VOLTAR

Multi-threading

Apresentar o recurso de multi-threading do Python para o processamento em paralelo.

NESTE TÓPICO

Marcar tópico

Olá alunos,

Os processos simultâneos foram um dos recursos principais para levar o homem à Lua, na época, eles criaram um sistema operacional chamado EXEC que poderia executar oito processos simultaneamente, isto fez com que os sistemas operacionais que surgiram após, como UNIX, fossem multitarefas. O hardware, através do processador, desenvolveu a técnica de pipeline (em paralelo) para a execução das linhas de códigos.

Hoje temos processadores com vários núcleos, que podem realizar vários processos em paralelo ou independentemente. Em algumas linguagens, como CUDA, já é possível a programação em paralelo quando se trata de processamentos mais complexos e pesados para o computador, como processar elementos gráficos.

Além de podermos utilizar processos múltiplos, também podemos trabalhar com threads múltiplos ou multi-threads, threads são execuções de linhas de código de forma encadeada. No caso do thread, a divisão na hora da execução fica a cargo do sistema operacional, o SO é quem vai encadear a execução dos threads.

Em Python é possível criar programas com threads e distribuir processos de forma independente. Para isto, é melhor trabalharmos com as técnicas de orientação a objetos, criando classes e funções.

Lembrando que podemos trabalhar com funções utilizando a clausula **DEF** e nomeando-a, quando criamos scripts e agora vamos criar uma classe e incluiremos funções.

Vamos criar um **script** com o nome **thread.py:**

```
1.  import threading
2.
3.  # Instanciando a classe principal
4.  thread1 = principal(1, "THREAD 1", 20)
5.  thread2 = principal(2, "THREAD 2", 15)
```

Primeiro importamos o módulo **threading** e instanciamos a classe principal, com duas varáveis **thread 1** e **thread 2**, passando um ID, um nome e uma quantidade de processos.

```
1.  import threading
2.
3.  # Instanciando a classe principal
4.  thread1 = principal(1, "THREAD 1", 20)
5.  thread2 = principal(2, "THREAD 2", 15)
6.
7.  # iniciando as threads
8.  thread1.start()
9.  thread2.start()
```

Conforme a **linha 8 e 9**, iniciamos as threads com a função **start**.

```
1.  # classe principal com duas funções
2.  class principal (threading.Thread):
3.      def __init__(self, threadID, nome, cont):
4.          threading.Thread.__init__(self)
5.          self.threadID = threadID
6.          self.nome = nome
7.          self.cont = cont
8.
9.      def run(self):
10.         print (" Iniciando a %s com %d processos" % (self.nome,self.cont))
11.         processo(self.nome, self.cont)
```

Ao importarmos o módulo **threading**, criamos a classe **principal** e passamos como argumento **(threading.Thread)**, utilizando a função **Thread.**

Definimos o método (função) **__init__** , função padrão inicial do Python com os argumentos **(self, threadID, nome, cont)**. A **self** é uma função base para referenciar qualquer objeto. A **threadID** recebe o ID, o **nome** e o valor do **cont**.

Na **linha 4**, a função inicial recebe a função base **self**.

Nas **linhas 5, 6 e 7**, a função base **self** recebe os valores das variáveis: **threadID, nome e cont.**

Na **linha 9,** criamos outra função utilizando a função **run ( )** para iniciar a execução das threads, passando como argumento a função **self**.

Na **linha 11**, chamamos a função **processo**, passando os valores **nome e cont.**

```
1.  import threading
2.
3.  # classe principal com duas funções
4.  class principal (threading.Thread):
5.      def __init__(self, threadID, nome, cont):
6.          threading.Thread.__init__(self)
7.          self.threadID = threadID
8.          self.nome = nome
9.          self.cont = cont
10.
11.     def run(self):
12.         print (" Iniciando a %s com %d processos" % (self.nome,self.cont))
13.         processo(self.nome, self.cont)
14.
15. def processo(nome, cont):
16.     while cont:
17.         print (" A %s realiza o processo %d" % (nome, cont))
18.         cont = cont - 1
19.
20. # Instanciando a classe principal
21. thread1 = principal(1, "THREAD 1", 20)
22. thread2 = principal(2, "THREAD 2", 15)
23.
24. # iniciando as threads
25. thread1.start()
26. thread2.start()
```

Na **linha 15**, criamos a função **processo**, recebendo os valores do **nome** e **cont**. Note que a função **processo** não pertence à classe **principal**. Com um **while** sob a condição do valor **cont**, vai imprimindo a mensagem dos threads sendo executados até finalizar com o decremento: cont = cont – 1.

```
1.  import threading
2.
3.  # classe principal com duas funções
4.  class principal (threading.Thread):
5.      def __init__(self, threadID, nome, cont):
6.          threading.Thread.__init__(self)
7.          self.threadID = threadID
8.          self.nome = nome
9.          self.cont = cont
10.
11.     def run(self):
12.         print (" Iniciando a %s com %d processos" % (self.nome,self.cont))
13.         processo(self.nome, self.cont)
14.
15. def processo(nome, cont):
16.     while cont:
17.         print (" A %s realiza o processo %d" % (nome, cont))
18.         cont = cont - 1
19.
20. # Instanciando a classe principal
21. thread1 = principal(1, "THREAD 1", 20)
22. thread2 = principal(2, "THREAD 2", 15)
23.
24. # iniciando as threads
25. thread1.start()
26. thread2.start()
27.
28. pipes = []
29. pipes.append(thread1)
30. pipes.append(thread2)
31.
32. for i in pipes:
33.     i.join()
34.
35. input('\nTecle ENTER para sair...')
```

Complementando o código, criamos a variável **pipes** com uma lista de **threads** e a função **append** vai encadeando as threads **(linhas 28, 29 e 30)**.

