

# Criando animações com

Manim 

Live de Python #243



## 1. Apresentação

e conceitos básicos

## 2. Cenas

Manipulando o contexto

## 3. Mobjects

Os objetos de cena

## 4. Animações

Dando vida aos objetos na cena



[picpay.me/dunossauro](https://picpay.me/dunossauro)



[apoia.se/livedepython](https://apoia.se/livedepython)



[pix.dunossauro@gmail.com](mailto:pix.dunossauro@gmail.com)



Ajude o projeto <3



Ademar Peixoto, Adilson Herculano, Adriano Ferraz, Alemão, Alexandre Harano, Alexandre Lima, Alexandre Takahashi, Alexandre Villares, Alex Lima, Alisson Souza, Alysson Oliveira, Ana Carneiro, Ana Padovan, Andre Azevedo, Andre Mesquita, Andre Paula, Aquiles Coutinho, Arnaldo Turque, Aslay Clevisson, Aurelio Costa, Bernardo At, Bernardo Fontes, Bruno Almeida, Bruno Barcellos, Bruno Batista, Bruno Freitas, Bruno Lopes, Bruno Ramos, Caio Nascimento, Carlos Ramos, Christiano Morais, Christian Semke, Damianth, Daniel Freitas, Daniel Wojcickoski, Danilo Boas, Danilo Segura, Danilo Silva, David Couto, David Kwast, Davi Goivinho, Davi Souza, Delton Porfiro, Denis Bernardo, Diego Farias, Diego Guimarães, Dilenon Delfino, Diogo Paschoal, Diogo Silva, Edgar, Eduardo Silveira, Eduardo Tolmasquim, Emerson Rafael, Eneas Teles, Ennio Ferreira, Erick Andrade, Érico Andrei, Everton Silva, Fabiano, Fabiano Tomita, Fabio Barros, Fábio Barros, Fabio Castro, Fabio Correa, Fábio Thomaz, Fabricio Biazotto, Fabricio Patrocinio, Felipe Augusto, Felipe Rodrigues, Fernanda Prado, Fernando Celmer, Firehouse, Flávio Meira, Francisco Faria, Francisco Neto, Francisco Silvério, Gabriel Espindola, Gabriel Mizuno, Gabriel Moreira, Gabriel Paiva, Gabriel Simonetto, Geanderson Costa, Geizelder, Gilberto Abrao, Giovanna Teodoro, Giuliano Silva, Guilherme Beira, Guilherme Felitti, Guilherme Gall, Guilherme Silva, Guionardo Furlan, Gustavo Suto, Harold Gautschi, Heitor Fernandes, Helvio Rezende, Higor Monteiro, Italo Silva, Janael Pinheiro, Jean Victor, Joelson Sartori, Jônatas Oliveira, Jônatas Silva, Jon Cardoso, Jorge Silva, José Gomes, Joseíto Júnior, Jose Mazolini, Josir Gomes, Juan Felipe, Juan Gutierrez, Juliana Machado, Julio Franco, Júlio Gazeta, Julio Silva, Kaio Peixoto, Kálita Lima, Kaneson Alves, Leandro Miranda, Leandro Silva, Leo Ivan, Leonardo Mello, Leonardo Nazareth, Leon Solon, Luancomputacao Roger, Lucas Adorno, Lucas Carderelli, Lucas Mendes, Lucas Nascimento, Lucas Pavelski, Lucas Schneider, Lucas Simon, Lucas Valino, Luciano Filho, Luciano Ratamero, Luciano Teixeira, Luis Alves, Luis Eduardo, Luiz Duarte, Luiz Lima, Luiz Paula, Luiz Perciliano, Mackilem Laan, Marcelo Araujo, Marcelo Campos, Marcio Moises, Marco Mello, Marcos Gomes, Maria Clara, Marina Passos, Mateus Lisboa, Mateus Ribeiro, Mateus Silva, Matheus Silva, Matheus Vian, Mauricio Nunes, Mírian Batista, Mlevi Lsantos, Moisés Ferreira, Murilo Viana, Natan Cervinski, Nathan Branco, Nicolas Teodosio, Otávio Carneiro, Patricia Minamizawa, Patrick Felipe, Paulo Tadei, Pedro Henrique, Pedro Pereira, Peterson Santos, Priscila Santos, Pydocs Pro, Pytonyc, Rafael Lopes, Rafael Romão, Raimundo Ramos, Ramayana Menezes, Regis Santos, Renato Oliveira, Renê Barbosa, Rene Bastos, Ricardo Silva, Richard Carvalho, Riverfount, Rjribeiro, Robson Maciel, Rodrigo Barretos, Rodrigo Freire, Rodrigo Oliveira, Rodrigo Quiles, Rodrigo Ribeiro, Rodrigo Vaccari, Rodrigo Vieira, Rogério Nogueira, Ronaldo R., Ronaldo Silveira, Rui Jr, Samanta Cicilia, Samuel Santos, Téó Calvo, Thaynara Pinto, Thiago Araujo, Thiago Borges, Thiago Curvelo, Thiago Souza, Tiago Minuzzi, Tony Dias, Tyrone Damasceno, Valcilon Silva, Valdir Tegen, Vcwild, Vicente Marcal, Vinícius Costa, Vinicius Stein, Walter Reis, William Vitorino, Willian Lopes, Wilson Duarte, Wilson Neto, Zeca Figueiredo



