

---

# **Flying Circus Documentation**

***Release 0.1.0***

**Carlo E. T. Oliveira**

July 11, 2016



|          |                                             |           |
|----------|---------------------------------------------|-----------|
| <b>1</b> | <b>Sumário</b>                              | <b>3</b>  |
| 1.1      | Como brincar com os desafios . . . . .      | 3         |
| 1.2      | Criando uma Câmara com Constantes . . . . . | 3         |
| 1.3      | Criando uma Câmara com Listas . . . . .     | 4         |
| 1.4      | Criando uma Câmara com Dicionário . . . . . | 5         |
| 1.5      | Uma Câmara Instável . . . . .               | 7         |
| 1.6      | Uma Câmara Muito Instável . . . . .         | 8         |
| 1.7      | Uma Câmara Extremamente Instável . . . . .  | 9         |
| 1.8      | Primeiro Cenário do Jogo . . . . .          | 11        |
| 1.9      | Montando a Cena com Ladrilhos . . . . .     | 11        |
| 1.10     | Cena com Ladrilhos Girados . . . . .        | 14        |
| 1.11     | Cena com Mais Ladrilhos Girados . . . . .   | 17        |
| 1.12     | Cena com Muitos Ladrilhos Girados . . . . . | 19        |
| 1.13     | Colocando Personagens . . . . .             | 21        |
| 1.14     | Colocando Mais Personagens . . . . .        | 22        |
| 1.15     | Animando Personagens . . . . .              | 24        |
| 1.16     | Movimentando Personagens . . . . .          | 25        |
| <b>2</b> | <b>Índices e Tabelas</b>                    | <b>29</b> |



Aqui vamos ter uma introdução rápida de como programar jogos para Web usando Python. Na verdade vamos usar o Brython que é o Python que funciona dentro de um navegador web como o Firefox.



---

## Sumário

---

### 1.1 Como brincar com os desafios

Entre na plataforma <nome do seu projeto>.is-by.us. Clique no projeto determinado pelo instrutor. Vamos começar importando o circus para criar um jogo. Você faz a sua implementação e depois invoca o circus para conferir o resultado.

```
from _spy.circus.circus import circus
# faça aqui a sua implementação do desafio
if __name__ == "__main__":
    #circus(<ponha aqui o número do desafio e descomente a linha>, <parâmetro indicado>)
```

---

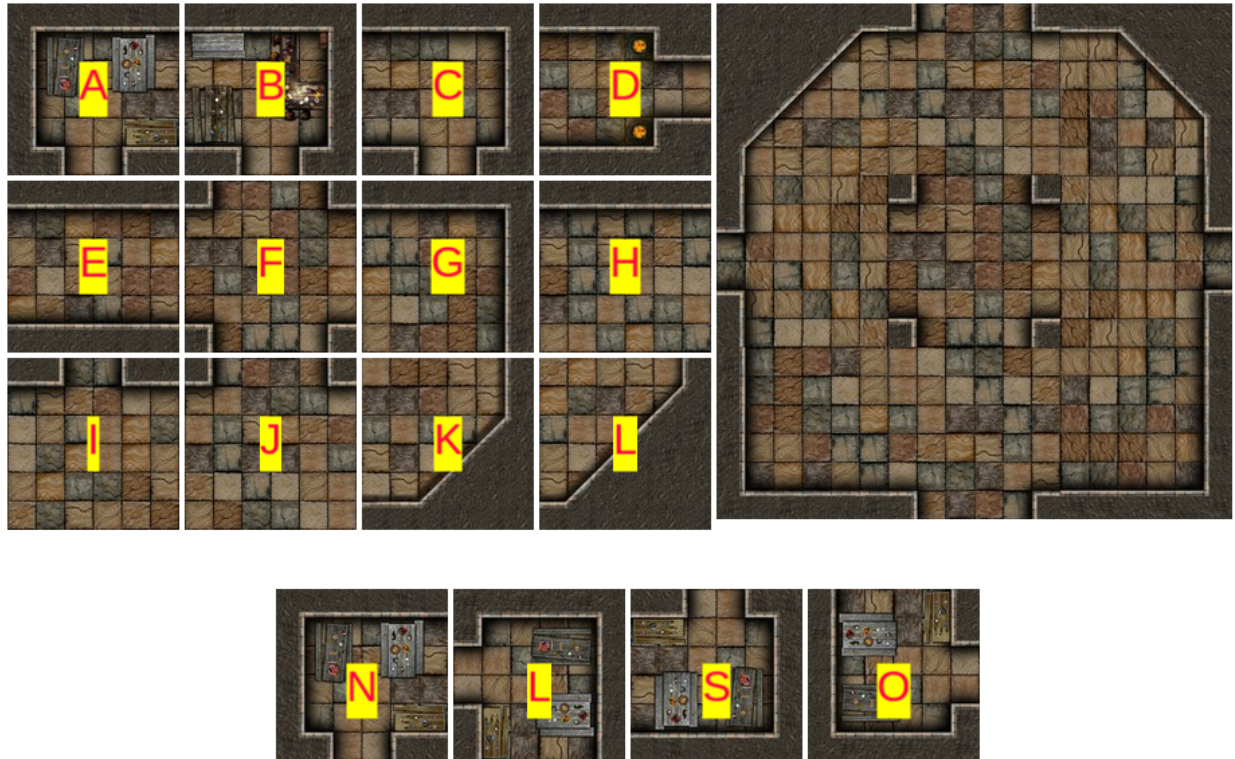
**Note:** Procure ser cooperativo com a sua equipe.

