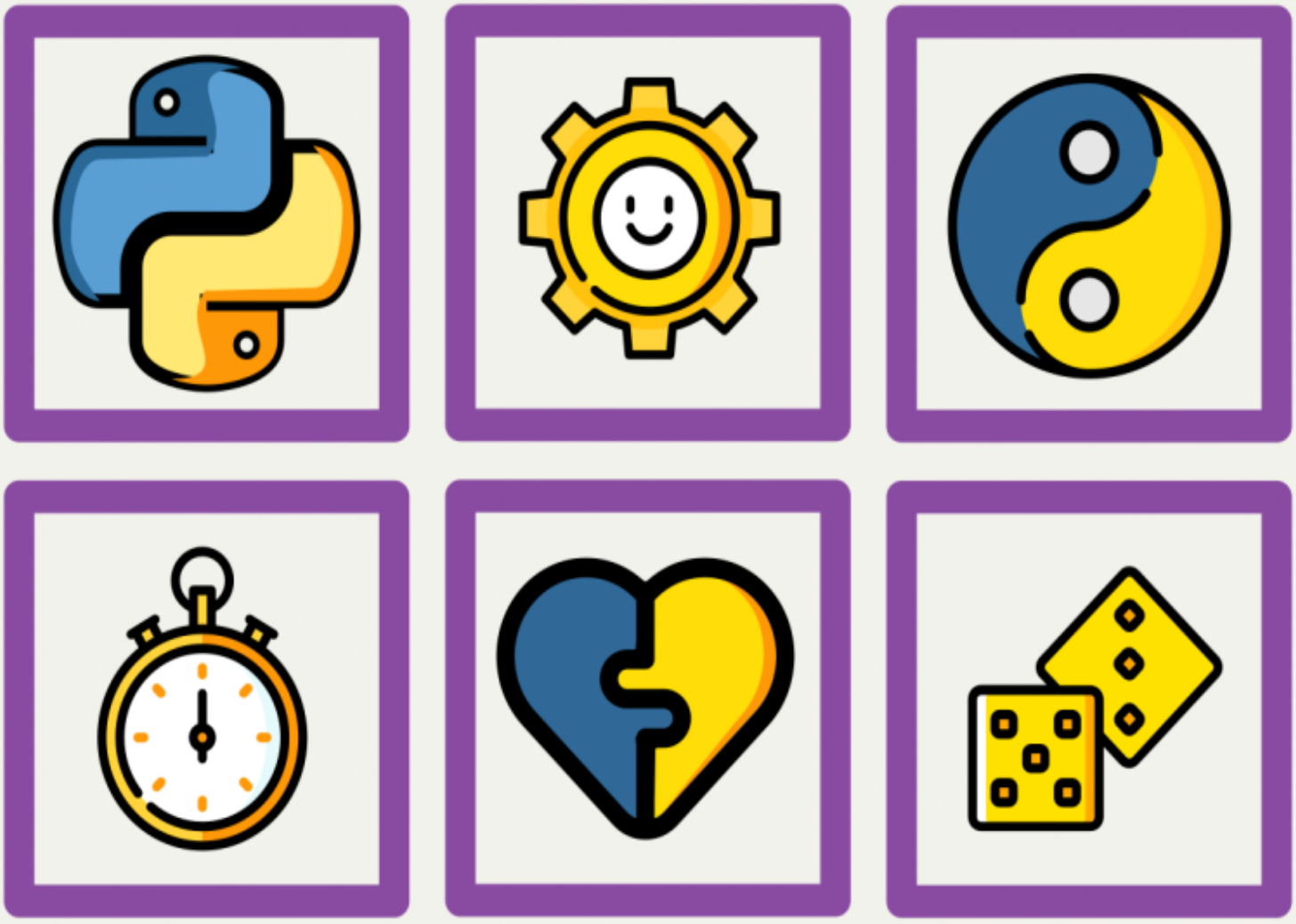
SIMPY: SIMULAÇÃO EM PYTHON

um guia prático



AFONSO C. MEDINA

ISBN: 978-65-01-19235-2

SimPy: Simulação em Python

Um guia prático

Afonso C. Medina

Este livro está disponível em https://leanpub.com/simpy

Esta versão foi publicada em 2025-07-18 ISBN 978-65-01-19235-2

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Medina, Afonso C.

