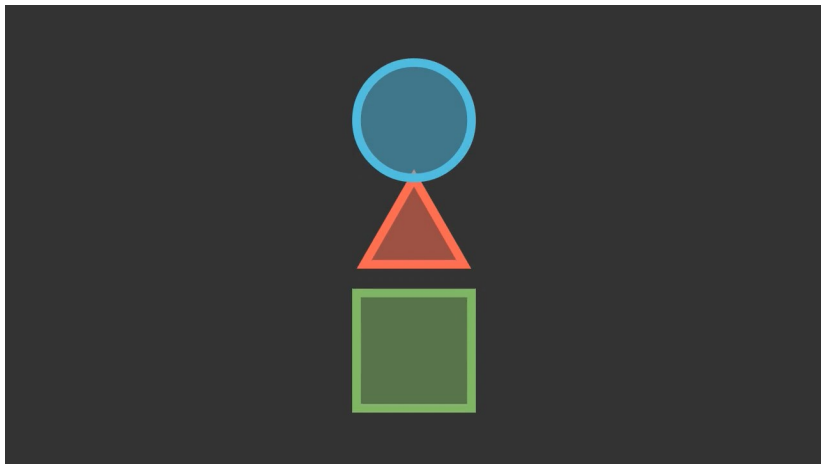
物件形狀			
物件名稱	Circle	Square	Triangle
個別參數	<code>radius=1.0</code> <code>arc_center=ORIGIN</code>	<code>side_length=2.0</code>	<code>start_angle=90*DEGREES</code>

物件形狀		
物件名稱	Rectangle	RoundedRectangle
個別參數	<code>width=4.0</code> <code>height=2.0</code>	<code>width=4.0</code> <code>height=2.0</code> <code>corner_radius=0.5</code>


練習時間 ~ (ex1)

你可以使用目前教到的物件與移動方法來畫出下圖中的形狀嗎？

動畫請參考 [Ex1.mp4](#)



Manim 2D 幾何物件 (2)

物件形狀		
物件名稱	Polygon	RegularPolygon
個別參數	*vertices	$\text{start_angle} = \begin{cases} n=6 \\ 0 \text{ (even)} \\ 90 * \text{DEGREES (odd)} \end{cases}$

`Polygon()` 的參數 `*vertices` 是不定個數的頂點座標，上面的四邊形定義如下。

```
Polygon((-0.5, -0.5, 0), (1.5, 0.5, 0), (0, 2, 0), (-1, 0.5, 0))
```

也可以把頂點寫成清單 (list)，再把清單內容引入參數。

```
v = [(-0.5, -0.5, 0), (1.5, 0.5, 0), (0, 2, 0), (-1, 0.5, 0)]  
P = Polygon(*v)  
self.play(ShowCreation(P))
```




正三角形 `Triangle()` 就是正多邊形 `RegularPolygon(n=3)`。




練習時間 ~ (ex2)

你能夠使用 `Polygon()` 來畫出如圖邊長 3 與 2，夾角 60 度的平行四邊形嗎？ 動畫請參考 [Ex2.mp4](#)

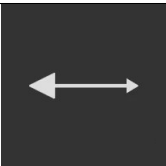
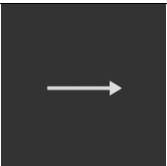





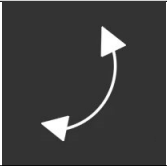
Manim 2D 幾何物件 (3)

物件形狀			
物件名稱	Dot	SmallDot	Line
個別參數	<code>radius=0.08</code> <code>point=ORIGIN</code>	<code>radius=0.04</code> <code>point=ORIGIN</code>	<code>start=LEFT</code> <code>end=RIGHT</code>

物件形狀			
物件名稱	DashedLine	Elbow	Arrow
個別參數	<code>start=LEFT</code> <code>end=RIGHT</code> <code>dash_length=0.05</code>	<code>width=0.2</code> <code>angle=0</code>	<code>start=LEFT</code> <code>end=RIGHT</code> <code>buff=0.25</code>

Manim 2D 幾何物件 (4)