---

### 1.2 Criando uma Câmara com Constantes

Uma constante é um valor que não se modifica ao longo de um programa. Em Python a constante é escrita com todas as letras maiúsculas como no nome TOPO\_ESQUERDA abaixo.

Use os ladrilhos nomeados de A a L para montar a câmara mostrada à direita.



```
from _spy.circus.circus import circus

TOPO_ESQUERDA = "AN"
TOPO_DIREITA = "AN"
TOPO_CENTRO = "AN"
MEIO_ESQUERDA, CENTRO, MEIO_DIREITA = "AN", "AN", "AN"
FUNDO_ESQUERDA, FUNDO_CENTRO, FUNDO_DIREITA = "AN", "AN", "AN"

# O comando abaixo voce vai entender no próximo desafio
circus(1, [[TOPO_ESQUERDA, TOPO_CENTRO, TOPO_DIREITA], [MEIO_ESQUERDA, CENTRO,
    MEIO_DIREITA], [FUNDO_ESQUERDA, FUNDO_CENTRO, FUNDO_DIREITA]])
```

**Note:** No texto “AN” a primeira letra determina o ladriho e a segunda se está girada para Norte, Leste, Sul ou Oeste.

## 1.3 Criando uma Câmara com Listas

Uma lista é um conjunto de coisas, pode ser um conjunto de números, letras, palavras ou qualquer outro objeto. Em Python a lista é escrita assim: [*<uma coisa>*, *<outra coisa>*].

Use os ladrilhos nomeados de A a L para montar a câmara mostrada abaixo, consulte o exercício anterior.

```
from _spy.circus.circus import circus

MASMORRA = [ [ "AN", "AN", "AN", "AN", "AN", "AN"],
    [ "AN", "AN", "AN", "AN", "AN", "AN"],
    [ "AN", "AN", "AN", "AN", "AN", "AN"],
```



```
[ "AN", "AN", "AN", "AN", "AN", "AN" ],  
[ "AN", "AN", "AN", "AN", "AN", "AN" ]  
]  
  
circus(2, MASMORRA)
```

---

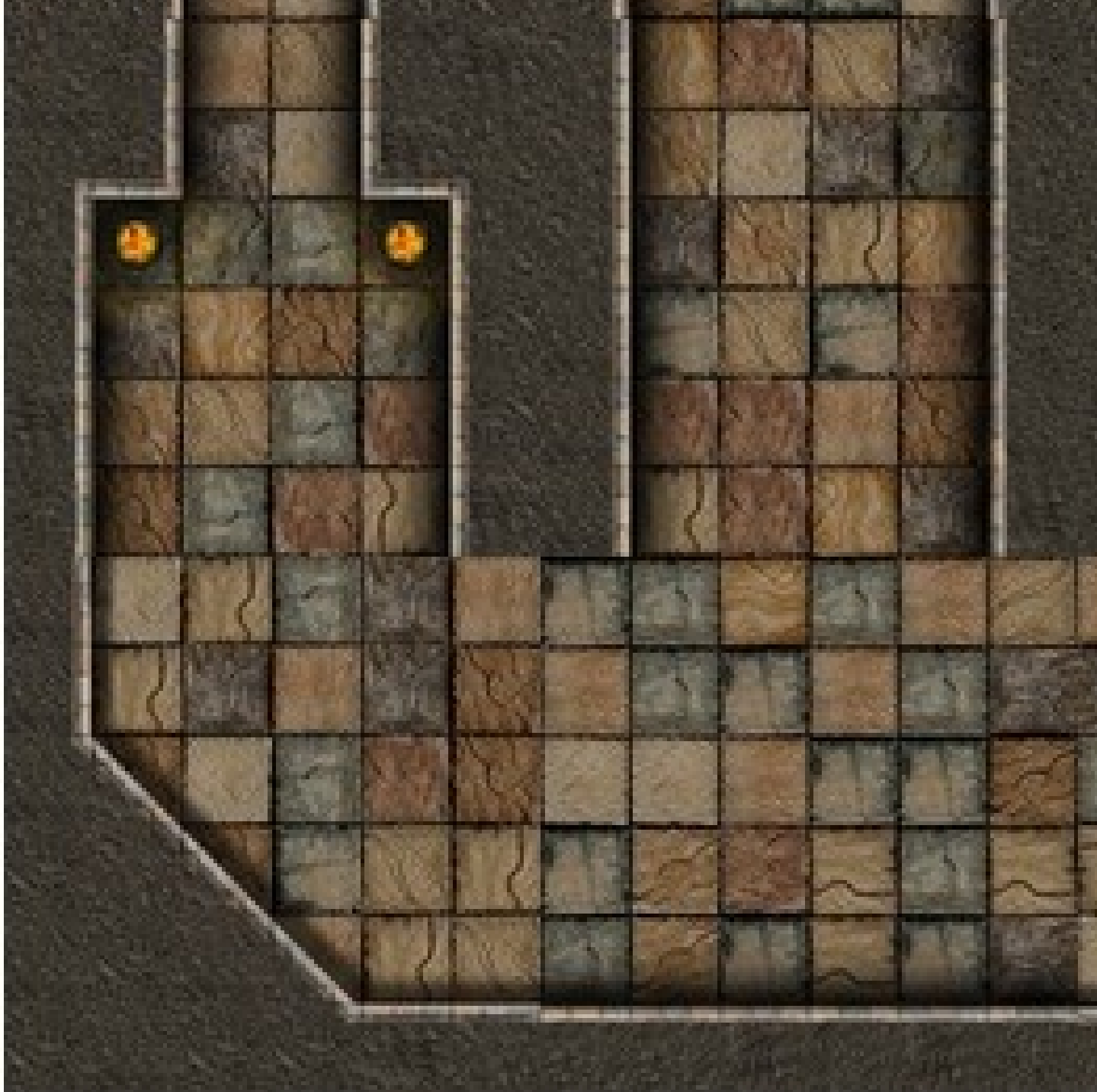
**Note:** No texto “AN” a primeira letra determina o ladriho e a segunda se está girada para Norte, Leste, Sul ou Oeste.

