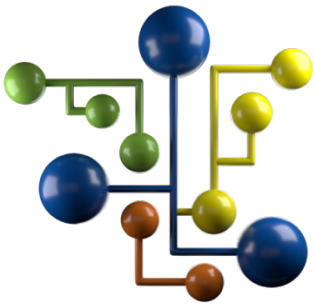

Data Science Academy

www.datascienceacademy.com.br

# Introdução à Lógica de Programação

Você deve compreender 100%, duas estruturas fundamentais em Programação:

- Estrutura Condicional
- Estrutura de Repetição

A Estrutura Condicional if/else, por exemplo, tem a lógica:

Se uma instrução for verdadeira, faça isso, senão, faça aquilo.

Assim, damos ao computador a chance de tomar decisões com base em regras estabelecidas por nós.

Mas em muitos casos precisamos repetir uma operação um determinado número de vezes. Precisamos, portanto, de uma estrutura de repetição.

Vejamos um exemplo:

E se eu pedisse a você para criar um programa e somar os dígitos de 1 a 5?

(ou seja: 1 + 2 + 3 + 4 + 5 = 15).

Como você resolveria isso? Aqui um pseudocódigo:

1- Define e inicializa a variável soma com 0.
2- Define e inicializa a variável i com 1.
3- A variável soma recebe seu próprio valor mais i, por 5 vezes.
4- Incrementa i
5- Retorna soma

Em Python, ficaria assim:

```
# Função
def soma():
----soma = 0
----i = 1

----soma = soma + i
----i = i + 1
----soma = soma + i
----i = i + 1
----soma = soma + i
----i = i + 1
----soma = soma + i
----i = i + 1
```

```
----soma = soma + i

----return soma

# Excecuta função
soma()
```

Resultado: 15

Perceba no código acima que vamos mudando o valor de i a cada execução e agrupando a adição na variável soma. Funciona. Mas será que é a melhor opção?

E se quisermos somar os dígitos de 1 a 6? Teríamos que adicionar mais 2 linhas à nossa função. E se quisermos adicionar os dígitos de 1 a 1000? Adicionamos 1000 linhas em nossa função? Nem pensar! kkkkkk. Temos que automatizar o processo de alguma forma. Usaremos, portanto, estruturas de repetição ou Loops, como por exemplo o Loop For.

Pseudocódigo:

1- Define e inicializa a variável soma com 0.

2- Para cada valor de i começando por 1, repetimos por n vezes:

2.1- A variável soma recebe seu próprio valor mais i

3- Retorna soma

Em Python, ficaria assim:

```
# Função
def soma(n):
----soma = 0
----for i in range(n+1):
--------soma = soma + i
----return soma

# Excecuta função
soma(5)
soma(1000)
```

Perceba que automatizamos nosso código. Podemos passar qualquer valor de n e assim teremos a soma de todos os dígitos até esse valor.

O Loop For repete uma instrução um número determinado de vezes. Mas observe alguns detalhes:

1- Não precisamos inicializar a variável i. Python faz isso por nós (em outras linguagens isso é diferente).

2- Eu usei n + 1. Por quê? Se você acompanhou o curso gratuito de Python já sabe a resposta: a função range usa o valor como "exclusivo", ou seja, se considerarmos n = 5, teríamos os valores 0, 1, 2, 3 e 4. Mas queremos até o dígito 5. Por isso adicionei +1.

3- O Loop For faz o incremento da variável i automaticamente em Python (em outras linguagens isso é diferente).

Não à toa Python é considerada uma das linguagens de programação mais fáceis da atualidade, o que ajuda a explicar seu tamanho sucesso!

Mas o Loop For não é a única forma de resolver isso. Podemos usar o Loop While:

Enquanto uma condição for verdadeira, executa uma ou mais instruções.

Aqui nossa função soma, usando Loop While e gerando o mesmo resultado:

```
# Função
def soma(n):
----soma = 0
----i = 1
----while i <= n:
--------soma = soma + i
--------i = i + 1
----return soma

# Excecuta função
soma(5)
soma(1000)
```

Algumas considerações:

1- No Loop For, SEMPRE entraremos no Loop e executaremos a instrução pelo menos uma vez e saímos do Loop quando a condição chegar ao fim.

2- No Loop While, SOMENTE entraremos no Loop se a condição for verdadeira. E somente sairemos do Loop quando a condição se tornar falsa.

Por isso, cabe ao programador, garantir que em algum momento a condição mude dentro do Loop, e assim colocamos i = i + 1, pois em algum momento i deixará de ser menor que n e a condição passará a ser falsa.

3- Observe ainda que devemos inicializar i antes do Loop While. No Loop For isso não é necessário, embora possa ser feito.

Essas não são as duas únicas estruturas de repetição, mas são as mais usadas.

A escolha entre uma ou outra depende de como você quer controlar a execução do seu programa!

#aula04