SimPy [livro eletrônico] : simulação em Python :
um guia prático / Afonso C. Medina.–São Paulo :
Ed. do Autor, 2024.

PDF

ISBN 978-65-01-19235-2

1. Python (Linguagem de programação para computadores) 2. Simulação computacional I. Título.

24-233532 CDD-005.133

Índices para catálogo sistemático:

1. Python : Desenvolvimento de aplicações Web : Programação : Processamento de dados 005.133

Eliete Marques da Silva - Bibliotecária - CRB-8/9380

Também De Afonso C. Medina

Modelagem e Simulação de Eventos Discretos 5ª ed.

Conteúdo

1 See me, Fell me, README[1]

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

[1] https://www.youtube.com/watch?v=RC_MS-tG5vw

2 Apresentação

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

2.1 Por que utilizar o SimPy?

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

2.2 Prós e contras

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

2.3 Um breve histórico do SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

2.4 Onde procurar ajuda sobre o SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

2.5 Como utilizar este livro

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

3 Instalando o SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

3.1 Passo 1: Anaconda, the easy way

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

3.2 Passo 2: Instalando o Pip (para quem não instalou o Anaconda)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

3.3 Passo 3: Instalando o SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

3.4 Passo 4: Instalando algum Ambiente Integrado de Desenvolvimento (IDE)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

4 Tudo depende do Python

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

4.1 Teste seus conhecimentos em Python: o problema da ruína do apostador

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

4.2 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

4.3 Solução do desafio 1

> **Desafio 1**: dois apostadores iniciam um jogo de cara ou coroa em que cada um deles aposta $1 sempre em um mesmo lado da moeda. O vencedor leva a aposta total ($2). Cada jogador tem inicialmente $5 disponíveis para apostar. O jogo termina quando um dos jogadores atinge a ruína e não tem mais dinheiro para apostar.

O código a seguir é uma possível solução para o desafio 1 da seção anterior. Naturalmente é possível deixá-lo mais claro, eficiente, obscuro, maligno, elegante, rápido ou lento, como todo código de programação. O importante é que se você fez alguma que coisa que funcione, acredito que é o suficiente para começar com o SimPy.

```
1  import random                    # gerador de números aleatórios
2
3  names = ['Chewbacca', 'R2D2']    # jogadores
4
5  def transfer(winner, looser, bankroll, tossCount):
6      # função que transfere o dinheiro do winner para o looser
7      # imprime o vencedor do lançamento  e o bankroll de cada jogador
8      bankroll[winner] += 1
9      bankroll[looser] -= 1
10     print("\nLançamento: %d\tVencedor: %s" % (tossCount, names[winner]))
11     print("%s possui: $%d e %s possui: $%d"
12             % (names[0], bankroll[0], names[1], bankroll[1]))
13
14 def coinToss(bankroll, tossCount):
15     # função que sorteia a moeda e chama a transfer
16     if random.uniform(0, 1) < 0.5:
17         transfer(1, 0, bankroll, tossCount)
18     else:
19         transfer(0, 1, bankroll, tossCount)
20
21 def run2Ruin(bankroll):
22     # função que executa o jogo até a ruina de um dos jogadores
23     tossCount = 0                # contador de lançamentos
24     while bankroll[0] > 0 and bankroll[1] > 0:
25         tossCount += 1
26         coinToss(bankroll,tossCount)
27     winner = bankroll[1] > bankroll[0]
28     print("\n%s venceu depois de %d lançamentos, fim de jogo!"
29             % (names[winner], tossCount))
30
31 bankroll = [5, 5]               # dinheiro disponível para cada jogador
32 run2Ruin(bankroll)              # inicia o jogo
```

No meu computador, o problema anterior fornece o seguinte resultado:

```
Lançamento: 1   Vencedor: Chewbacca
Chewbacca possui: $6 e R2D2 possui: $4

Lançamento: 2   Vencedor: Chewbacca
Chewbacca possui: $7 e R2D2 possui: $3

Lançamento: 3   Vencedor: Chewbacca
Chewbacca possui: $8 e R2D2 possui: $2

Lançamento: 4   Vencedor: Chewbacca
Chewbacca possui: $9 e R2D2 possui: $1

Lançamento: 5   Vencedor: Chewbacca
Chewbacca possui: $10 e R2D2 possui: $0

Chewbacca venceu depois de 5 lançamentos, fim de jogo!
```