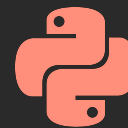
Obrigado você



e conceitos básicos

Apresen  
tação

# Manim Community Edition



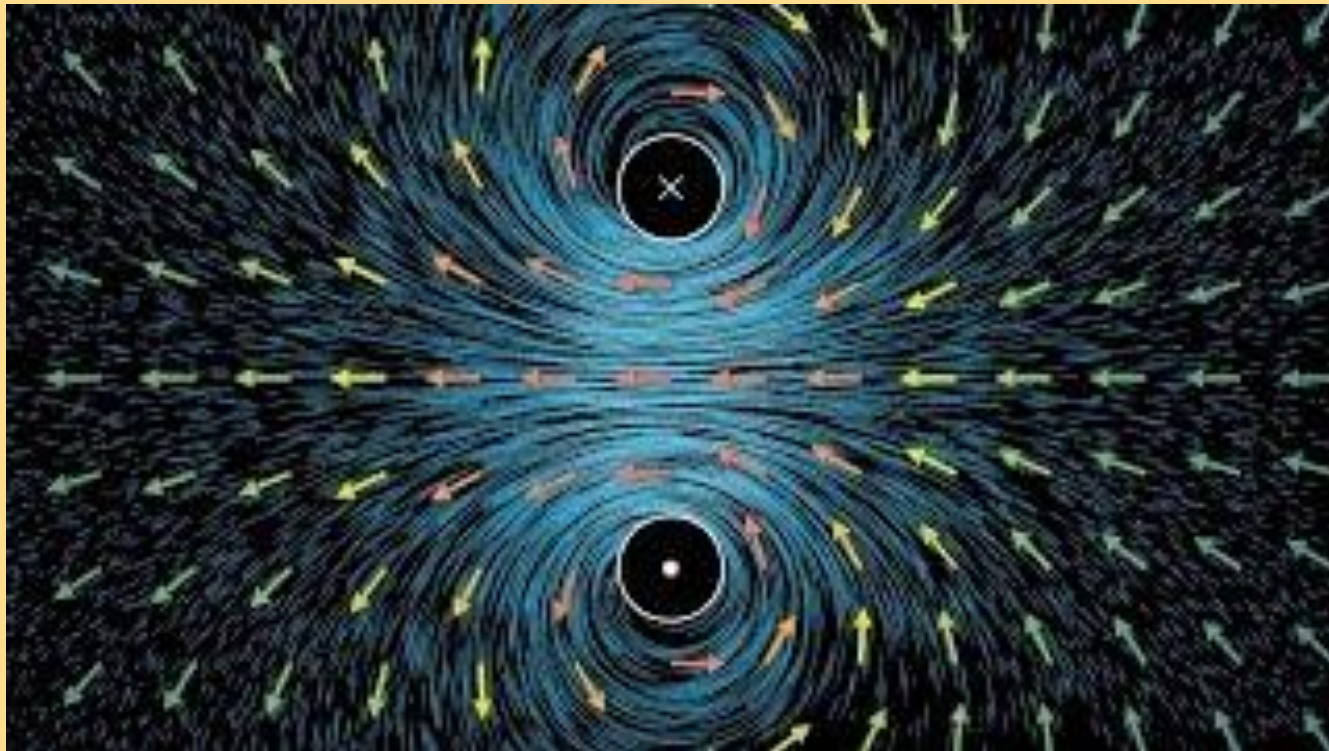
Uma biblioteca criada por Grant Sanderson, do canal 3 Blue 1 Brown em março de 2015 para criação de seus vídeos, como um projeto pessoal.

Em 2020 a comunidade decidiu fazer um fork da versão pessoal do Grant e criaram o **ManimCE**.