---

## 1.4 Criando uma Câmara com Dicionário

Uma lista é um conjunto de coisas, pode ser um conjunto de números, letras, palavras ou qualquer outro objeto. Em Python a lista é escrita assim:  $\{<umnome: umvalo>, <outronome: outrovalor>\}$ .

Use os ladrilhos nomeados de A a L para montar a câmara mostrada abaixo, consulte o exercício A. Descubra quais posições os nomes misteriosos indicam.



```
from _spy.circus.circus import circus

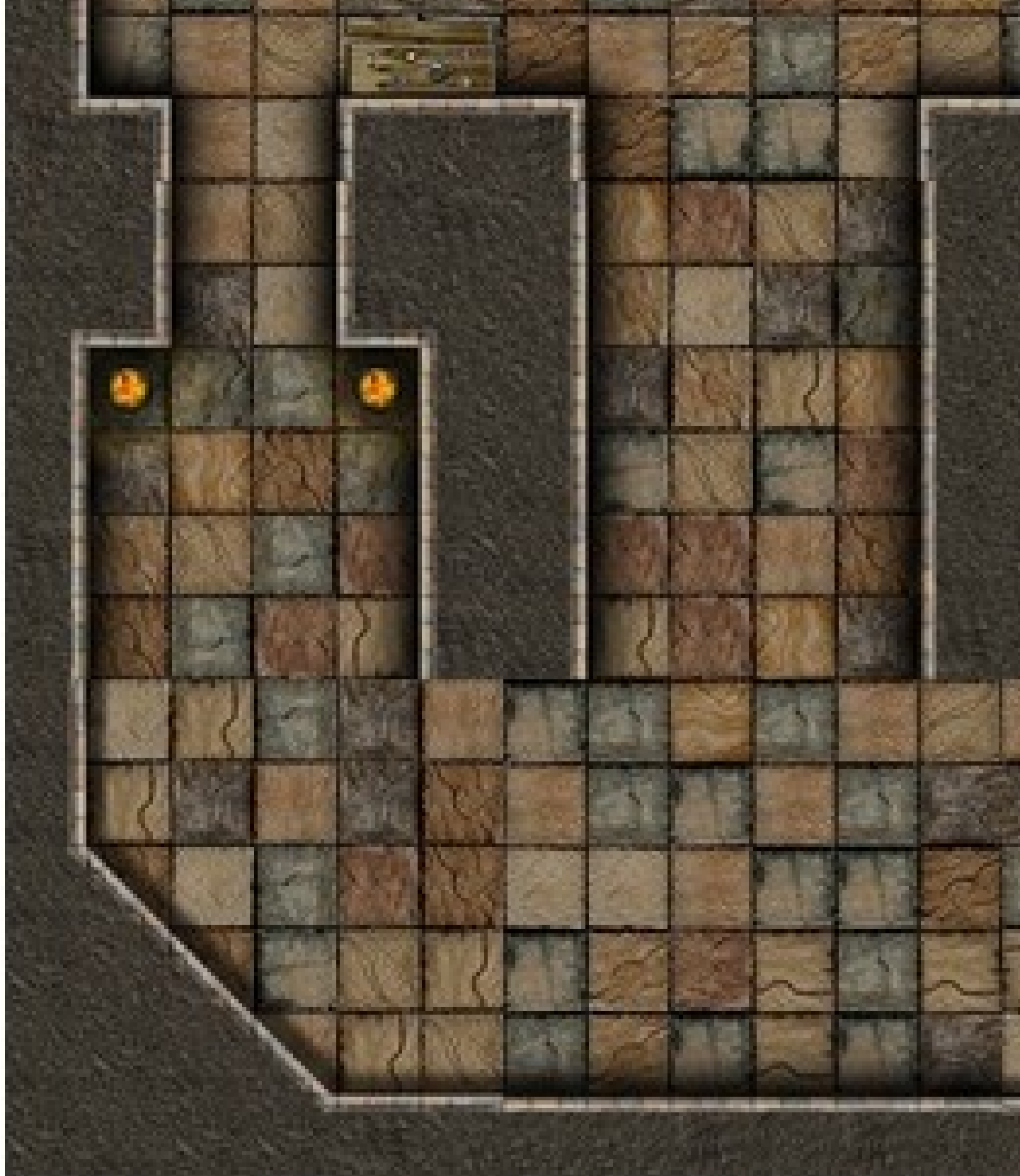
MASMORRA = {'Cahuitz': 'AN', 'Cauha': 'AN', 'Coycol': 'AN',
             'Huatlya': 'AN', 'Micpe': 'AN', 'Nenea': 'AN',
             'Pallotl': 'AN', 'Tetlah': 'AN', 'Zitllo': 'AN'}

circus(3, MASMORRA)
```

