Python - Fluxo de controle

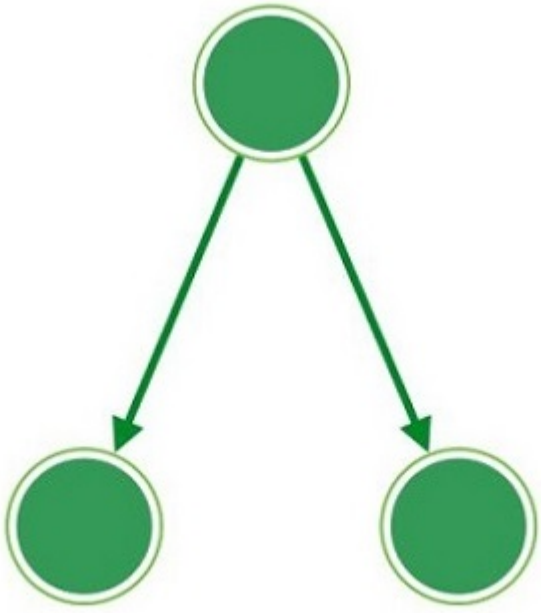
O fluxo de controle do programa Python é regulado por vários tipos de instruções condicionais , loops e chamadas de função . Por padrão, as instruções de um programa de computador são executadas de maneira sequencial, de cima para baixo ou do início ao fim. No entanto, esses programas de execução sequencial podem executar apenas tarefas simplistas. Gostaríamos que o programa tivesse capacidade de tomada de decisão, de modo que executasse diferentes etapas dependendo de diferentes condições.

A maioria das linguagens de programação, incluindo Python, fornece funcionalidade para controlar o fluxo de execução de instruções. Normalmente, existem dois tipos de instruções de fluxo de controle em qualquer linguagem de programação e Python também as suporta.

Declarações de tomada de decisão

Declarações de tomada de decisão são usadas nos programas Python para torná-los capazes de decidir qual grupo alternativo de instruções será executado, dependendo do valor de uma determinada expressão booleana.

O diagrama a seguir ilustra como funcionam as declarações de tomada de decisão -



As declarações if

Python fornece instruções de controle if..elif..else como parte da marcação de decisão. A seguir está um exemplo simples que faz uso de if..elif..else. Você pode tentar executar este programa usando marcas diferentes e verificar o resultado.

```
marks = 80
result = ""
```

```
if marks < 30:
   result = "Failed"
elif marks > 75:
   result = "Passed with distinction"
else:
   result = "Passed"

print(result)
```

Isso produzirá o seguinte resultado:

```
Aprovado com distinção
```

A declaração de correspondência

Python oferece suporte à instrução Match-Case , que também pode ser usada como parte da tomada de decisão. A seguir está um exemplo simples que faz uso da instrução match.

```
def checkVowel(n):
   match n:
      case 'a': return "Vowel alphabet"
      case 'e': return "Vowel alphabet"
      case 'i': return "Vowel alphabet"
      case 'o': return "Vowel alphabet"
      case 'u': return "Vowel alphabet"
      case _: return "Simple alphabet"
print (checkVowel('a'))
print (checkVowel('m'))
print (checkVowel('o'))
```

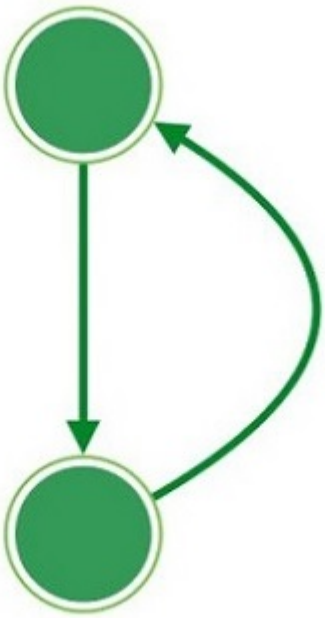
Isso produzirá o seguinte resultado:

```
Alfabeto vocálico
Alfabeto simples
Alfabeto vocálico
```

Loops ou instruções de iteração

A maioria dos processos exige que um grupo de instruções seja executado repetidamente. Na terminologia de programação, é chamado de **loop** . Em vez da próxima etapa, se o fluxo for redirecionado para qualquer etapa anterior, constitui um loop.

O diagrama a seguir ilustra como funciona o loop -



Se o controle voltar incondicionalmente, ele formará um loop infinito que não é desejado, pois o restante do código nunca seria executado.

Em um loop condicional, a iteração repetida do bloco de instruções continua até que uma determinada condição seja atendida. Python suporta vários loops, como o loop for e o loop while, que estudaremos nos próximos capítulos.

O loop for

O loop for itera sobre os itens de qualquer sequência, como uma lista, tupla ou string.

A seguir está um exemplo que faz uso do For Loop para iterar por meio de um array em Python:

```
words = ["one", "two", "three"]
for x in words:
  print(x)
```

Isso produzirá o seguinte resultado:

```
um
dois
três
```

O loop while

O loop while executa repetidamente uma instrução de destino, desde que uma determinada expressão booleana seja verdadeira.

A seguir está um exemplo que faz uso do While Loop para imprimir os primeiros 5 números em Python:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Isso produzirá o seguinte resultado:

```
1
2
3
4
5
```

Declarações de salto

As instruções de salto são usadas para saltar para uma instrução específica, interrompendo o fluxo atual do programa. Em Python, existem duas instruções de salto break e continue .

A declaração de ruptura

Termina o loop atual e retoma a execução na próxima instrução.

O exemplo a seguir demonstra o uso da instrução break -

```
x = 0

while x < 10:
    print("x:", x)
    if x == 5:
        print("Breaking...")
        break
    x += 1

print("End")
```

Isso produzirá o seguinte resultado:

```
x: 0
x: 1
x: 2
x: 3
x: 4
x: 5
```

```
Quebra...
Fim
```

A declaração de continuação

Ele pula a execução do bloco de programa e retorna o controle ao início do loop atual para iniciar a próxima iteração.

O exemplo a seguir demonstra o uso da instrução continue -

```
for letter in "Python":
    # continue when letter is 'h'
    if letter == "h":
        continue
    print("Current Letter :", letter)
```

Isso produzirá o seguinte resultado:

```
Letra Atual: P
Carta Atual: y
Carta Atual: t
Carta Atual: o
Carta Atual: n
```