Note que o resultado fornecido deve ser diferente em seu computador, assim como ele se modifica a cada nova rodada no programa. Para que os resultados entre o meu computador e o seu sejam semelhantes, precisamos garantir que a sequência de números aleatórios nos dois computadores sejam as mesmas. Isso é possível com o comando `random.seed(semente)`, que estabelece um valor inicial fixo para a semente da sequência de números aleatórios gerada (veja o item 1 na seção "Teste seus conhecimentos", na seção a seguir).

4.4 Teste seus conhecimentos

5 Primeiros passos em SimPy: criando entidades

Algo elementar em qualquer pacote de simulação é uma função para criar entidades dentro do modelo. É o "Alô mundo!"[1] dos pacotes de simulação. Sua primeira missão, caso decida aceitá-la, será construir uma função que gere entidades com intervalos entre chegadas sucessivas exponencialmente distribuídos, com média de 2 min. Simule o sistema por 10 minutos apenas.

5.1 Chamada das bibliotecas random e simpy

Inicialmente, serão necessárias duas bibliotecas do Python: a `random` – biblioteca de geração de números aleatórios – e a `simpy`, que é o próprio SimPy.

Nosso primeiro modelo de simulação em SimPy começa com as chamadas das respectivas bibliotecas de interesse:

```
1  import random              # gerador de números aleatórios
2  import simpy               # biblioteca de simulação
3
4  random.seed(1000)          # semente do gerador de números aleatórios
```

Note a linha final `random.seed(1000)`. Ela garante que a geração de números aleatórios sempre começará pela mesma semente, de modo que a sequência de números aleatórios gerados a cada execução programa será sempre a mesma, facilitando o processo de verificação do programa.

5.2 Criando um environment de simulação

Tudo no SimPy gira em torno de **eventos** gerados por funções e todos os eventos devem ocorrer em um **environment**, ou um "ambiente" de simulação criado a partir da função `simpy.Environment()`.

Assim, nosso programa deve conter ao menos uma chamada à função `simpy.Environment()`, criando um *environment* "env":

[1] http://pt.wikipedia.org/wiki/Programa_Olá_Mundo

```
import random              # gerador de números aleatórios
import simpy               # biblioteca de simulação

random.seed(1000)          # semente do gerador de números aleatórios
env = simpy.Environment() # cria o environment do modelo na variável env
```

Se você executar o programa anterior, nada acontece. No momento, você apenas criou um *environment*, mas não criou nenhum processo, portanto, não existe ainda nenhum evento a ser simulado pelo SimPy.

5.3 Criando um gerador de chegadas dentro do environment

Vamos escrever uma função `geraChegadas()` que cria entidades no sistema enquanto durar a simulação. Nosso primeiro gerador de entidades terá três parâmetros de entrada: o *environment*, um atributo que representará o nome da entidade e a taxa desejada de chegadas de entidades por unidade de tempo. Para o SimPy, equivale dizer que você vai construir uma *função geradora de eventos* dentro do *environment* criado. No caso, os eventos gerados serão as chegadas de entidades no sistema.

Assim, nosso código começa a ganhar corpo:

```
import random              # gerador de números aleatórios
import simpy               # biblioteca de simulação

def geraChegadas(env, nome, taxa):
    # função que cria chegadas de entidades no sistema
    pass

random.seed(1000)          # semente do gerador de números aleatórios
env = simpy.Environment() # cria o environment do modelo
```

Precisamos informar ao SimPy que a função `geraChegadas()` é, de fato, um processo que deve ser executado ao longo de toda a simulação. Um processo é criado dentro do `environment`, pelo comando:

```
env.process(função_que_gera_o_processo)
```

A chamada ao processo é sempre feita após a criação do `env`, então basta acrescentar uma nova linha ao nosso código:

```
import random              # gerador de números aleatórios
import simpy               # biblioteca de simulação

def geraChegadas(env, nome, taxa):
# função que cria chegadas de entidades no sistema
    pass

random.seed(1000)          # semente do gerador de números aleatórios
env = simpy.Environment() # cria o environment do modelo
# cria o processo de chegadas
env.process(geraChegadas(env, "Cliente", 2)))
```