- Primeira release (oficial) em 2020
- Release atual: 0.17.3 — Lançada em maio de 2023
- Sob licença: MIT

<https://docs.manim.community/en/stable/faq/installation.html#different-versions>

# Um exemplo de conteúdo criado com Manim



<https://www.youtube.com/watch?v=rB83DpBJQsE>

pip install manim



Instalação





# Installing Manim locally

Manim is a Python library, and it can be [installed via pip](#). However, in order for Manim to work properly, some additional system dependencies need to be installed first. The following pages have operating system specific instructions for you to follow.

Manim requires Python version `3.7` or above to run.

## Hint

Depending on your particular setup, the installation process might be slightly different. Make sure that you have tried to follow the steps on the following pages carefully, but in case you hit a wall we are happy to help: either [join our Discord](#), or start a new Discussion [directly on GitHub](#).

- [Windows](#)
  - [Required Dependencies](#)
  - [Optional Dependencies](#)
  - [Working with Manim](#)
- [macOS](#)
  - [Required Dependencies](#)
  - [Optional Dependencies](#)
  - [Working with Manim](#)
- [Linux](#)
  - [Required Dependencies](#)
  - [Optional Dependencies](#)
  - [Working with Manim](#)

<https://docs.manim.community/en/stable/installation.html>

# Conceitos básicos



O Manim pode parecer complicado de início, então gostaria de desmembrar alguns componentes básicos antes de adentrarmos na biblioteca:

- Cena
- Objetos
- Animações
- CLI

Existem mais componentes, mas para iniciar *\*precisamos\** entender estes!

# Cena



A Cena é onde a animação irá acontecer. Ela é quem adiciona os objetos, liga as animações, pausa, adiciona legendas, etc. A cena é **construída**.

```
1  from manim import Scene
2
3
4  class MyScene(Scene):
5      def construct(self):
6          ...
```



Vamos pensar na cena como a base de todo o vídeo, um canvas, ou uma "tela", onde colocaremos os objetos



# Objetos



Para compor uma cena, precisamos adicionar objetos a ela.

Existem diversos tipos de objetos, que veremos mais adiante, mas vamos iniciar com duas formas:

```
1  from manim import Scene  # Scene
2  from manim import Square, Circle  # MObjects
3
4  class MyScene(Scene):
5      def construct(self):
6          quadrado = Square()
7          circulo = Circle()
```

# Animações [exemplo\_00.py]



Para que esses objetos sejam exibidos na tela, precisamos animar sua construção na cena. Usaremos duas animações simples `Create` e `Transform`.

```
1  from manim import Scene # Scene
2  from manim import Square, Circle # MObjects
3  from manim import Create, Transform # Animations
4
5  class MyScene(Scene):
6      def construct(self):
7          quadrado = Square()
8          circulo = Circle()
9
10         self.play(Create(quadrado))
11         self.play(Transform(quadrado, circulo))
```

# Entendendo o relacionamento



```
1  from manim import Scene # Scene
2  from manim import Square, Circle # MObjects
3  from manim import Create, Transform # Animations
4
5  class MyScene(Scene):
6      def construct(self):
7          quadrado = Square()
8          circulo = Circle()
9
10         self.play(Create(quadrado))
11         self.play(Transform(quadrado, circulo))
```

# Entendendo o relacionamento



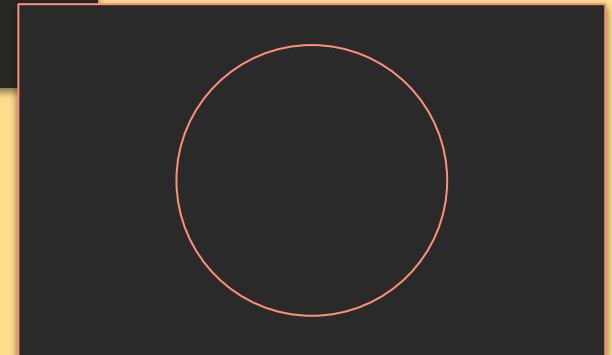
```
1  from manim import Scene # Scene
2  from manim import Square, Circle # MObjects
3  from manim import Create, Transform # Animations
4
5  class MyScene(Scene):
6      def construct(self):
7          quadrado = Square()
8          circulo = Circle()
9
10         self.play(Create(quadrado))
11         self.play(Transform(quadrado, circulo))
```