物件形狀			
物件名稱	DoubleArrow	Vector	Arc
個別參數	<code>start=LEFT</code> <code>end=RIGHT</code> <code>buff=0.25</code>	<code>direction=RIGHT</code>	<code>radius=1.0</code> <code>start_angle=0</code> <code>angle=TAU/4</code> <code>arc_center=ORIGIN</code>

物件形狀			
物件名稱	ArcBetweenPoints	CurvedArrow	CurvedDoubleArrow
個別參數	<code>start</code> <code>end</code> <code>angle=TAU/4</code>	<code>start_point</code> <code>end_point</code> <code>angle=TAU/4</code>	<code>start_point</code> <code>end_point</code> <code>angle=TAU/4</code>

Vector 與物件位移方法 shift

Vector 其實是起點 (start) 是 ORIGIN，終點 (end) 是 **direction** 且 buff=0 的箭頭，就是數學上的向量定義。

物件子方法中有一個行為跟向量加法很像的 **shift()** 位移：

Mobject.shift(vector)

物件 Mobject 相對移動向量 vector。

例如：添加一個圓心在 (2,2,0) 的紅圓，等待 1 秒後動態往下位移 2 單位。

```
C = Circle().move_to((2,2,0))
self.add(C)
self.wait()
self.play(C.shift, DOWN*2)
```

上述圓心位置 shift 其實就是操作向量加法：

`np.array((2,2,0))+np.array((0,-2,0))=np.array((2,0,0))`

Vector 向量加法範例

給定向量 $\vec{A} = (3, 0)$, $\vec{B} = (1, 2)$ ，利用 `shift` 動態演示向量 $\vec{A} + \vec{B}$ 。

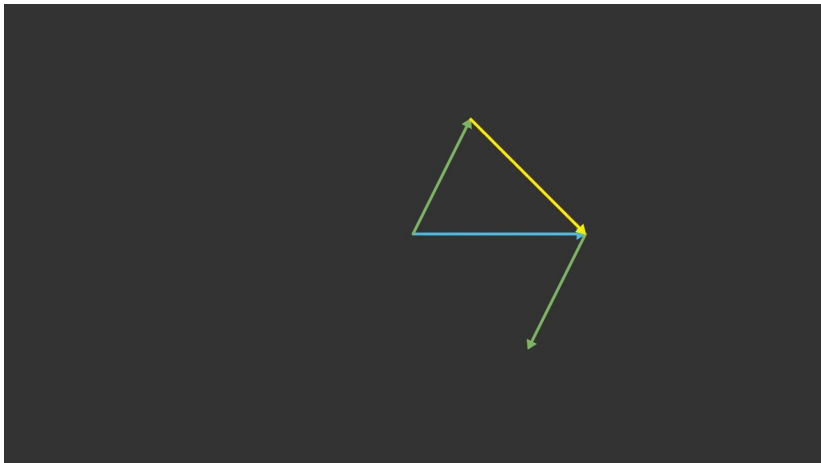
(example2)

```
a = np.array((3,0,0))
b = np.array((1,2,0))
A = Vector(a,stroke_color=BLUE)
B = Vector(b,stroke_color=GREEN)
self.play(ShowCreation(A))
self.play(ShowCreation(B))
self.wait()
self.play(B.shift,a) # B沿向量a移動
self.wait()
C = Vector(a+b,stroke_color=YELLOW) # 用向量a+b定義C
self.play(ShowCreation(C))
```




練習時間 ~ (ex3)

同上頁向量加法的範例，你能修改成動態表現向量減法 $\vec{A} - \vec{B}$ 嗎？



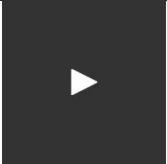
提示：你可以利用 $\mathbf{A} - \mathbf{B} = \mathbf{A} + (-\mathbf{B})$ ，動畫請參考 [Ex3.mp4](#)



Manim 2D 幾何物件 (5)

物件形狀			
物件名稱	AnnularSector	Sector	Annulus
個別參數	<code>inner_radius=1</code> <code>outer_radius=2</code> <code>angle=TAU/4</code> <code>start_angle=0</code> <code>arc_center=ORIGIN</code>	<code>inner_radius=0</code> <code>outer_radius=1</code> <code>angle=TAU/4</code> <code>start_angle=0</code> <code>arc_center=ORIGIN</code>	<code>inner_radius=1</code> <code>outer_radius=2</code> <code>arc_center=ORIGIN</code>