5.4 Criando intervalos de tempo de espera com env.timeout(tempo_de_espera)

Inicialmente, precisamos gerar intervalos de tempos aleatórios, exponencialmente distribuídos, para representar os tempos entre chegadas sucessivas das entidades. Para gerar chegadas com intervalos exponenciais, utilizaremos a biblioteca `random`, bem detalhada na sua documentação[2], e que possui a função:

```
random.expovariate(lambd)
```

Onde `lambd` é a taxa de ocorrência dos eventos ou, matematicamente, o inverso do tempo médio entre eventos sucessivos. No caso, se queremos que as chegadas ocorram entre intervalos médios de 2 min, a função ficaria:

```
random.expovariate(lambd=1.0/2.0)
```

A linha anterior é basicamente nosso gerador de números aleatórios exponencialmente distribuídos. O passo seguinte será informar ao SimPy que queremos nossas entidades surgindo no sistema segundo a distribuição definida. Isso é feito pela chamada da palavra reservada `yield` com a função do SimPy `env.timeout(intervalo)`, que nada mais é do que uma função que causa um atraso de tempo, um *delay* no tempo dentro do *enviroment* `env` criado:

```
yield env.timeout(random.expovariate(1.0/2.0))
```

Na linha de código anterior estamos executando `yield env.timeout(0.5)` para que o modelo retarde o processo num tempo aleatório gerado pela função `random.expovariate(0.5)`.

Oportunamente, discutiremos mais a fundo qual o papel do palavra `yield` (*spoiler*: ela não é do SimPy, mas originalmente do próprio Python). Por hora, considere que ela é apenas uma maneira de **criar eventos** dentro do `env` e que, caso uma função represente um processo, obrigatoriamente

[2] https://docs.python.org/2/library/random.html

ela precisará conter o comando `yield *alguma coisa*`, bem como o respectivo `environment` do processo.

Uma função criada no Python (com o comando `def`) só é tratada como um **processo** ou **gerador de eventos** para o SimPy, caso ela contenha ao menos uma linha de código com o comando `yield`. Mais adiante, a seção "O que são funções geradoras", explica em mais detalhe o funcionamento do `yield`.

Colocando tudo junto na função `geraChegadas()`, temos:

```
1  import random              # gerador de números aleatórios
2  import simpy               # biblioteca de simulação
3
4  def geraChegadas(env, nome, taxa):
5      # função que cria chegadas de entidades no sistema
6      contaChegada = 0
7      while True:
8          yield env.timeout(random.expovariate(1.0/taxa))
9          contaChegada += 1
10         print("%s %i chega em: %.1f " % (nome, contaChegada, env.now))
11
12 random.seed(1000)          # semente do gerador de números aleatórios
13 env = simpy.Environment() # cria o environment do modelo
14
15 # cria o processo de chegadas
16 env.process(geraChegadas(env, "Cliente", 2)))
```

O código deve ser autoexplicativo: o laço `while` é **infinito** enquanto dure a simulação; um contador, `contaChegada`, armazena o total de entidades geradas e a função `print`, imprime na tela o instante de chegada de cada cliente. Note que, dentro do `print`, existe uma chamada para a **hora atual de simulação** `env.now`.
Por fim, uma chamada a função `random.seed()` garante que os números aleatórios a cada execução do programa serão os mesmos.

5.5 Executando o modelo por um tempo determinado com env.run(until=tempo_de_simulacao)

Se você executar o código anterior, nada acontece novamente, pois ainda falta informarmos ao SimPy qual o tempo de duração da simulação. Isto é feito pelo comando:

```
1  env.run(until=tempo_de_simulação)
```

No exemplo proposto, o tempo de simulação deve ser de 10 min, como representado na linha 15 do código a seguir:

```
import random              # gerador de números aleatórios
import simpy               # biblioteca de simulação

def geraChegadas(env, nome, taxa):
    # função que cria chegadas de entidades no sistema
    contaChegada = 0
    while True:
        yield env.timeout(random.expovariate(1/taxa))
        contaChegada += 1
        print("%s %i chega em: %.1f " % (nome, contaChegada, env.now))

random.seed(1000)          # semente do gerador de números aleatórios
env = simpy.Environment() # cria o environment do modelo

# cria o processo de chegadas
env.process(geraChegadas(env, "Cliente", 2))

# roda a simulação por 10 unidades de tempo
env.run(until=10)
```

Ao executar o programa, temos a saída:

```
Cliente 1 chega em: 3.0
Cliente 2 chega em: 5.2
Cliente 3 chega em: 5.4
Cliente 4 chega em: 6.3
Cliente 5 chega em: 7.6
Cliente 6 chega em: 9.1
```

Agora sim!