---

**Note:** No texto “AN” a primeira letra determina o ladrilho e a segunda se está girada para Norte, Leste, Sul ou Oeste.

---



```
from _spy.circus.circus import circus

MASMORRA = {'Cahuitz': 'AN', 'Cauha': 'AN', 'Coycol': 'AN',
             'Huatlya': 'AN', 'Micpe': 'AN', 'Nenea': 'AN',
             'Pallotl': 'AN', 'Tetlah': 'AN', 'Zitllo': 'AN'}
```



```
from _spy.circus.circus import circus

MASMORRA = {'Cahuitz': 'AN', 'Cauha': 'AN', 'Coycol': 'AN',
            'Huatlya': 'AN', 'Micpe': 'AN', 'Nenea': 'AN',
            'Pallotl': 'AN', 'Tetlah': 'AN', 'Zitllo': 'AN'}

circus(5, MASMORRA)
```

---

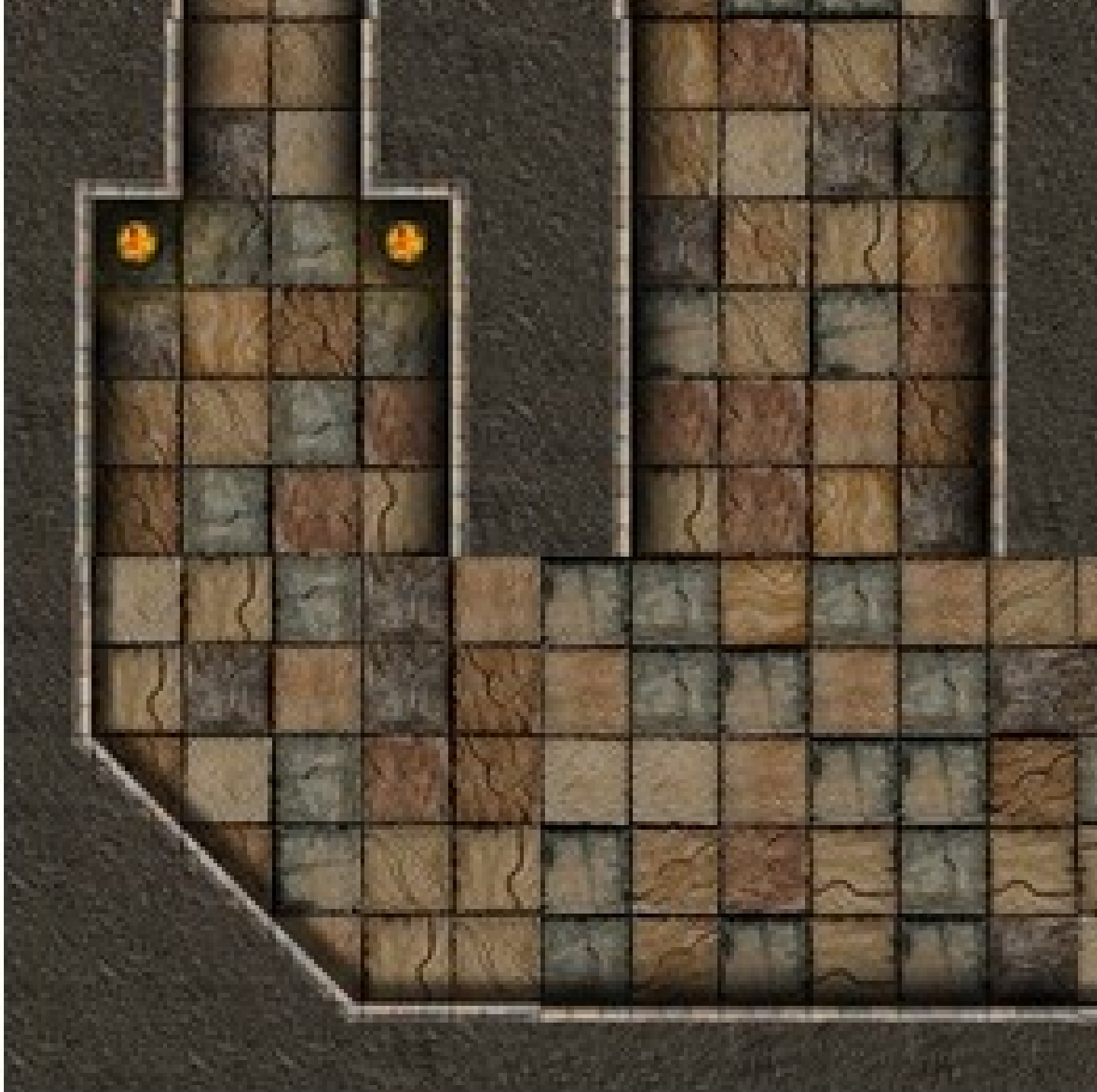
**Note:** No texto “AN” a primeira letra determina o ladrilho e a segunda se está girada para Norte, Leste, Sul ou Oeste.