Manim 2D 幾何物件 (6)

物件形狀			
物件名稱	Ellipse	CubicBezier	ArrowTip
個別參數	<code>width=2</code> <code>height=1</code> <code>arc_center=ORIGIN</code>	<code>a0</code> <code>h0</code> <code>h1</code> <code>a1</code>	<code>width=0.35</code> <code>length=0.35</code> <code>angle=0</code> <code>tip_style=0</code>

`CubicBezier()` 是 3 維貝茲曲線，參數是 4 個點座標 `a0,h0,h1,a1`。(詳見 [wiki](#))

上面曲線的定義：

```
CubicBezier((-2,1,0),(1,-1,0),(3,0,0),(3,2,0))
```

`ArrowTip()` 的箭頭形狀參數 `tip_style = 0` 三角形 / 1 內部圓形 / 2 圓形

物件的外觀設定子方法

除了參數外，還可以利用物件的子方法來設定外觀。

`Object.set_color(color)` (影響整體顏色)

`Object.set_stroke(color=None, width=None, opacity=None)`

(影響描線顏色、寬度與透明度)

`Object.set_fill(color=None, opacity=None)` (影響填充區顏色與透明度)

`Object.set_opacity(opacity)` (影響整體透明度)

`Object.set_width(width)` (設定物件寬度)

`Object.set_height(height)` (設定物件高度)

`Object.scale(scale_factor, about_point=None, about_edge=ORIGIN)`

(延展 (即等比例放大縮小) · `about_point` 由此點延展 · `about_edge` 由物件邊緣延展 · 內定值為由物件中心延展)

`Object.stretch(factor, dim)`

(沿 `dim` 維度方向作延展 · `dim = 0` x軸 / `1` y軸 / `2` z軸)

物件的位置或座標有關的子方法

除了 `move_to` 與 `shift` 外，比較常用跟位置與座標有關的子方法。

`Mobject.set_x(x)` (將物件 x 座標設為 x)

`Mobject.set_y(y)` (將物件 y 座標設為 y)

`Mobject.set_z(z)` (將物件 z 座標設為 z)

`Mobject.next_to(mobject, direction=RIGHT, buff=0.25)`

(將物件放到 `mobject` 旁邊)

`Mobject.to_edge(edge=LEFT, buff=0.5)` (將物件放到畫面邊緣)

`Mobject.center()` (將物件放到畫面中央)

`Mobject.rotate(angle, axis=OUT)` (自身旋轉 angle 度， axis 是旋轉軸)

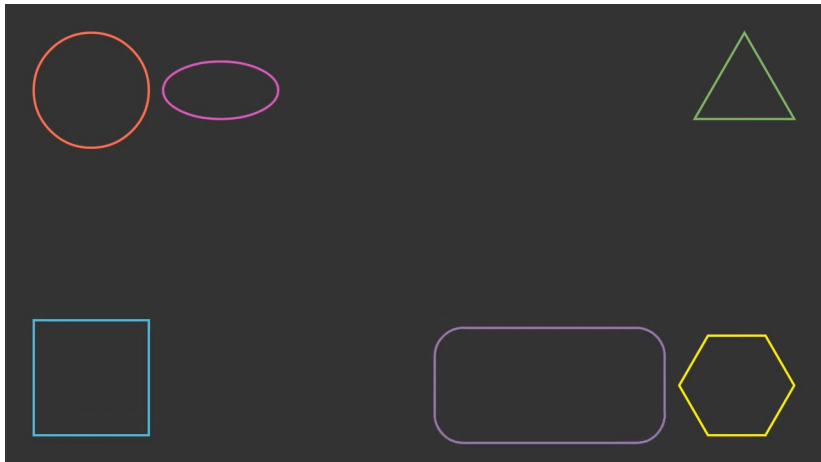
`Mobject.rotate_about_origin(angle, axis=OUT)` (旋轉點在 `ORIGIN`)

`Mobject.flip(axis=UP)` (旋轉軸 `UP`，旋轉 $\text{TAU}/4$)

練習時間 ~ (ex4)

使用 `to_edge` 與 `next_to` 由中央動態移動至角落，如下圖及動畫所示。

動畫請參考 [Ex4.mp4](#)