Note que `env.process(geraChegadas(env))` é um comando que **torna** a função `geraChegadas()` um **processo** ou um **gerador de eventos** dentro do `Environment env`. Esse processo só começa a ser executado na linha seguinte, quando `env.run(until=10)` informa ao SimPy que todo processo pertencente ao `env` deve ser executado por um **tempo de simulação** igual a 10 minutos.

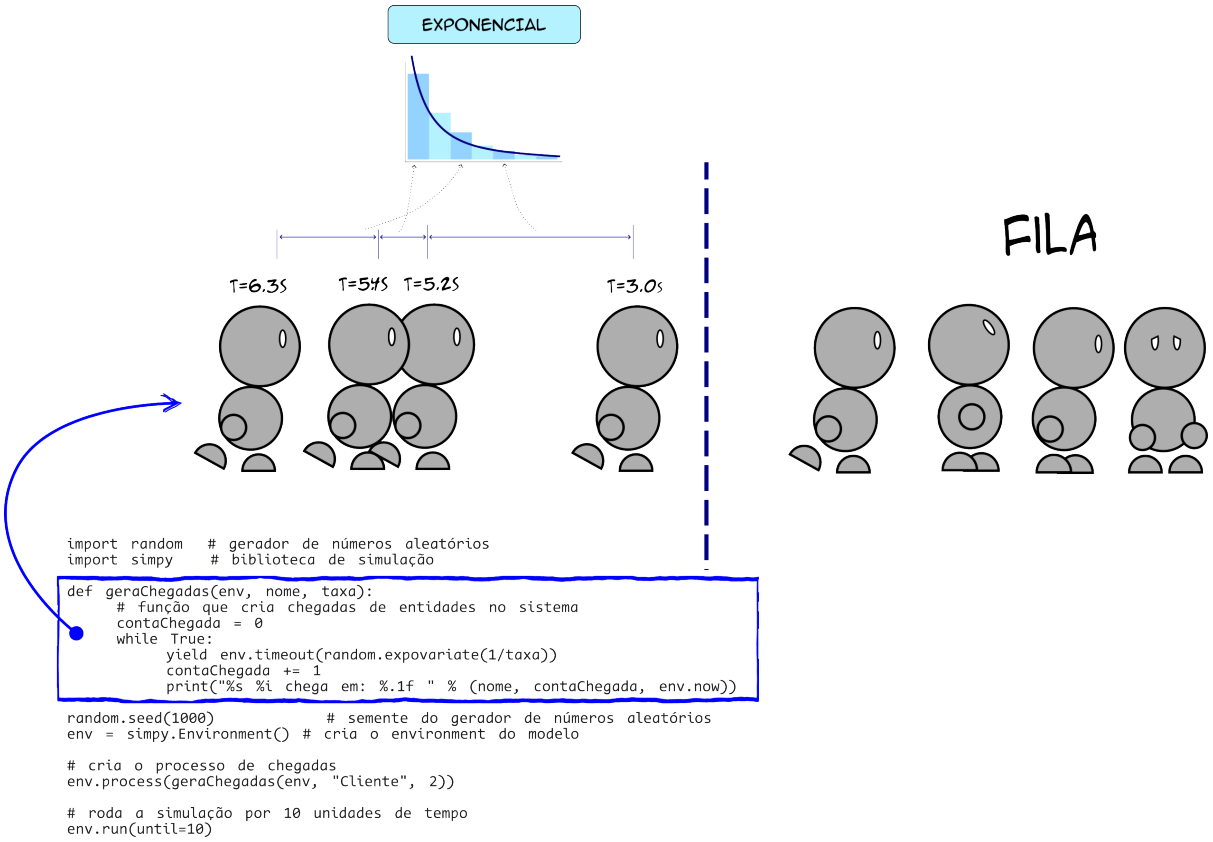


Figura 5.1. Representação das chegadas na fila.

