Python - registro em log

O termo “logging” refere-se ao mecanismo de registro de diferentes eventos intermediários em um determinado processo. A gravação de logs em um aplicativo de software é útil para o desenvolvedor na depuração e rastreamento de quaisquer erros na lógica do aplicativo. A biblioteca padrão do Python inclui o módulo de registro com o qual os logs do aplicativo podem ser gerados e registrados.

É uma prática normal usar instruções print() intermitentemente em um programa para verificar valores intermediários de diferentes variáveis e objetos. Ajuda o desenvolvedor a verificar se o programa está se comportando conforme o esperado ou não. No entanto, o registro em log é mais benéfico do que as instruções de impressão intermitentes, pois fornece mais informações sobre os eventos.

Níveis de registro

Um dos recursos importantes do log é que você pode gerar mensagens de log com diferentes níveis de gravidade. O módulo de registro define os seguintes níveis com seus valores.

Nível	Quando é usado	Valor
DEPURAR	Informações detalhadas, normalmente de interesse apenas no diagnóstico de problemas.	10
INFORMAÇÕES	Confirmação de que as coisas estão funcionando conforme o esperado.	20
AVISO	Uma indicação de que algo inesperado aconteceu ou de algum problema no futuro próximo (por exemplo, “pouco espaço em disco”). O software ainda está funcionando conforme o esperado.	30
ERRO	Devido a um problema mais sério, o software não conseguiu realizar alguma função.	40
CRÍTICO	Um erro grave, indicando que o próprio programa pode não conseguir continuar em execução.	50

Exemplo

O código a seguir ilustra como gerar mensagens de log.

```
import logging
```

```
logging.debug('This is a debug message')
logging.info('This is an info message')
logging.warning('This is a warning message')
logging.error('This is an error message')
logging.critical('This is a critical message')
```

Ele produzirá a seguinte **saída** -

```
WARNING:root:This is a warning message
ERROR:root:This is an error message
CRITICAL:root:This is a critical message
```

Observe que apenas as mensagens de log após o nível WARNING são exibidas aqui. Isso ocorre porque root - o criador de logs padrão ignora todos os níveis de gravidade acima do nível de gravidade WARNING. Observe que o nível de gravidade é registrado antes dos primeiros dois pontos (:) de cada linha. Da mesma forma, root é o nome do logger e também é exibido o LogRecord.

Configuração de registro em log

Os logs gerados pelo programa podem ser customizados com o método BasicConfig(). Você pode definir um ou mais dos seguintes parâmetros para configuração -

- **filename** - Especifica que um FileHandler seja criado, usando o nome de arquivo especificado, em vez de um StreamHandler.
- **filemode** - Se o nome do arquivo for especificado, abra o arquivo neste modo. O padrão é 'a'.
- **datefmt** - Use o formato de data/hora especificado, conforme aceito por time.strftime().
- **style** - Se o formato for especificado, use este estilo para a string de formato. Um de '%', '{' ou '$' para estilo printf, str.format() ou string.Template respectivamente. O padrão é '%'.
- **level** - Defina o nível do criador de logs raiz para o nível especificado.
- **erros** - Se este argumento de palavra-chave for especificado junto com o nome do arquivo, seu valor será usado quando o FileHandler for criado e, portanto, usado ao abrir o arquivo de saída. Se não for especificado, o valor 'backslashreplace' será usado. Observe que se None for especificado, ele será passado como tal para open(), o que significa que será tratado da mesma forma que passar 'erros'.

Exemplo

Para registrar todas as mensagens acima do nível DEBUG, defina o parâmetro de nível como logging.DEBUG

```
import logging

logging.basicConfig(level=logging.DEBUG)
logging.debug('This message will get logged')
```

Ele produzirá a seguinte **saída** -

```
DEBUG:root:This message will get logged
```

Para gravar as mensagens de log em um arquivo em vez de repeti-las no console, use o parâmetro filename.

```
import logging

logging.basicConfig(filename='logs.txt', filemode='w', level=logging.DEBUG)
logging.warning('This messagewill be saved to a file')
```

Nenhuma saída será exibida no console. No entanto, um arquivo logs.txt é criado no diretório atual com o texto **WARNING:root:Esta mensagem será salva em um arquivo** nele.

Dados variáveis na mensagem de registro

Na maioria das vezes, você gostaria de incluir valores de uma ou mais variáveis nas mensagens de log para obter mais informações sobre a causa, especialmente dos erros gerados enquanto o aplicativo está em execução. Para fazer isso, qualquer uma das técnicas de formatação dinâmica de strings, como o método format() da classe **str** ou strings f, pode ser usada.

Exemplo

```
import logging

logging.basicConfig(level=logging.DEBUG)
marks = 120
logging.error("Invalid marks:{} Marks must be between 0 to 100".format(marks))
subjects = ["Phy", "Maths"]
logging.warning("Number of subjects: {}. Should be at least three".format(len(subject
```

Ele produzirá a seguinte **saída** -

```
ERROR:root:Invalid marks:120 Marks must be between 0 to 100
WARNING:root:Number of subjects: 2. Should be at least three
```