文字與數學式篇

Manim 數學式物件 Tex

Tex(tex_string, ...) (定義符合 Latex 語法字串 tex_string 的數學式物件)

內定參數：

`font_size = 48` (字體大小，單位 pts)

`fill_opacity = 1` (內部透明度 1)

`stroke_width = 0` (描線寬度 0)

可使用 `self.play(Write())` 動畫式秀出文字，或 `self.add()` 搭配 `self.wait()`。
數學式採用標準 LaTeX 語法，反斜線在 Python 字串中為跳脫符號，所以請用兩個反斜線表示一個反斜線。

數學符號或寫法請參考 [LaTeX 符號大全](#) 及 [線上 LaTeX 產生器](#)。

例如： $\sin^2 x + \cos^2 x = 1$ ，字體大小 96，顏色藍色。

```
self.play(Write(Tex("\\sin^2 x+\\cos^2 x=1",font_size=96,
                    ,color=BLUE)))
```

例如： $\sin^2 x + \cos^2 x = 1$ ，字體大小 96，描線顏色藍色，描線寬度 2，內部背景色。

```
self.play(Write(Tex("\\sin^2 x+\\cos^2 x=1",font_size=96,
                    ,stroke_color=BLUE,
                    stroke_width=2,fill_color="#
                    333333")))
```

多行數學式物件 Tex 的呈現方式 (一)

方式 (一)：使用 `next_to` 方法。

例如：定義兩個 Tex 物件 $A^2 + B^2 = C^2$ 與 $A^2 = C^2 - B^2$ ，然後用 `next_to` 定位。

```
T1 = Tex("A^2+B^2=C^2")
T2 = Tex("A^2=C^2-B^2")
self.play(Write(T1))
T2.next_to(T1,DOWN)
self.play(Write(T2))
```

動態效果。

```
T1 = Tex("A^2+B^2=C^2")
T2 = Tex("A^2=C^2-B^2")
self.play(Write(T1))
self.play(T2.next_to,T1,DOWN)
```

備註：此方式適用於自行個別調整每行位置使用。

多行數學式物件 Tex 的呈現方式 (二)

方式 (二)：使用 `VGroup` 類別，並搭配 `arrange` 排列子方法。

`VGroup(*vmobject)` (將向量化的物件群組化使用)

`Mobject.arrange(direction=RIGHT, center=True, buff=0.25)`

(將群組內子物件排列，內定往右排列，群組移至畫面中央)

例如：用 `VGroup` 群組化兩個 Tex 物件 $A^2 + B^2 = C^2$ 與 $A^2 = C^2 - B^2$ 。

```
T = VGroup(Tex("A^2+B^2=C^2"), Tex("A^2=C^2-B^2"))
T.arrange(DOWN) # 往下排列
self.play(Write(T))
```

可以使用索引方式存取 `VGroup` 中個別的物件，如設定兩行數學式的顏色。

```
T[0].set_color(BLUE)
T[1].set_color(GREEN)
```

備註：此方式適用於群組化所有數學式，並可用索引個別存取設定。

練習時間 ~ (ex5)

完成下面數學式的解說，並加上顏色與移動到適當位置。

(開根號的 Latex 語法為 `\sqrt{\quad}`) 動畫請參考 [Ex5.mp4](#)

$$\begin{aligned}A^2 + B^2 &= C^2 \\A^2 &= C^2 - B^2 \\A^2 &= (C - B)(C + B) \\A &= \sqrt{(C - B)(C + B)}\end{aligned}$$

Manim 的文字物件 Text

Text(text, ...) (定義字串 text 非數學式的文字物件)

內定參數：

font = "" (字體設定，可為系統字型的英文名稱)

font_size = 48 (字體大小，單位 pts)

fill_opacity = 1 (內部透明度 1)

stroke_width = 0 (描線寬度 0)

```
T = Text("向量加法", font="Microsoft JhengHei")
self.play(Write(T))
```

「Microsoft JhengHei」為 Win* 系統的「微軟正黑體」字型英文名稱。

Q: 如何得知系統中文字型的英文名稱呢？

A: LaTeX 系統有提供一個檢視系統字型的指令 `fc-list`：

先啟動 cmd.exe 或 mac 下的終端機，並切換到可寫入的目錄下，執行下列指令：

```
fc-list -f "%{family}\n" :lang=zh > zhfont.txt
```

zhfont.txt 中會列出系統中各字型對應的英文名稱。

測試字型：[演示斜黑體](#) [王漢宗中隸書繁](#)¹ ([example3](#))