---

## 1.7 Uma Câmara Extremamente Instável

Esta câmara embaralha as posições e as direções dependem das posições. Se os ladrilhos não forem colocados corretamente tudo será embaralhado. Você terá que usar uma estratégia para montar os ladrilhos antes que eles embaralhem.

Use os ladrilhos nomeados de A a L para montar a câmara mostrada abaixo, consulte o exercício A. Descubra quais posições os nomes misteriosos indicam.



```
from _spy.circus.circus import circus

MASMORRA = {'Cahuitz': 'AN', 'Cauha': 'AN', 'Coycol': 'AN',
            'Huatlya': 'AN', 'Micpe': 'AN', 'Nenea': 'AN',
            'Pallotl': 'AN', 'Tetlah': 'AN', 'Zitllo': 'AN'}

circus(6, MASMORRA)
```

---

**Note:** No texto “AN” a primeira letra determina o ladrilho e a segunda se está girada para Norte, Leste, Sul ou Oeste.

---

## 1.8 Primeiro Cenário do Jogo

Vamos começar importando o módulo Circus para criar um jogo baseado na biblioteca Phaser. Vamos criar uma organização em Python chamada *class* que vai ter duas operações *preload* e *create*. O objetivo é mostrar a imagem abaixo:

```
from _spy.circus.game import Circus

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""

    def preload(self):
        """Aqui no preload carregamos os recursos usados no jogo, neste caso a imagem masmorra"""
        self.image("fundo", "http://<descubra um jeito de achar a url que vai ser posta aqui>")

    def create(self):
        """Aqui colocamos a imagem masmorra na tela do jogo"""
        self.sprite("fundo")

if __name__ == "__main__":
    Jogo()
```

**Note:** Ainda é um programa bem simples.

## 1.9 Montando a Cena com Ladrilhos

Na maior parte dos jogos o cenário tem que ser montado a partir de uma folha de ladrilhos. As folhas de ladrilho são numeradas de 0 a n da esquerda para direita de cima para baixo. Esta é a folha de ladrilhos:







```
from _spy.circus.game import Circus

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""

    def preload(self):
        """Aqui no preload carregamos os recursos usados no jogo, neste caso a folha de ladrilhos"""
        self.spritesheet("ladrilho", "http://<advinha!>", 128, 128, 12)

    def create(self):
        """Aqui colocamos cada ladrilho indicando a posição na tela e depois selecionando o ladrilho"""
        um_ladrilho = self.sprite("ladrilho", 0, 0)
        um_ladrilho.frame = 5 # este número seleciona o ladrilho que vai ser colocado
```

```
um_ladrilho = self.sprite("ladrilho", 0, 0) # mude a posição do ladrilho
um_ladrilho.frame = 5 # troque o ladrilho!
# Coloque mais dois outros ladrilhos

if __name__ == "__main__":
    Jogo()
```

