Python - Adicionar itens de dicionário

Usando o Operador

O operador "[]" (usado para acessar o valor mapeado para uma chave de dicionário) é usado para atualizar um par chave-valor existente, bem como adicionar um novo par.

Sintaxe

```
dict["key"] = val
```

Se a chave já estiver presente no objeto dicionário, seu valor será atualizado para val. Se a chave não estiver presente no dicionário, um novo par chave-valor será adicionado.

Exemplo

Neste exemplo, as marcas de “Laxman” são atualizadas para 95.

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: ", marks)
marks['Laxman'] = 95
print ("marks dictionary after update: ", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update: {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update: {'Savita': 67, 'Imtiaz': 88, 'Laxman': 95, 'David': 49}
```

Exemplo

No entanto, um item com 'Krishnan' como chave não está disponível no dicionário, portanto, um novo par de valores-chave é adicionado.

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: ", marks)
marks['Krishan'] = 74
print ("marks dictionary after update: ", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update: {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update: {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49, 'Krishan': 74}
```

Usando o método update()

Você pode usar o método update() na classe dict de três maneiras diferentes:

Atualizar com outro dicionário

Neste caso, o argumento do método update() é outro dicionário. O valor das chaves comuns em ambos os dicionários é atualizado. Para novas chaves, o par chave-valor é adicionado ao dicionário existente

Sintaxe

```
d1.update(d2)
```

Valor de retorno

O dicionário existente é atualizado com novos pares de valores-chave adicionados a ele.

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
marks1 = {"Sharad": 51, "Mushtaq": 61, "Laxman": 89}
marks.update(marks1)
print ("marks dictionary after update: \n", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
{'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
{'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```

Atualizar com Iterável

Se o argumento para o método update() for uma lista ou tupla de duas tuplas de itens, um item para cada um será adicionado no dicionário existente ou atualizado se a chave existir.

Sintaxe

```
d1.update([(k1, v1), (k2, v2)])
```

Valor de retorno

O dicionário existente é atualizado com novas chaves adicionadas.

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
marks1 = [("Sharad", 51), ("Mushtaq", 61), ("Laxman", 89)]
marks.update(marks1)
print ("marks dictionary after update: \n", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```

Atualização com argumentos de palavras-chave

A terceira versão do método update() aceita uma lista de argumentos de palavras-chave no formato nome=valor. Novos pares kv são adicionados ou o valor da chave existente é atualizado.

Sintaxe

```
d1.update(k1=v1, k2=v2)
```

Valor de retorno

O dicionário existente é atualizado com novos pares de valores-chave adicionados.

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
```

```
marks.update(Sharad = 51, Mushtaq = 61, Laxman = 89)
print ("marks dictionary after update: \n", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```

Usando o operador Unpack

O símbolo "**" prefixado em um objeto de dicionário o descompacta em uma lista de tuplas, cada tupla com chave e valor. Dois objetos **dict** são descompactados e mesclados e obtêm um novo dicionário.

Sintaxe

```
d3 = {**d1, **d2}
```

Valor de retorno

Dois dicionários são mesclados e um novo objeto é retornado.

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
marks1 = {"Sharad": 51, "Mushtaq": 61, "Laxman": 89}
newmarks = {**marks, **marks1}
print ("marks dictionary after update: \n", newmarks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```

Usando o Operador União (|)

Python apresenta o "|" (símbolo de barra vertical) como operador de união para operandos de dicionário. Ele atualiza as chaves existentes no objeto dict à esquerda e adiciona novos pares de valores-chave para retornar um novo objeto dict.

Sintaxe

```
d3 = d1 | d2
```

Valor de retorno

O operador Union retorna um novo objeto dict após mesclar os dois operandos dict

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
marks1 = {"Sharad": 51, "Mushtaq": 61, "Laxman": 89}
newmarks = marks | marks1
print ("marks dictionary after update: \n", newmarks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```

Usando o operador "|="

O operador "|=" é um operador Union aumentado. Ele executa a atualização local no operando do dicionário à esquerda, adicionando novas chaves no operando à direita e atualizando as chaves existentes.

Sintaxe

```
d1 |= d2
```

Exemplo

```
marks = {"Savita":67, "Imtiaz":88, "Laxman":91, "David":49}
print ("marks dictionary before update: \n", marks)
marks1 = {"Sharad": 51, "Mushtaq": 61, "Laxman": 89}
```

```
marks |= marks1
print ("marks dictionary after update: \n", marks)
```

Ele produzirá a seguinte **saída** -

```
marks dictionary before update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 91, 'David': 49}
marks dictionary after update:
 {'Savita': 67, 'Imtiaz': 88, 'Laxman': 89, 'David': 49, 'Sharad': 51, 'Mushtaq': 61}
```