# Entendendo o relacionamento



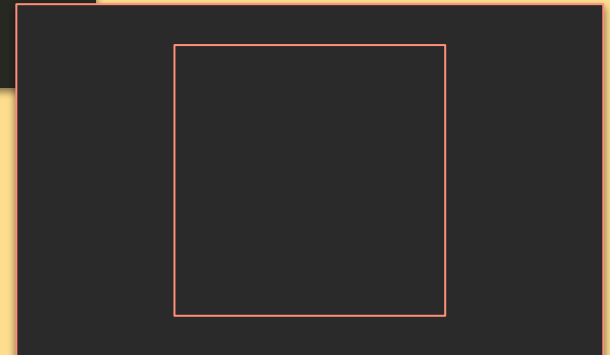
```
1  from manim import Scene # Scene
2  from manim import Square, Circle # MObjects
3  from manim import Create, Transform # Animations
4
5  class MyScene(Scene):
6      def construct(self):
7          quadrado = Square()
8          circulo = Circle()
9
10         self.play(Create(quadrado))
11         self.play(Transform(quadrado, circulo))
```



# Entendendo o relacionamento



```
1  from manim import Scene # Scene
2  from manim import Square, Circle # MObjects
3  from manim import Create, Transform # Animations
4
5  class MyScene(Scene):
6      def construct(self):
7          quadrado = Square()
8          circulo = Circle()
9
10         self.play(Create(quadrado))
11         self.play(Transform(quadrado, circulo))
```



# O CLI



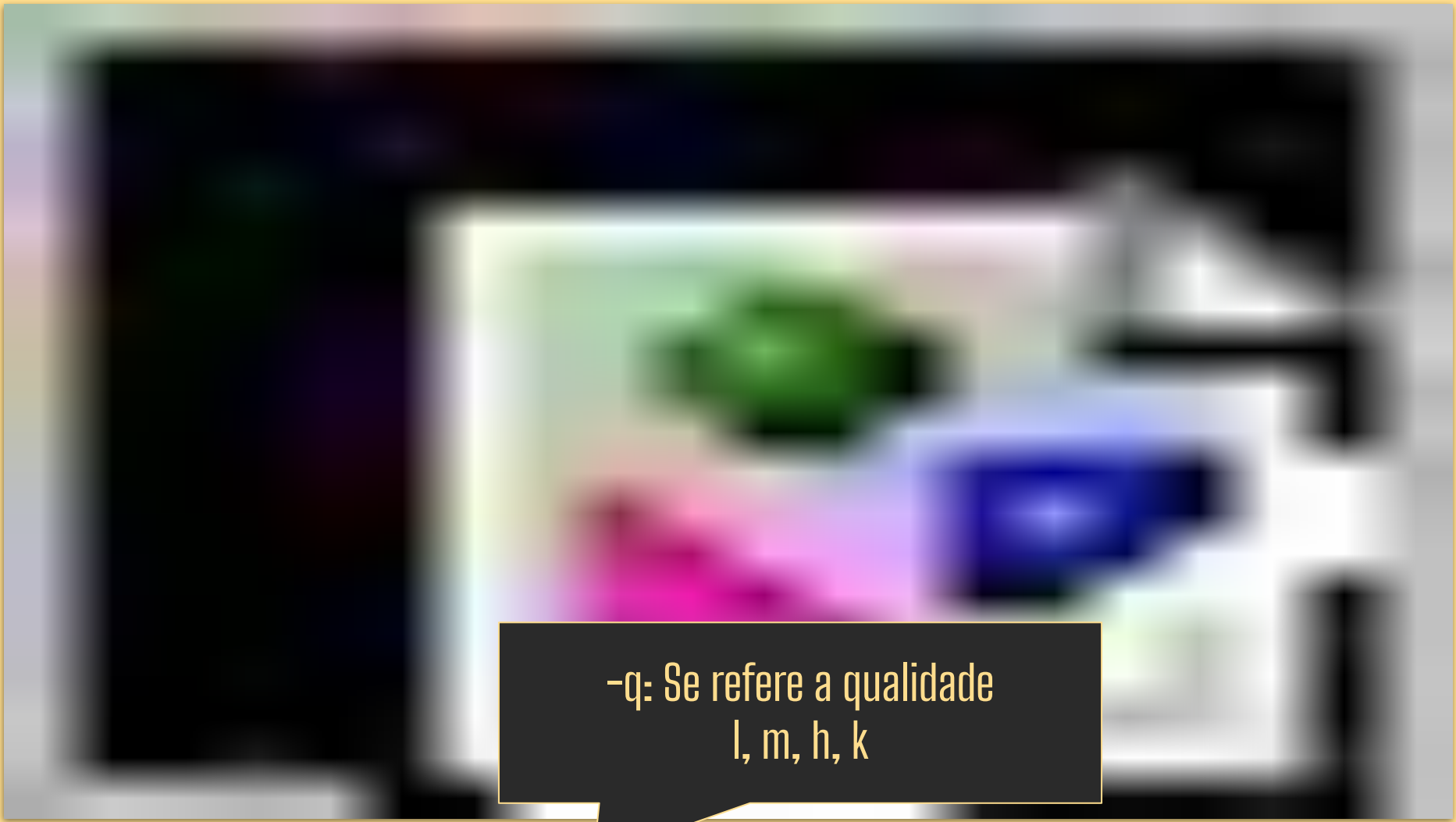
Quando instalamos o manim, ele fornece uma ferramenta de linha de comando.