---

**Note:** Ainda é um programa bem simples.

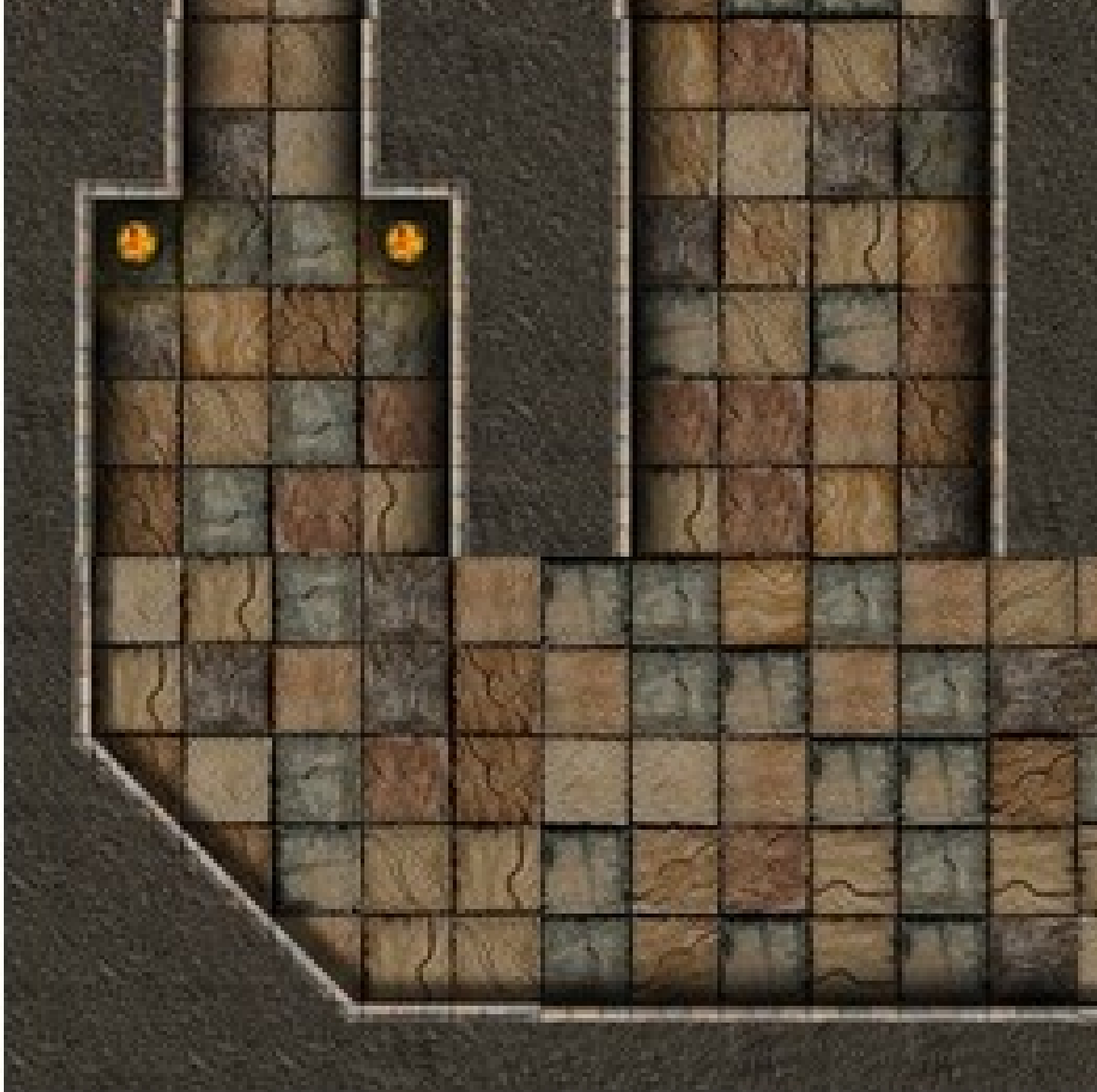
---

## 1.10 Cena com Ladrilhos Girados

Vamos montar uma outra masmorra, mas que agora requer que rotacionemos os ladrilhos. Veja os comando *angle* e *anchor* no código. A folha de ladrilhos continua esta:



Procure reproduzir este labirinto:



```
from _spy.circus.game import Circus

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus necessários para criar um jogo"""

    def preload(self):
        """Aqui no preload carregamos os recursos usados no jogo, neste caso a folha de ladrilhos"""
        self.spritesheet("ladrilho", "http://<advinha!>", 128, 128, 12)

    def create(self):
        """Aqui colocamos cada ladrilho indicando a posição na tela e depois selecionando o ladrilho"""
        um_ladrilho = self.sprite("ladrilho", 0, 0)
        um_ladrilho.frame = 5 # este número seleciona o ladrilho que vai ser colocado
        um_ladrilho.anchor.setTo(0.5, 0.5) # este comando faz com que a rotação seja no centro do l
```

```
um_ladrilho.angle = 90 # está é a rotação em graus do ladrilho
um_ladrilho = self.sprite("ladrilho", 0, 0) # mude a posição do ladrilho
um_ladrilho.frame = 5 # troque o ladrilho!
um_ladrilho.anchor.setTo(0.5, 0.5)
um_ladrilho.angle = 90 # troque o ângulo!
# Coloque aqui o resto dos ladrilhos

if __name__ == "__main__":
    Jogo()
```

---

**Note:** Observe que ao colocar a âncora no centro do ladrilho, este centro também servirá para posicionar o ladrilho. Ajuste as coordenadas para que tudo fique correto.

---

## 1.11 Cena com Mais Ladrilhos Girados

Vamos montar uma outra masmorra, um pouco maior. Em vez de copiar várias vezes o código, vamos usar uma *lista* de *tuplas* e o comando *for*. A folha de ladrilhos continua esta:



Procure reproduzir este labirinto:

```
from _spy.circus.game import Circus

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus necessários para criar um jogo"""
    MOSAICO = [
        [(5, 90), (5, 90), (5, 90), (5, 90)],
        [(5, 90), (5, 90), (5, 90), (5, 90)],
        [(5, 90), (5, 90), (5, 90), (5, 90)],
        [(5, 90), (5, 90), (5, 90), (5, 90)]
    ] # esta é uma lista de listas. Cada lista é uma linha e cada tupla dá o ladrilho (5) e o ângulo
```

```
def preload(self):
    """Aqui no preload carregamos os recursos usados no jogo, neste caso a folha de ladrilhos"""
    self.spritesheet("ladrilho", "http://<advinha!>", 128, 128, 12)

def create(self):
    """Aqui colocamos um for que varre cada linha e cada coluna da matriz"""
    azulejo = self.MOSAICO
    for y in range(4):
        for x in range(4):
            frame, angle = self.MOSAICO[y][x] # este comando desmembra a tupla em duas variáveis,
            um_ladrilho = self.sprite("ladrilho", 64+x*128, 64+y*128) # cria e muda a posição do
            um_ladrilho.frame = frame
            um_ladrilho.anchor.setTo(0.5, 0.5)
            um_ladrilho.angle = angle

if __name__ == "__main__":
    Jogo()
```

---

**Note:** Observe que ao colocar a âncora no centro do ladrilho, este centro também servirá para posicionar o ladrilho. Ajuste as cordenadas para que tudo fique correto.

---

## 1.12 Cena com Muitos Ladrilhos Girados

Vamos montar a masmorra original, bem maior. Em vez de copiar várias vezes o código, vamos usar um *dicionário* de *tuplas* e o comando *for*. A folha de ladrilhos continua esta:

Procure reproduzir este labirinto:

```
from _spy.circus.game import Circus

A, B, C, D, E, F, G = 0, 128, 256, 256+128, 512, 512+128, 512+256

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus necessários para criar um jogo"""
    MOSAICO = {
        (A, A): (5, 90), (B, A): (5, 90), (C, A): (5, 90), (D, A): (5, 90), (E, A): (5, 90), (F, A): (5, 90),
        (A, B): (5, 90), (B, B): (5, 90), (C, B): (5, 90), (D, B): (5, 90), (E, B): (5, 90), (F, B): (5, 90),
        (C, C): (5, 90), (B, C): (5, 90), (C, C): (5, 90), (D, C): (5, 90), (E, C): (5, 90), (F, C): (5, 90),
        (D, D): (5, 90), (B, D): (5, 90), (C, D): (5, 90), (D, D): (5, 90), (E, D): (5, 90), (F, D): (5, 90),
```



```

(E, E): (5, 90), (B, E): (5, 90), (C, E): (5, 90), (D, E): (5, 90), (E, E): (5, 90), (F, E): (5, 90),
} # este é um dicionário de tuplas. Cada chave é a posição do ladrilho e cada tupla dá o ladrilho

def preload(self):
    """Aqui no preload carregamos os recursos usados no jogo, neste caso a folha de ladrilhos"""
    self.spritesheet("ladrilho", "http://<advinha!>", 128, 128, 12)

def create(self):
    """Aqui colocamos cada ladrilho indicando a posição na tela e depois seleccionando o ladrilho
    for (x, y), (frame, angle) in self.MOSAICO.items():
        # coloque o resto aqui

if __name__ == "__main__":
    Jogo()

```

**Note:** Observe que ao colocar a âncora no centro do ladrilho, este centro também servirá para posicionar o ladrilho. Ajuste as cordenadas para que tudo fique correto.

## 1.13 Colocando Personagens

Vamos criar uma outra classe para um personagem monstro. O monstro vai ser criado na invocação `__init__` do Jogo. Use a folha de ladrilhos para programar os monstros:



```
from _spy.circus.game import Circus, Actor

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""
    def __init__(self):
        super().__init__() # super é invocado aqui para preservar os poderes recebidos do Circus
        self.ladrilho_monstro = "monstro"
        self.monstro = Monstro(self.ladrilho_monstro, 0, 0, 0)

    def preload(self):
        """Aqui no preload carregamos a imagem masmorra e a folha de ladrilhos dos monstros"""
        self.image("fundo", "http://<descubra um jeito de achar a url que vai ser posta aqui>")
        self.spritesheet(self.ladrilho_monstro, "http://<advinha!>", 64, 63, 16*12)

    def create(self):
        """Aqui colocamos a imagem masmorra na tela do jogo"""
        self.sprite("fundo")

class Monstro(Actor):
    """Essa é a classe Monstro que controla os personagens do jogo"""
    def __init__(self, nome, frame, x, y):
        super().__init__()
        self.nome, self.frame, self.x, self.y = nome, frame, x, y

    def create(self):
        """Aqui colocamos o sprite do monstro e selecionamos o frame que o representa"""
        self.monstro = "<troque isto pela criação do sprite, use self.nome, self.x e self.y>"
        "escolha aqui o frame e use self.frame"
        self.monstro.anchor.setTo(0.5, 0.5)

if __name__ == "__main__":
    Jogo()
```