5.6 Conceitos desta seção

Conteúdo	Descrição
`env = simpy.Environment()`	cria um `Environment` de simulação
`random.expovariate(lambd)`	gera números aleatórios exponencialmente distribuídos, com taxa de ocorrência (eventos/unidade de tempo) igual a `lambd`
`yield env.timeout(time)`	gera um atraso dado por `time`
`random.seed(seed)`	define o gerador de sementes aleatórias para um mesmo valor a cada nova simulação
`env.process(geraChegadas(env))`	inicia a função `geraChegadas` como um *processo* em `env`
`env.run(until=tempoSim)`	executa a simulação (executa todos os processos criados em `env`) pelo tempo `tempoSim`
`env.now`	retorna o instante atual da simulação

5.7 Desafios

Desafio 2: é comum que os comandos de criação de entidades nos softwares proprietários[3] tenham a opção de limitar o número máximo de entidades geradas durante a simulação.
Modifique a função `geraChegadas` de modo que ela receba como parâmetro `numeroMaxChegadas` e limite a criação de entidades a este número.

Desafio 3: modifique a função `geraChegadas` de modo que as chegadas entre entidades sejam distribuídas segundo uma distribuição triangular de moda 1, menor valor 0,1 e maior valor 1,1.

[3] https://pt.wikipedia.org/wiki/Software_proprietário

5.8 Solução dos desafios 2 e 3

> **Desafio 2:** é comum que os comandos de criação de entidades nos softwares proprietários tenham a opção de limitar o número máximo de entidades geradas durante a simulação.
> Modifique a função `geraChegadas` de modo que ela receba como parâmetro o `numeroMaxChegadas` e limite a criação de entidades a este número.

Neste caso, o *script* em Python é autoexplicativo, apenas note que limitei o número de chegadas em 5 e fiz isso antes da chamada do processo gerado pela função `geraChegadas()`:

```
import random      # gerador de números aleatórios
import simpy       # biblioteca de simulação

def geraChegadas(env, nome, taxa, numeroMaxChegadas):
    # função que cria chegadas de entidades no sistema
    contaChegada = 0
    while (contaChegada < numeroMaxChegadas):
        yield env.timeout(random.expovariate(1/taxa))
        contaChegada += 1
        print("%s %i chega em: %.1f " % (nome, contaChegada, env.now))

random.seed(1000)   # semente do gerador de números aleatórios
env = simpy.Environment() # cria o environment do modelo
# cria o processo de chegadas
env.process(geraChegadas(env, "Cliente", 2, 5))
env.run(until=10) # executa a simulação por 10 unidades de tempo
```

> **Desafio 3:** modifique a função `geraChegadas` de modo que as chegadas entre entidades sejam distribuídas segundo uma distribuição triangular de moda 1, menor valor 0,1 e maior valor 1,1.

Neste caso, precisamos verificar na documentação da biblioteca `random`, quais são nossas opções. A tabela a seguir, resume as distribuições disponíveis:

Função	Distribuição
`random.random()`	gera números aleatórios no intervalo [0.0, 1.0)
`random.uniform(a, b)`	uniforme no intervalo [a, b]
`random.triangular(low, high, mode)`	triangular com menor valor *low*, maior valor *high* e moda *mode*
`random.betavariate(alpha, beta)`	beta com parâmetros *alpha* e *beta*
`random.expovariate(lambd)`	exponencial com média 1/*lambd*
`random.gammavariate(alpha, beta)`	gamma com parâmetros *alpha* e *beta*
`random.gauss(mu, sigma)`	normal com média *mu* e desvio padrão *sigma*
`random.lognormvariate(mu, sigma)`	lognormal com média *mu* e desvio padrão *sigma*
`random.normalvariate(mu, sigma)`	equivalente à random.gauss, mas um pouco mais lenta
`random.vonmisesvariate(mu, kappa)`	distribuição de von Mises[4] com parâmetros *mu* e *kappa*
`random.paretovariate(alpha)`	pareto com parâmetro *alpha*
`random.weibullvariate(alpha, beta)`	weibull com parâmetros *alpha* e *beta*