Basicamente o que precisamos agora é visualizar a nossa animação, então:

```
manim -p exemplo_00.py  
# P: Preview
```



```
manim -p -q k exemplo_00.py --format=gif
```



-q: Se refere a qualidade  
l, m, h, k

```
manim -p -q k exemplo_00.py --format=gif
```

# Cenas

Manipulando o  
contexto

# Os objetos Scenes



O objeto de cena é o componente principal do Manim. É ele que controla toda a animação, objetos de cena, legendas, áudio, câmera e o canvas.

Vamos pelos métodos de execução primeiro:

- **play()**: Inicia alguma animação
  - **pause()**: Pausa a cena toda
  - **wait()**: Espera por X segundos algo que está sendo animado
- 
- **add()**: Adiciona algum objeto na cena
  - **remove()**: Remove algum objeto da cena
  - **bring\_toback/front()**: Coloca os objetos atrás ou na frente

```
1  from manim import Scene
2  from manim.animation.creation import Create, Uncreate, SpiralIn
3  from manim.animation.fading import FadeIn, FadeOut
4  from manim.camera.camera import Camera
5  from manim.constants import DOWN
6  from manim.mobject.geometry.polygram import Rectangle
7  from manim.mobject.text.text_mobject import Text
8
9
10 class Intro(Scene):
11     camera: Camera
12
13     def construct(self):
14         text = Text('Live de Python', color='#763bb1')
15         text2 = Text('#243 - Manim', color='#763bb1').shift(DOWN).scale(0.5)
16         retangulo = Rectangle(color='#ffc410', fill_opacity=1).scale(2)
17
18         self.camera.background_color = '#2f304d'
19
20         self.add(text)
21         self.bring_to_back(retangulo)
22         self.play(Create(retangulo), FadeIn(text2))
23         self.pause(3)
24         self.play(Uncreate(text), Uncreate(retangulo), FadeOut(text2))
25         self.wait()
```



# Live de Python

```
manim -p -q m exemplo_01.py --format=gif
```

```
1 from manim import Scene
2 from manim.animation.creation import Create, Uncreate, SpiralIn
3 from manim.animation.fading import FadeIn, FadeOut
4 from manim.camera.camera import Camera
5 from manim.constants import DOWN
6 from manim.mobject.geometry.polygram import Rectangle
7 from manim.mobject.text.text_mobject import Text
8
9
10 class Intro(Scene):
11     camera: Camera
12
13     def construct(self):
14         text = Text('Live de Python', color='#763bb1')
15         text2 = Text('#243 - Manim', color='#763bb1').s
16         retangulo = Rectangle(color='#ffc410', fill_opa
17
18         self.camera.background_color = '#2f304d'
19
20         self.add(text)
21         self.bring_to_back(retangulo)
22         self.play(Create(retangulo), FadeIn(text2))
23         self.pause(3)
24         self.play(Uncreate(text), Uncreate(retangulo),
25         self.wait()
```

## Manim Community v0.17.3

🔍 Search

[Example Gallery](#)

[Installation](#)

[Tutorials & Guides](#)

[Reference Manual](#)

[Animations](#)

[Cameras](#)

[Configuration](#)

[Mobjects](#)

[Scenes](#)

[Utilities and other modules](#)

# Outros tipos de cena



Existem diferentes tipos de cena.