¹[演示斜黑體](#)為更改自思源黑體之免費字型，王漢宗中隸書繁為[王漢宗自由字型](#)裡的繁體隸書字型。

練習時間 ~ (ex6)

安裝字型 演示斜黑體 與 王漢宗中隸書繁 並修改 (example3) 使得字型與顏色如同下圖所示。

提示：第一行是演示斜黑體，第二行王漢宗中隸書繁體，其餘微軟正黑體，顏色 BLUE_A 到 BLUE_E。動畫請參考 Ex6.mp4

Wang Wei - Lu Chai

空山不見人

但聞人語响

深い森に戻る

이끼에 다시 사진

結合 Text 與 Tex 物件

利用 `VGroup` 類別與 `arrange` 方法來結合文字與數學式。

(example4)

```
T1 = VGroup(  
    Text("多項式函數",font="Microsoft JhengHei"),  
    Tex("f(x)=ax^3+bx^2+cx+c")  
)  
T1.arrange() # 同一行內定往右排列  
T2 = VGroup(  
    Text("其中",font="Microsoft JhengHei"),  
    Tex("a,b,c"),  
    Text("均為有理數。",font="Microsoft JhengHei")  
)  
T2.arrange() # 同一行內定往右排列  
T = VGroup(T1,T2) # 結合T1,T2兩行  
T.arrange(DOWN) # 兩行間是往下排列  
self.play(Write(T),run_time=5)
```

練習時間 ~ (ex7)

使用 `VGroup` 與 `arrange` 完成下列的數學命題。

(開根號的 Latex 語法為 `\sqrt{\}`) 動畫請參考 [Ex7.mp4](#)

設 $(1 + \sqrt{2})^6 = a + b\sqrt{2}$ ，其中 a, b 為整數。
請問 b 等於下列哪一個選項？

VGroup 用在結合文字與物件

例如：向量加法的例子，可以在向量旁邊加上數學符號。(example5)

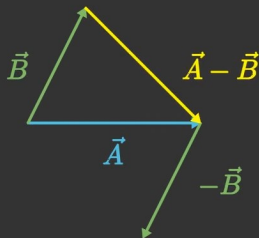
```
a = np.array((3,0,0))
b = np.array((1,2,0))
A = Vector(a,stroke_color=BLUE)
B = Vector(b,stroke_color=GREEN)
self.play>ShowCreation(A))
tex_A = Tex("\\vec{A}",color=BLUE).next_to(A,DOWN)
self.play(Write(tex_A))
self.play>ShowCreation(B))
tex_B = Tex("\\vec{B}",color=GREEN).next_to(B,RIGHT,
                                         buff=0)

self.play(Write(tex_B))
BGroup = VGroup(B,tex_B) # 將向量B跟符號tex_B群組化
self.play(BGroup.shift,a) # B沿向量a移動
C = Vector(a+b,stroke_color=YELLOW)
self.play>ShowCreation(C))
tex_C = Tex("\\vec{A}+\\vec{B}",color=YELLOW).next_to(C,
                                                    ,UP,buff=0)
tex_C.shift(DOWN*0.5) # 不夠接近,微調一下
self.play(Write(tex_C))
```

練習時間 ~ (ex8)

完成向量減法的練習，並為你的向量加上對應的數學符號。

動畫請參考 [Ex8.mp4](#)





方程式圖形篇

FunctionGraph(function, ...) (函數 function 的圖形物件)

必要參數：

- `function` (函數的名稱，可用 `def` 或 `lambda` 定義)

內定參數：

- `color = YELLOW` (顏色內定是黃色)
- `x_range = [-8, 8, 0.25]` (畫圖的 x 範圍與間隔：跟平滑度有關)

例如：函數 $\sin x$ 與 $x^3 - 3x$ 繪圖 (example6)