A biblioteca NumPy, que veremos oportunamente, possui mais opções para distribuições estatísticas. Por enquanto, o desafio 3 pode ser solucionado de maneira literal:

```
1  import random      # gerador de números aleatórios
2  import simpy       # biblioteca de simulação
3
4  def geraChegadas(env, nome, numeroMaxChegadas):
5      # função que cria chegadas de entidades no sistema
6      contaChegada = 0
7      while (contaChegada < numeroMaxChegadas):
8          yield env.timeout(random.triangular(0.1,1,1.1))
9          contaChegada += 1
10         print("%s %i chega em: %.1f " % (nome, contaChegada, env.now))
11
12 random.seed(1000)          # semente do gerador de números aleatórios
13 env = simpy.Environment() # cria o environment do modelo
14 # cria o processo de chegadas
15 env.process(geraChegadas(env, "Cliente", 5))
16 env.run(until=10)
```

Dica: os modelos de simulação, com muitos processos de chegadas e atendimento, tendem a utilizar diversas funções diferentes de distribuição de probabilidades, deixando as coisas meio confusas para o programador. Uma dica bacana é criar uma função que armazene todas as distribuições do modelo em um único lugar, como uma prateleira de distribuições.

Por exemplo, imagine um modelo em SimPy que possui 3 processos: um exponencial com média 10 min, um triangular com parâmetros (10, 20, 30) min e um normal com média 0 e desvio 1 minuto. A função `distribution()` a seguir, armazena todos os geradores de números aleatórios em um único local:

```
import random

def distributions(tipo):
    return {
        'arrival': random.expovariate(1/10.0),
        'singing': random.triangular(10, 20, 30),
        'applause': random.gauss(10, 1),
    }.get(tipo, 0.0)
```

O próximo exemplo testa como chamar a função:

```
import random

def distributions(tipo):
    return {
        'arrival': random.expovariate(1/10.0),
        'singing': random.triangular(10, 20, 30),
        'applause': random.gauss(10, 1),
    }.get(tipo, 0.0)

tipo = 'arrival'
print(tipo, distributions(tipo))

tipo = 'singing'
print(tipo, distributions(tipo))

tipo = 'applause'
print(tipo, distributions(tipo))
```

O qual produz a saída:

```
arrival 6.231712146858156
singing 22.192356552471104
applause 10.411795571842426
```

Essa foi a nossa dica do dia!

Fique a vontade para implementar funções de geração de números aleatórios ao seu gosto. Note, e isso é importante, que **praticamente todos os seus modelos de simulação em SimPy precisarão deste tipo de função!**

5.9 Teste seus conhecimentos

1. Acrescente ao programa inicial, uma função `distribution` como a proposta na seção "Dica" e faça o tempo entre chegadas sucessivas de entidades chamar a função para obter o valor correto.
2. Considere que 50% das entidades geradas durante a simulação são do sexo feminino e 50% do sexo masculino. Modifique o programa para que ele sorteie o gênero dos clientes. Faça esse sorteio dentro da função `distribution` já criada.

6 Criando, ocupando e desocupando recursos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

6.1 Criando

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

6.2 Ocupando

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

6.3 Desocupando

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

6.4 Status do recurso

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

6.5 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7 Juntando tudo em um exemplo: a fila M/M/1

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.1 Geração de chegadas de entidades

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.2 Realizando o atendimento no servidor

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.3 Uma representação alternativa para a ocupação e desocupação de recursos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.4 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.5 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.6 Solução dos desafios 4, 5 e 6

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

7.7 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8 Atributos e variáveis: diferenças em SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8.1 Atributos em modelos orientados ao objeto

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8.2 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8.3 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8.4 Solução dos desafios 7 e 8

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

8.5 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9 Environments: controlando a simulação

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.1 Controle de execução com env.run(until=fim_da_simulação)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.2 Parada por execução de todos os processos programados

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.3 Parada por fim de execução de processo específico por env.run(until=processo)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.4 Simulação passo a passo: peek & step

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.5 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.6 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

9.7 Solução dos desafios 9 e 10

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10 Outros tipos de recursos: com prioridade e preemptivos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10.1 Recursos com prioridade: PriorityResource