- Álgebra: Vector e Linear
- Movimentação: Moving, Zoomed
- 3D: ThreeD e SpecialThreeD

## Scenes



# Um exemplo com a câmera 3D [exemplo\_02.py]

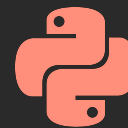


```
1 class Intro(ThreeDScene):
2     def construct(self):
3         text = Text('Live de Python', color='#763bb1')
4         text2 = Text('#243 - Manim', color='#763bb1').shift(DOWN).scale(0.5)
5         retangulo = Rectangle(color='#ffc410', fill_opacity=1).scale(2)
6
7         self.camera.background_color = '#2f304d'
8         self.add(text)
9         self.bring_to_back(retangulo)
10        self.play(Create(retangulo), FadeIn(text2))
11
12        self.move_camera(phi=75 * DEGREES)
13        self.wait()
14        self.move_camera(phi=0 * DEGREES)
15
16        self.play(Uncreate(text), Uncreate(retangulo), FadeOut(text2))
17        self.wait()
```

# Live de Python

```
manim -p -q m_exemplo_02.py --format=gif
```

# Multiplas cenas



O manim permite que você tenha mais de uma cena no mesmo arquivo, para renderizar cada uma delas, você pode chamar a cena específica no CLI

```
manim -q m arquivo.py NomeDaCena
```

Se quiser unir duas cenas em um único redeer



```
1 class AllScenes(Scene):  
2     def construct(self):  
3         CenaA.construct(self)  
4         CenaB.construct(self)
```

# Mobj ects

Os objetos em cena!



# Objetos de cena



Dentro no Manim existe uma grande variedade de objetos que podem ser usados. O Manim os divide em 15 categorias. Das quais exploraremos somente algumas:

- **Text:** Objetos de texto, Código, Números e Latex
- **Geometry:** Linhas, Shapes e poligonos
- **Mobjects:** Alguns elementos com funções específicas
  - Grupos verticais e horizontais
- **Types:** Categoria de objetos específicos
  - Imagens são um grande representante dessa categoria

# Objetos de cena



Dentro no Manim existe uma grande variedade de objetos usados. O Manim os divide em 15 categorias. Das quais eu apresento somente algumas:

- **Text:** Objetos de texto, Código, Números e Latex
- **Geometry:** Linhas, Shapes e poligonos
- **Mobjects:** Alguns elementos com funções específicas
  - Grupos verticais e horizontais
- **Types:** Categoria de objetos específicos
  - Imagens são um grande representante dessa categoria

Mobjects	^
frame	v
geometry	v
graph	v
graphing	v
logo	v
matrix	v
mobject	v
svg	v
table	v
text	v
three_d	v
types	v
utils	
value_tracker	v
vector_field	v

# Atributos padrões para Mobjects



Podemos controlar alguns atributos dos objetos:

- Tamanho: `.scale()`
- Cor:
  - Linha: `stroke_color`
  - Preenchimento: `fill_color`
- Opacidade:
  - Linha: `stroke_opacity`
  - Preenchimento: `fill_opacity`
- font (texto): `font`
- `next_to`: Posição em relação a outro objeto

# Aplicando os atributos [exemplo\_03.py]

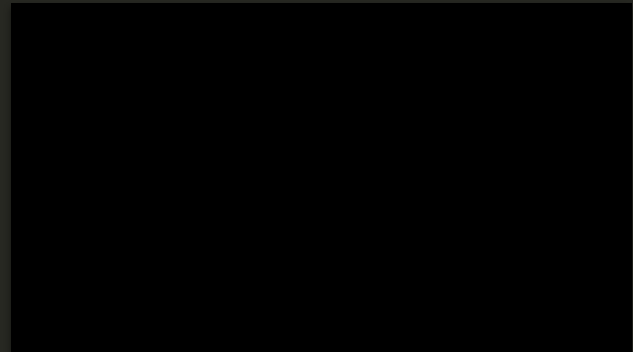


```
1  class MyScene(Scene):  
2      def construct(self):  
3          c = Circle(  
4              stroke_color=BLUE,  
5              stroke_opacity=1,  
6              fill_color=RED,  
7              fill_opacity=1,  
8          ).scale(3)  
9  
10         t = Text('LIVE DE PYTHON!', font='Hansief', color=GOLD).scale(2)
```

# Aplicando os atributos [exemplo\_03.py]



```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(
4             stroke_color=BLUE,
5             stroke_opacity=1,
6             fill_color=RED,
7             fill_opacity=1,
8         ).scale(3)
9
10        t = Text('LIVE DE PYTHON!', font='Hansief', color=GOLD).scale(2)
```



# Grupos [exemplo\_04.py]



Em alguns momentos queremos que N objetos sejam manipulados ao mesmo tempo. Existe um Mobject para agrupar outros objetos. O VGroup:

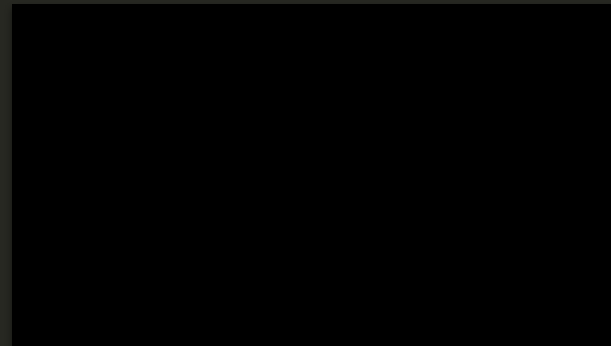
```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(
4             stroke_color=BLUE,
5             stroke_opacity=1,
6             fill_color=RED,
7             fill_opacity=1,
8         ).scale(3)
9
10        t = Text('LIVE DE PYTHON!', font='Hansief', color=GOLD).scale(2)
11
12        g = VGroup(t, c)
13
14        self.play(Create(g))
```

# Grupos [exemplo\_04.py]



Em alguns momentos queremos que N objetos sejam manipulados ao mesmo tempo. Existe um Mobject para agrupar outros objetos. O VGroup:

```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(
4             stroke_color=BLUE,
5             stroke_opacity=1,
6             fill_color=RED,
7             fill_opacity=1,
8         ).scale(3)
9
10        t = Text('LIVE DE PYTHON!', font='Hansief', color=GOLD).scale(2)
11
12        g = VGroup(t, c)
13
14        self.play(Create(g))
```



# Posicionamento [exemplo\_05.py]



Quanto temos mais de um objeto na tela, precisamos que eles não ocupem o mesmo espaço. Por padrão, os objetos são adicionados sempre no centro da cena.

Os objetos são sempre criados em relação a outro objeto. Dado que temos um no centro. O método **next\_to**(objeto, **direção**) pode nos ajudar com isso:

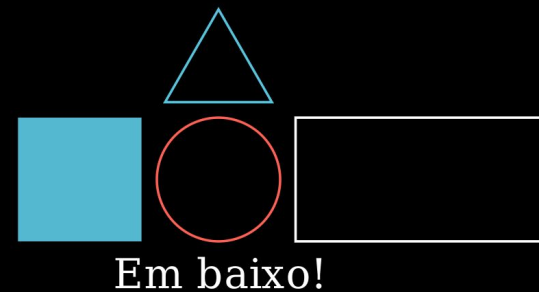
```
1 class MyScene(Scene):
2     def construct(self):
3         ci = Circle()
4         cu = Cube().next_to(ci, LEFT)
5         re = Rectangle().next_to(ci, RIGHT)
6         tr = Triangle().next_to(ci, UP)
7         te = Text('Em baixo!').next_to(ci, DOWN)
8
9         g = VGroup(ci, cu, re, tr, te)
10
11         self.add(g)
```



# Posicionamento [exemplo\_05.py]



```
1 class MyScene(Scene):
2     def construct(self):
3         ci = Circle()
4         cu = Cube().next_to(ci, LEFT)
5         re = Rectangle().next_to(ci, RIGHT)
6         tr = Triangle().next_to(ci, UP)
7         te = Text('Em baixo!').next_to(ci, DOWN)
8
9         g = VGroup(ci, cu, re, tr, te)
10
11         self.add(g)
```



Dando vida a cena!

Anima  
ções

# Atributos para MObjects



Existem dois tipos de animações no Manim: as animações de cena e as animações de objetos:

- 15 grupos de animações de cena
- Os atributos dos objetos podem ser animados
  - cor
  - scala
  - next
  - shift e etc..

# Atributos para MObjects



Existem dois tipos de animações no Manim: as animações de objetos e as animações de objetos:

- 15 animações de cena
- Os atributos dos objetos podem ser animados
  - cor
  - escala
  - next
  - shift e etc..

## Reference Manual

### Animations

animation

changing

composition

creation

fading

growing

indication

movement

numbers

rotation

specialized

speedmodifier

transform

transform\_matching\_parts

# Animações de cena



São as animações que já usamos, como Create, Uncreate, Fadein, FadeOut, ...