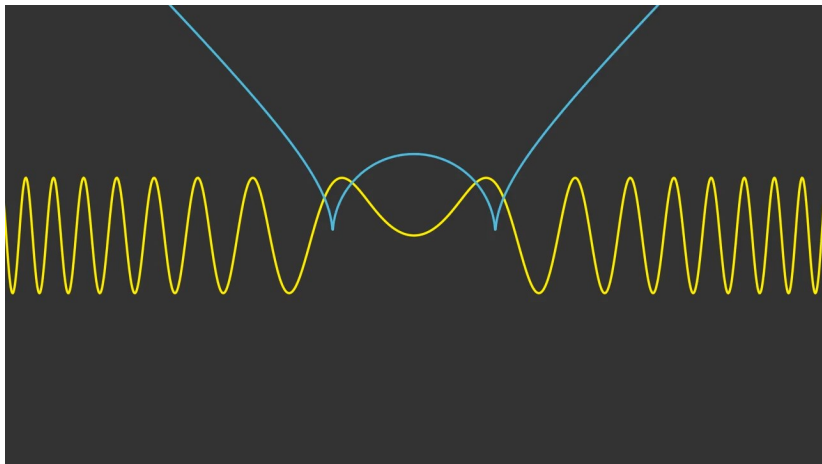
```
def f1(x):  
    return np.sin(x)  
  
F1 = FunctionGraph(f1)  
F2 = FunctionGraph(lambda x:x**3-3*x, color=BLUE)  
  
self.play(ShowCreation(F1))  
self.play(ShowCreation(F2))
```

練習時間 ~ (ex9)

使用 `FunctionGraph` 畫函數 f_1 與 f_2 圖形。

$$f_1(x) = \sin(x^2), \quad f_2(x) = \sqrt{|x^2 - 2|}$$

(絕對值可使用 `np.abs()`，開根號為 `np.sqrt()` 或 `**0.5`) 動畫請參考 [Ex9.mp4](#)



ParametricCurve(*t_func*, ...) (函數 *function* 的圖形物件)

必要參數：

- *t_func* (參數 *t* 的方程式函數名，以 `np.array((x(t),y(t),z(t)))` 為返回值)

內定參數：

- `color = WHITE` (顏色內定是白色)
- `t_range = [0, 1, 0.1]` (參數 *t* 範圍與間隔: 跟平滑度有關)

例如：參數方程式 $x = \sin(2t)$, $y = \sin(3t)$, $0 \leq t \leq 2\pi$ 繪圖 ([example7](#))

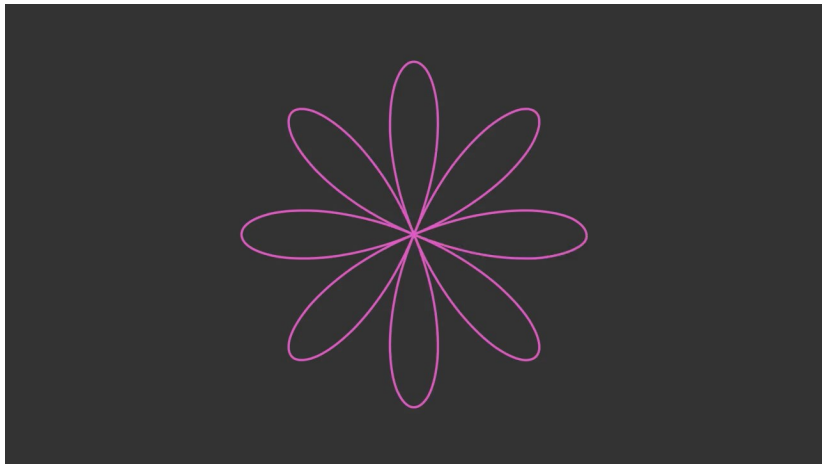
```
def f(t):  
    return np.array((np.sin(2*t), np.sin(3*t), 0))  
  
F = ParametricCurve(f, t_range=[0,TAU,0.1], color=BLUE)  
  
self.play(ShowCreation(F.scale(2))) # 放大兩倍
```

練習時間 ~ (ex10)

使用 `ParametricCurve` 畫參數方程式圖形。

參數方程式 $x = 3 \cos(4t) \cos(t)$, $y = 3 \cos(4t) \sin(t)$, $0 \leq t \leq 2\pi$.