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10.2 Recursos que podem ser interrompidos: PreemptiveResource

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10.3 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10.4 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

10.5 Solução dos desafios 11 e 12

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11 Interrupções de processos: simpy.Interrupt

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.1 Criando quebras de equipamento

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.2 Interrompendo um processo sem captura por try...except

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.3 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.4 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.5 Solução dos desafios 13 e 14

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

11.6 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

12 Armazenagem e seleção de objetos específicos com Store, FilterStore e PriorityStore

12.1 Construindo um conjunto de objetos com Store

12.2 Selecionando um objeto específico com FilterStore()

12.3 Criando um Store com prioridade: PriorityStore

12.4 Conceitos desta seção

12.5 Desafios

12.6 Solução dos desafios 15 e 16

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

12.7 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13 Enchendo ou esvaziando caixas, tanques, estoques ou objetos com Container()

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.1 Enchendo o meu container yield meuContainer.put(quantidade)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.2 Esvaziando o meu container: yield meuContainer.get(quantidade)

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.3 Criando um sensor para o nível atual do container

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.4 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.5 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.6 Solução dos desafios 17 e 18

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

13.7 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14 Criando lotes (ou agrupando) entidades durante a simulação

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14.1 Uma tática para agrupamento de lotes utilizando o Container

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14.2 Agrupando lotes por atributo da entidade utilizando o FilterStore

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14.3 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14.4 Solução dos desafios 19 e 20

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

14.5 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15 Criando, manipulando e disparando eventos com event()

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15.1 Criando um evento isolado com event()

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15.2 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15.3 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15.4 Solução dos desafios 21 e 22

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

15.5 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16 Aguardando múltiplos eventos ao mesmo tempo com AnyOf e AllOf

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.1 Aguardando até que, ao menos, um evento termine com AnyOf

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.2 Aguardando todos os eventos com AllOf

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.3 Comprendendo melhor as saídas dos comandos AllOf e AnyOf

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.4 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.5 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

16.6 Solução dos desafios 23 e 24

16.7 Teste seus conhecimentos

17 Propriedades úteis dos eventos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

17.1 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

17.2 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

17.3 Solução do desafio 25

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

17.4 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18 Adicionando callbacks aos eventos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18.1 Todo processo é um evento

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18.2 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18.3 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18.4 Solução do desafio 26

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

18.5 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

19 Interrupções de eventos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

19.1 Interrompendo um evento com o método interrupt

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

19.1.1 Método de controle de interrupção 1: lógica de exceção try...except

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

19.1.2 Método de controle de interrupção 2: alterando o atributo defused

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

19.2 Interrompendo um evento com o método fail

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

20 O que são funções geradoras? (ou como funciona o SimPy) - Parte I

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

20.1 Iterador

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

20.2 Funções geradoras

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

21 O que são funções geradoras? (ou como funciona o SimPy?) - Parte II

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

21.1 SimPy vs. funções geradoras

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22 Simulação Baseada em Agentes usando o SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.1 Como construir um modelo de simulação de agentes: passos básicos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.2 Modelo epidêmico baseado em agentes: o modelo SIR[1]

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.3 Modelagem do problema no SimPy

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.3.1 Importação de bibliotecas

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.3.2 Parâmetros de entrada e variáveis globais

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

[1]Baseado em: Borshchev, A. e Grigoryev, Illya. The Big Book of Simulation Modeling. Multimethod modeling with AnyLogic 8. AnyLogic North America. 2020.

22.3.3 Construtor de agentes

22.3.4 Inicialização

22.3.5 Envio e recebimento de mensagens

22.3.6 Processo de infecção

22.3.7 Processo de incubação

22.3.8 Monitoramento e exibição de gráficos

22.3.9 Execução (ufa!)

22.4 Melhorando o desempenho do código de simulação

22.4.1 Criando uma malha de conexões entre vizinhos mais próximos

22.5 Como seguir a partir daqui

22.6 Conceitos desta seção

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.7 Desafios

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.8 Solução dos Desafios 27 e 28

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.

22.9 Teste seus conhecimentos

Este conteúdo não está disponível na amostra do livro. O livro pode ser adquirido no Leanpub em https://leanpub.com/simpy.