centro universitário facvest
unifacvest

Fábrica de Software

Aula 08

Python

A função range()

A função range() retorna uma série numérica no intervalo enviado como argumento. É comum o uso da função range() com a estrutura for loop. Desta formta temos que a cada ciclo o próximo elemento da sequência será utilizado de tal forma que é possível partirmos de um ponto e ir incrementando, decrementando x unidades.

A função `range()` permite-nos especificar o início da sequência, o passo, e o valor final. O único parâmetro obrigatório é o que define quem será o último elemento da sequência.

- start - início da sequência
- stop - último elemento da sequência
- step - intervalo entre os elementos

```
#Exemplo range só com valor final
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

```
#Exemplo com começo e fim
list(range(2,12))
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
#Exemplo com início,fim e intervalo
list(range(2,12,2))
```

```
[2, 4, 6, 8, 10]
```

Break, Continue e Pass

O uso de loops do tipo “for” e loops do tipo “while” em Python permite que você automatize e repita tarefas de maneira eficiente.

No entanto, pode acontecer de um fator externo influenciar a maneira como seu programa é executado. Quando isso ocorre, é desejável que seu programa saia de um loop completamente, ignore parte de um loop antes de continuar, ou ignore aquele fator externo. É possível realizar essas ações com as instruções `break`, `continue` e `pass`.

Instrução break

Em Python, a instrução `break` oferece a possibilidade de sair de um loop quando uma condição externa é acionada. A instrução `break` será colocada dentro do bloco de código abaixo da sua instrução de loop, geralmente após uma instrução condicional `if`.

```
valor = 0

for valor in range(10):
    if valor == 5:
        break    # break ---pára aqui!

    print('valor =  ' + str(valor))

print('Out of loop')
```

```
valor =  0
valor =  1
valor =  2
valor =  3
valor =  4
Out of loop
```

Neste pequeno programa, o valor da variável é inicializado em 0. Em seguida, uma instrução for constrói o loop, desde que o valor da variável seja menor que 10. Dentro do loop for, há uma instrução if que apresenta a condição que *se* o valor da variável for equivalente ao número inteiro 5, *então* o loop será quebrado.

Dentro do loop também há uma instrução print() que será executada com cada iteração do loop for até que o loop seja quebrado, uma vez que está localizada após a instrução break.Para saber quando estamos fora do loop, incluímos uma instrução final print() fora do loop for.

Outro exemplo…

```
var = 10                    # Second Example
while var > 0:
    print ('Valor atual :', var)
    var = var -1
    if var == 5:
          break

print ("Fim!Obrigada!")
```

```
Valor atual : 10
Valor atual : 9
Valor atual : 8
Valor atual : 7
Valor atual : 6
Fim!Obrigada!
```

Tente entender esse exemplo mais complexo…

```
alunos = ['Silvia', 'Tina', 'Gil', 'Davi', 'Ryan', 'Teco']

for i in range(len(alunos)):
    print(alunos[i])
    if alunos[i] == 'Davi':
        print('Achamos o Davi!!')
        break


print('Loop terminou!!')
```

```
Silvia
Tina
Gil
Davi
Achamos o Davi!!
Loop terminou!!
```

Instrução continue

A instrução `continue` dá a opção de ignorar a parte de um loop onde uma condição externa é acionada, mas continuar e completar o resto do loop. Ou seja, a iteração atual do loop será interrompida, mas o programa retornará ao topo do loop.

A instrução `continue` ficará dentro do bloco de código abaixo da instrução de loop, geralmente após uma instrução condicional `if`.

```
valor = 0

for valor in range(10):
    if valor == 5:
        continue    # continue here

    print('Valor =  ' + str(valor))

print('Out of loop')
```

```
Valor =  0
Valor =  1
Valor =  2
Valor =  3
Valor =  4
Valor =  6
Valor =  7
Valor =  8
Valor =  9
Out of loop
```

A diferença entre usar a instrução continue, em vez de uma instrução break, é que o nosso código continuará apesar da interrupção quando a variável valor for avaliada como equivalente a 5. Aqui, valor = 5 nunca ocorre no resultado, mas o loop continua após esse ponto e imprime linhas para os números 6-10 antes de ser finalizado.

Você pode usar a instrução continue para evitar um código condicional extremamente aninhado, ou para otimizar um loop, eliminando casos que ocorram com frequência e que você gostaria de rejeitar.A instrução continue faz com que um programa pule certos fatores que surjam dentro de um loop, mas depois continuem pelo restante do loop.

Declaração pass

Quando uma condição externa é acionada, a instrução `pass` permite lidar com a condição sem que o loop seja impactado; todo o código continuará sendo lido a menos que um `break` ou outra instrução ocorra.

Assim como ocorre com outras instruções, a instrução `pass` ficará dentro do bloco de código abaixo da instrução de loop, normalmente após uma instrução condicional `if`.

```
valor = 0

for valor in range(10):
    if valor == 5:
        pass    # pass here

    print('valor =  ' + str(valor))

print('Out of loop')
```

```
valor =  0
valor =  1
valor =  2
valor =  3
valor =  4
valor =  5
valor =  6
valor =  7
valor =  8
valor =  9
Out of loop
```

A instrução `pass` que ocorre após a instrução condicional `if` está dizendo ao programa para continuar executando o loop e ignorar o fato de que a variável `number` é avaliada como equivalente a 5 durante uma das iterações.

Ao usar a instrução `pass` neste programa, notamos que o programa é executado exatamente como seria se não houvesse nenhuma instrução condicional no programa. A instrução `pass` diz ao programa para desconsiderar essa condição e continuar executando o programa como sempre.

A instrução `pass` pode criar classes mínimas, ou agir como um espaço reservado enquanto estamos trabalhando em novos códigos e pensando em um nível algorítmico antes de construir detalhes.

```
for letter in 'Python':
    if letter == 'h':
        pass
        print ('This is pass block')
    print ('Current Letter :', letter)

print ("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```

EXERCÍCIOS

1) Crie uma lista vazia e uma variável com valor 4. Enquanto o valor da variável for menor ou igual a 20,

adicione à lista, apenas os valores pares e imprima a lista

2)Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.
3)Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.
4) Faça um programa que leia e valide as seguintes informações:

a. Nome: maior que 3 caracteres;
b. Idade: entre 0 e 150;
c. Salário: maior que zero;
d. Sexo: 'f' ou 'm';
e. Estado Civil: 's', 'c', 'v', 'd';

Use a função len(string) para saber o tamanho de um texto (número de caracteres).

5) Faça um programa que peça a idade e o sexo de varíos usuários.No final ele deve mostrar as estatísticas (quantidade de usuário com mais de 18 anos e dos seos fem e masc). Seu programa deve perguntar seu o usuário quer parar de cadastrar ou continuar. Quando ele escolhe parar, as estatísticas aparecem. Quando ele escolhe continuar, ele continua cadastrando.

6) Faça o programa de uma papelaria. Você passa a lista de compras (produtos e valores) e o programa mostra o valor final, o produto mais barato e o de menor valor.