動畫請參考 [Ex10.mp4](#)





動畫類別篇

ShowCreation(object) (基本動態創作物件的類別)

Uncreate(object) (動態銷毀物件，連物件定義都會刪除)

DrawBorderThenFill(object, stroke_width=2, run_time=2)

(動態先畫形狀，再填封閉區域顏色)

Write(object, stroke_width=2, run_time=2)

(動態先畫形狀，再填顏色，通常用在文字或數學式)

本篇類別均需搭配 `self.play()` 指令才會有動態效果，如果只想移除畫面上的某物件，應使用 `self.remove(object)`，若想清除畫面上所有物件，可使用 `self.clear()`。

例如：動態創造一個內部藍色，外框粉紅色的正三角形，然後動態銷毀它。

```
T = Triangle(color=PINK,fill_color=BLUE,fill_opacity=1)
self.play(Write(T))
self.play(Uncreate(T))
```

Transform 類別 - Transform

Transform(mobject, target_mobject)

(物件 mobject 動態變形成物件 target_mobject，但名字還是 mobject 的名字)

例如：A 紅圓，變形為 B 藍方形，但名字還是 A，所以 A 再變 C 黃三角。(example8)

```
A = Circle(fill_opacity=1)
B = Square(color=BLUE, fill_opacity=1).shift(RIGHT*2)
C = Triangle(color=YELLOW, fill_opacity=1)
self.play(Write(A))
self.wait()
self.play(Transform(A,B)) # A變B, 名字還是A, 形狀已成B
self.wait()
self.play(Transform(A,C)) # 原名A再變C
self.wait()
```

例如：A 式保留，A 式的複製品變形成 B 式。(example9)

```
A = Tex("A^2+B^2=C^2",color=BLUE).shift(UP*2)
B = Tex("A^2=C^2-B^2",color=PINK).next_to(A,DOWN)
self.play(Write(A))
self.wait()
self.play(Transform(A.copy(),B)) # A的複製品變形到B
```


ReplacementTransform(mobject, target_mobject)

(物件 mobject 動態變形成物件 target_mobject，名字也變成 target_mobject 的名字)

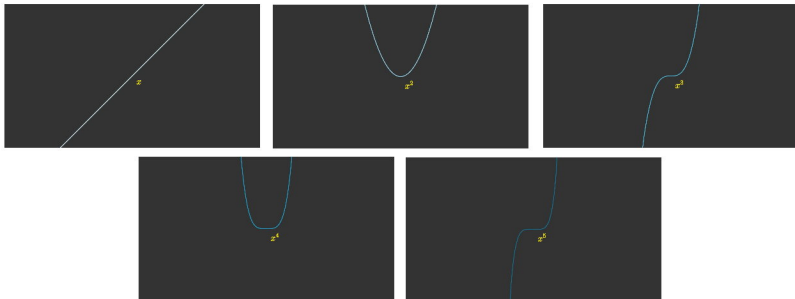
例如：A 紅圓，變形為 B 藍方形，B 藍方形再變 C 黃三角。(example10)

```
A = Circle(fill_opacity=1)
B = Square(color=BLUE, fill_opacity=1).shift(RIGHT*2)
C = Triangle(color=YELLOW, fill_opacity=1)
self.play(Write(A))
self.wait()
self.play(ReplacementTransform(A,B)) # A身心都變B
self.wait()
self.play(ReplacementTransform(B,C)) # B身心都變C
self.wait()
```

練習時間 ~ (ex11)

使用 `FunctionGraph` 畫函數 x ，並標上函數式 x 在原點右下方，然後用 `Transform` 或 `ReplacementTransform` 變形到 x^2 ，再變形到 x^3 、 x^4 、 x^5 等等。

動畫請參考 [Ex11.mp4](#)



附錄

附錄 (1) 物件導向概念簡介 - 類別

類別宣告 類別名稱 繼承 Scene 類別

```
class Example1(Scene):  
    def construct(self):  
        self.add(Circle())  
        self.wait()
```

建構函數

物件導向以**類別**實現擬人化物件概念。
包含**屬性(attribute)**與**方法(method)**。



- [Manim's Github webpage](#)
- [Manim's documentation](#)
- [Manim 中文教程文檔](#)
- [Welcome to Python.org](#)
- [MiKTeX](#)
- [FFmpeg](#)