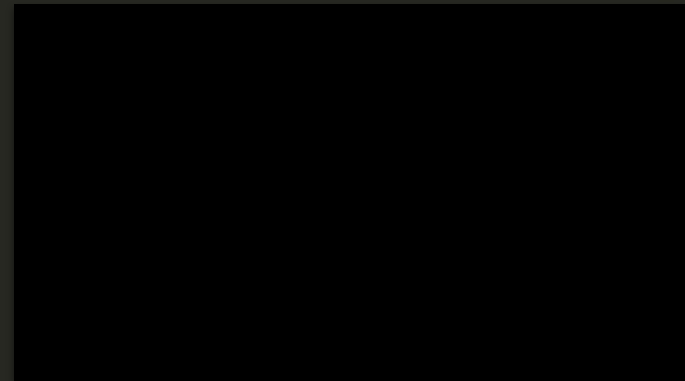
```
1  class MyScene(Scene):
2      def construct(self):
3          c = Cube()
4          text = Text('Batata!')
5
6          animations = [
7              Create(c), FadeOut(c), FadeIn(c), Rotate(c),
8              FadeTransform(c, text), Uncreate(text),
9          ]
10
11      for an in animations:
12          self.play(an)
13          self.wait()
```

# Animações de cena



São as animações que já usamos, como Create, Uncreate, Fadein, FadeOut, ...

```
1  class MyScene(Scene):
2      def construct(self):
3          c = Cube()
4          text = Text('Batata!')
5
6          animations = [
7              Create(c), FadeOut(c), FadeIn(c), Rotate(c),
8              FadeTransform(c, text), Uncreate(text),
9          ]
10
11      for an in animations:
12          self.play(an)
13          self.wait()
```



# Animações de objetos



Usando o método `animate` dos objetos, podemos fazer com que as cenas toquem a animação das propriedades do objeto

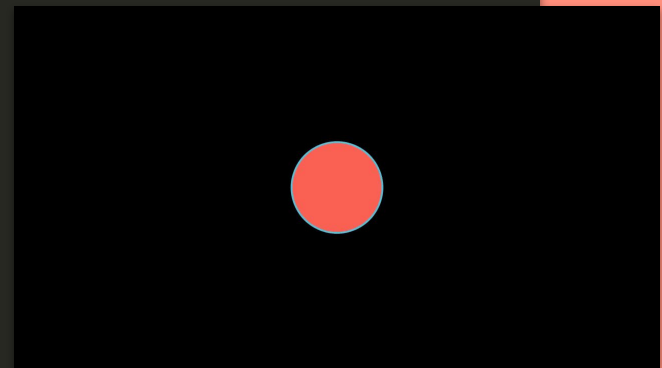
```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(
4             stroke_color=BLUE, fill_color=RED,
5             fill_opacity=1, stroke_opacity=1
6         )
7         d = Cube()
8
9         self.add(c)
10
11         self.play(c.animate.scale(2))
12         self.play(c.animate.become(d))
13         self.play(c.animate.flip())
14         self.play(c.animate.rotate(90))
15         self.play(c.animate.scale(0))
16         self.wait()
```

# Animações de objetos



Usando o método `animate` dos objetos, podemos fazer com que as cenas toquem a animação das propriedades do objeto

```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(
4             stroke_color=BLUE, fill_color=RED,
5             fill_opacity=1, stroke_opacity=1
6         )
7         d = Cube()
8
9         self.add(c)
10
11         self.play(c.animate.scale(2))
12         self.play(c.animate.become(d))
13         self.play(c.animate.flip())
14         self.play(c.animate.rotate(90))
15         self.play(c.animate.scale(0))
16         self.wait()
```





```
1 class MyScene(Scene):
2     def construct(self):
3         c = Circle(fill_color=RED, fill_opacity=1)
4         d = Cube()
5         e = Rectangle()
6         t = Text('Live de PythOn!!!', font='Hansief')
7         t2 = Text(
8             '#243 - Animações cOm Manim', font='Hansief'
9         ).scale(.6).next_to(t, DOWN)
10
11        self.play(Create(c))
12
13        self.play(c.animate().scale(2))
14        self.play(c.animate.become(d))
15
16        self.play(c.animate().flip())
17        self.play(c.animate().rotate(90))
18        self.play(c.animate().become(t))
19        self.wait()
20
21        self.play(Create(e.scale(2)), Create(t2))
22        self.wait()
23        self.play(
24            e.animate().scale(2.5).set_color(RED).flip(),
25            Uncreate(c),
26            Uncreate(t2)
27        )
28        self.wait()
```

— □ ×

```
1 class MyScene(Scene):  
2     def construct(self):
```

```
26         Uncreate(t2)  
27     )  
28     self.wait()
```



[picpay.me/dunossauro](https://picpay.me/dunossauro)



[apoia.se/livedepython](https://apoia.se/livedepython)



[pix.dunossauro@gmail.com](mailto:pix.dunossauro@gmail.com)



Ajude o projeto <3