Na **linha 32**, um loop com **for** é executado pela quantidade de threads sendo executados e a função **join( )** impede que ocorra o impasse (deadlock) no encadeamento dos pipes.

Agora, vamos executar o scritp acima por duas vezes e verificar os resultados abaixo:

Resultado da primeira execução:

```
  Iniciando a THREAD 1 com 20 processos
  A THREAD 1 realiza o processo 20
  Iniciando a THREAD 2 com 15 processos
  A THREAD 1 realiza o processo 19
  A THREAD 2 realiza o processo 15
  A THREAD 1 realiza o processo 18
  A THREAD 2 realiza o processo 14
  A THREAD 1 realiza o processo 17
  A THREAD 2 realiza o processo 13
  A THREAD 1 realiza o processo 16
  A THREAD 2 realiza o processo 12
  A THREAD 1 realiza o processo 15
  A THREAD 2 realiza o processo 11
  A THREAD 1 realiza o processo 14
  A THREAD 2 realiza o processo 10
  A THREAD 1 realiza o processo 13
  A THREAD 2 realiza o processo 9
  A THREAD 1 realiza o processo 12
  A THREAD 2 realiza o processo 8
  A THREAD 1 realiza o processo 11
  A THREAD 2 realiza o processo 7
  A THREAD 1 realiza o processo 10
  A THREAD 2 realiza o processo 6
  A THREAD 1 realiza o processo 9
  A THREAD 2 realiza o processo 5
  A THREAD 1 realiza o processo 8
  A THREAD 2 realiza o processo 4
  A THREAD 1 realiza o processo 7
  A THREAD 2 realiza o processo 3
  A THREAD 1 realiza o processo 6
  A THREAD 2 realiza o processo 2
  A THREAD 1 realiza o processo 5
  A THREAD 2 realiza o processo 1
  A THREAD 1 realiza o processo 4
  A THREAD 1 realiza o processo 3
  A THREAD 1 realiza o processo 2
  A THREAD 1 realiza o processo 1

 Tecle ENTER para sair...
```

Resultado da segunda execução:

```
Iniciando a THREAD 1 com 20 processos
A THREAD 1 realiza o processo 20
A THREAD 1 realiza o processo 19
A THREAD 1 realiza o processo 18
A THREAD 1 realiza o processo 17
A THREAD 1 realiza o processo 16
A THREAD 1 realiza o processo 15
A THREAD 1 realiza o processo 14
A THREAD 1 realiza o processo 13
A THREAD 1 realiza o processo 12
A THREAD 1 realiza o processo 11
A THREAD 1 realiza o processo 10
A THREAD 1 realiza o processo 9
A THREAD 1 realiza o processo 8
A THREAD 1 realiza o processo 7
A THREAD 1 realiza o processo 6
Iniciando a THREAD 2 com 15 processos
A THREAD 1 realiza o processo 5
A THREAD 2 realiza o processo 15
A THREAD 1 realiza o processo 4
A THREAD 2 realiza o processo 14
A THREAD 1 realiza o processo 3
A THREAD 2 realiza o processo 13
A THREAD 1 realiza o processo 2
A THREAD 2 realiza o processo 12
A THREAD 1 realiza o processo 1
A THREAD 2 realiza o processo 11
A THREAD 2 realiza o processo 10
A THREAD 2 realiza o processo 9
A THREAD 2 realiza o processo 8
A THREAD 2 realiza o processo 7
A THREAD 2 realiza o processo 6
A THREAD 2 realiza o processo 5
A THREAD 2 realiza o processo 4
A THREAD 2 realiza o processo 3
A THREAD 2 realiza o processo 2
A THREAD 2 realiza o processo 1

Tecle ENTER para sair...
```

O script foi executado duas vezes e notem que os processos foram iniciados de maneira diferentes: na primeira execução, foi iniciado o **THREAD 1** e executado o processo **20** e depois iniciou-se o **THREAD 2.**

Na segunda execução, foi iniciado o **THREAD 1** e executado os processos do 6 ao 20 e depois é que foi iniciado o **THREAD 2**. Isto aconteceu porque nós não temos a autonomia para determinar a ordem do encadeamento da execução das threads, isto fica a cargo do sistema operacional que faz o escalonamento das threads a serem executadas.

SAIBA MAIS...

Dê uma olhada nos links abaixo para saber mais sobre a linguagem Python:

https://www.python.org/doc/ (https://www.python.org/doc/)

https://wiki.python.org/moin/PythonBooks (https://wiki.python.org/moin/PythonBooks)

Neste tópico vimos o conceito de multi-threading e como implementa-la na linguagem Python através de uma aplicação..

Quiz

Exercício Final

Multi-threading

INICIAR

Referências

SUMMERFIELD, M. *Programação em Python 3*: Uma introdução completa à linguagem Python. Rio de Janeiro Alta Books, 2012. 495 p.

MENEZES, N. N. C. *Introdução à programação com Python:* algoritmos e lógica de programação para iniciantes. 2. ed. São Paulo: Novatec, 2014. 328 p.

SWEIGART, AL. *Automatize tarefas maçantes com Python:* programação prática para verdadeiros iniciantes. São Paulo: Novatec, 2015. 568 p.

PYTHON, doc. Disponível em: <https://www.python.org/doc/>. Acesso em: Junho/2018.

PYTHON, books. Disponível em: <https://wiki.python.org/moin/PythonBooks>. Acesso em: Junho/2018.

Avalie este tópico

ANTERIOR
Info Gather

Índice

Programação em GUI com Tkinter

PRÓXIMO