## 1.14 Colocando Mais Personagens

Vamos criar outros monstros. A operação  $[f(x) \text{ for } x \text{ in } \langle \text{Coleção} \rangle]$  gera uma lista varrendo a Coleção dada e executando a função  $f(x)$  para cada  $x$  na Coleção.



```

from _spy.circus.game import Circus, Actor

MONSTROS = [(0, 0, 0), (0, 0, 0), (0, 0, 0)]

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""
    def __init__(self):
        super().__init__() # super é invocado aqui para preservar os poderes recebidos do Circus
        self.ladrilho_monstro = "monstro"
        self.monstro = [Monstro(self.ladrilho_monstro, frame, x, y) for frame, x, y in MONSTROS]

    def preload(self):
        """Aqui no preload carregamos a imagem masmorra e a folha de ladrilhos dos monstros"""
        self.image("fundo", "http://<descubra um jeito de achar a url que vai ser posta aqui>")
        self.spritesheet(self.ladrilho_monstro, "http://<advinha!>", 64, 63, 16*12)

    def create(self):
        """Aqui colocamos a imagem masmorra na tela do jogo"""
        self.sprite("fundo")

class Monstro(Actor):
    """Essa é a classe Monstro que controla os personagens do jogo"""
    def __init__(self, nome, frame, x, y):
        super().__init__()

```

```

self.nome, self.frame, self.x, self.y = nome, frame, x, y

def create(self):
    """Aqui colocamos o sprite do monstro e selecionamos o frame que o representa"""
    self.monstro = "<troque isto pela criação do sprite, use self.nome, self.x e self.y>"
    "escolha aqui o frame e use self.frame"
    self.monstro.anchor.setTo(0.5, 0.5)

if __name__ == "__main__":
    Jogo()

```

## 1.15 Animando Personagens

Vamos criar uma animação para o monstro. Use os ladrilhos onde o monstro apresenta as diversas posições:



```

from _spy.circus.game import Circus, Actor

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""
    def __init__(self):
        super().__init__() # super é invocado aqui para preservar os poderes recebidos do Circus
        self.ladrilho_monstro = "monstro"
        self.monstro = Monstro(self.ladrilho_monstro, 0, 0, 0)

```

```

def preload(self):
    """Aqui no preload carregamos a imagem masmorra e a folha de ladrilhos dos monstros"""
    self.image("fundo", "http://<descubra um jeito de achar a url que vai ser posta aqui>")
    self.spritesheet(self.ladrilho_monstro, "http://<advinha!>", 64, 63, 16*12)

def create(self):
    """Aqui colocamos a imagem masmorra na tela do jogo"""
    self.sprite("fundo")

class Monstro(Actor):
    """Essa é a classe Monstro que controla os personagens do jogo"""
    def __init__(self, nome, frame, x, y):
        super().__init__()
        self.nome, self.frame, self.x, self.y = nome, frame, x, y

    def create(self):
        """Aqui colocamos o sprite e o frame do monstro e acrescentamos a animação"""
        self.monstro = "<troque isto pela criação do sprite, use self.nome, self.x e self.y>"
        "escolha aqui o frame e use self.frame"
        self.monstro.anchor.setTo(0.5, 0.5)
        self.monstro.animations.add('mon', [self.frame, self.frame+1, self.frame+2,], 4, True)
        self.monstro.play('mon')

if __name__ == "__main__":
    Jogo()

```

## 1.16 Movimentando Personagens

Vamos movimentar monstros. Para isso é preciso acrescentar a função update na classe Monstro.





```

from _spy.circus.game import Circus, Actor
from random import random

MONSTROS = [(100, 100, 100), (0, 200, 100), (50, 300, 100)]
DIR = [(1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1)]

class Jogo(Circus):
    """Essa é a classe Jogo que recebe os poderes da classe Circus de poder criar um jogo"""
    def __init__(self):
        super().__init__() # super é invocado aqui para preservar os poderes recebidos do Circus
        self.ladrilho_monstro = "monstro"
        self.monstro = [Monstro(self.ladrilho_monstro, frame, x, y) for frame, x, y in MONSTROS]

    def preload(self):
        """Aqui no preload carregamos a imagem masmorra e a folha de ladrilhos dos monstros"""
        self.image("fundo", "http://<descubra um jeito de achar a url que vai ser posta aqui>")
        self.spritesheet(self.ladrilho_monstro, "http://<advinha!>", 64, 63, 16*12)

    def create(self):
        """Aqui colocamos a imagem masmorra na tela do jogo"""
        self.sprite("fundo")

class Monstro(Actor):
    """Essa é a classe Monstro que controla os personagens do jogo"""

```

```
def __init__(self, nome, frame, x, y):
    super().__init__()
    self.nome, self.frame, self.x, self.y = nome, frame, x, y
    self.first = True
    self.direction = 0

def create(self):
    """Aqui colocamos o sprite do monstro e selecionamos o frame que o representa"""
    self.monstro = "<troque isto pela criação do sprite, use self.nome, self.x e self.y>"
    "bote aqui self.??o que??"
    self.frame = self.frame
    self.monstro.anchor.setTo(0.5, 0.5)
    "ponha aqui a animação"
    self.enable(self.monstro)

def update(self):
    player = self.monstro
    def redirect():
        self.first = False
        self.direction = d = int(random() * 8.0)
        x, y = DIR[d]
        return x * 150, y * 150

    player.angle = (self.direction*45+270) % 360
    if int(random() + 0.02) or self.first:
        player.body.velocity.x, player.body.velocity.y = redirect()

if __name__ == "__main__":
    Jogo()
```





---

## Indices e Tabelas

---

- `genindex`
- `modindex`
- `search`