Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES
Programação I / Programação Funcional
Prof. Oberlan Romão

Aula Prática 2

O objetivo dessa aula é colocar em prática tudo que já foi visto e, em especial, o uso de `if-else`. Além disso, resolveremos alguns problemas da Olimpíada Brasileira de Informática :).

Conforme os programas ficam maiores e mais complicados, eles são mais difíceis de ler. As linguagens formais são densas e muitas vezes é difícil ver um trecho de código e compreender o que ele faz ou por que faz isso. Por essa razão, é uma boa ideia acrescentar notas aos seus programas para explicar em linguagem natural o que o programa está fazendo. Essas notas são chamadas de **comentários** e começam, em Python, com o símbolo #.

Crie um arquivo, `ex1.py` e cole o código abaixo:

```
1 nota1 = 8 # Nota da Prova 1
2 nota2 = 7.5 # Nota da Prova 2
3 nota3 = 8.2 # Nota da Prova 3
4 media = (nota1 + nota2 + nota3) / 3.0
5 # print(f"Média = {media:.2f}")
6 if media < 7.0:
7     print("\nAluno de prova final\n")
8 else:
9     print("\nAluno aprovado\n")
```

Execute o *script*. Perceba que tudo que está depois do # até o final da linha é totalmente ignorado (inclusive os `print` da linha 5). Descomente a linha 5 (apagando o #) e execute o *script* novamente. Perceba que existem algumas coisas que são "novas" dentro do `print`. Na linha 5, queremos imprimir o valor da média, mas é interessante imprimirmos esse valor com uma certa quantidade de casas decimais, nesse caso, duas casas decimais, ou seja, devemos **formatar** a impressão desse valor. Para fazermos isso, devemos usar o seguinte padrão `{<nomeVariavel ou expressao>:.<numCasasDecimais>f}`. Por exemplo, se quisermos imprimir o valor contido na variável `media` com 5 casas decimais, devemos fazer: `print(f"{media:.5f}")`. Na linha 5, troque `print(f"Média = {media:.2f}")` por `print(f"Média = {media}")` e veja a diferença. Veremos mais detalhes e exemplos de como imprimir formatado nas próximas aulas.

Vamos agora analisar outro detalhe desse código. Observe que no `print` da linha 7 e da linha 9 contém um caractere/comando diferente: `\n`. Esse comando representa uma nova linha. Experimente colocar um `\n` logo após "Aluno" e execute para ver o resultado. Troque algum `\n` por `\n\n` e veja o que acontece.
Dica: No VSCode, para comentar um trecho de código você pode usar o atalho "`Ctrl + ;`". No Sublime, o atalho é "`Ctrl + /`".

Agora vamos analisar alguns códigos:

1. Crie um *script*, `ex2.py`, e cole o código abaixo:

```
x = input("Digite um número: ")
if x%2 = 0:
    print(f"{x} é par")
if x%2 = 1:
    print(f"{x} é impar")
if x%2 = 0 or x%3 = 0:
    print("...")
if x%2 = 0 and x%5 = 0:
    print("...")
```

   Execute o *script*. O que aconteceu? Por quê? Corrija os erros e substitua os `...`, dentro do dois últimos `print`, por mensagens que façam sentido.

   Pergunta: O segundo `if` pode ser substituído por uma clausula `else`? E o último?

2. (`ex3.py`) O código a seguir contém várias instruções `if-else`. Infelizmente, foi escrito sem o uso correto da indentação (recuo). Reescreva o código e use as convenções apropriadas de indentação. Além disso, corrija outros possíveis erros.

```
if media >= 9.0
print("A")
else:
if media >= 8.0
print("B")
else:
if media >= 7.0
print("C")
else:
if media >= 6.0
print("D")
else:
print("R")
```

Verificando seu programa - Parte 1

Uma das formas de verificar se um programa está correto é fazer vários testes com diferentes entradas. Para cada entrada o seu programa irá gerar uma saída e para saber se seu programa está correto, basta verificar se a saída gerada pelo programa corresponde a saída esperada.

Em muitos casos é muito difícil (ou até impossível) saber se um programa funciona para todos os possíveis valores de entrada. Por exemplo, se um programa deve receber um inteiro como entrada, faz sentido testarmos todos os inteiros? Isso seria possível? Uma alternativa é testá-lo com um subconjunto de valores. Esse subconjunto, normalmente, são valores que possuem mais probabilidade de gerarem problemas (*bugs*) ao programa.

Vamos a um exemplo. Suponha que um programador precise fazer um programa para verificar se um número inteiro é positivo ou negativo e lhe foi dado a tabela abaixo contendo oito **casos de testes**. Cada

caso de teste, contém um valor de entrada (valores digitados) com a respectiva saída esperada (resposta que o programa deve gerar).

Caso de teste	Entrada	Saída esperada
1	1	1 é positivo
2	-10	-10 é negativo
3	100	100 é positivo
4	-100	-100 é negativo
5	153	153 é positivo
6	854	854 é positivo
7	-21	-21 é negativo
8	0	0 é positivo

O programador, rapidamente, apresentou o seguinte código:

```
x = int(input())
if x > 0:
    print(f"{x} eh positivo")
else:
    print(f"{x} eh negativo")
```

Crie um arquivo, `ex4.py`, que contenha o código anterior. Teste o programa com os casos de testes apresentados na tabela e outros casos de testes que você desejar.

Pergunta: O código apresentado pelo programador gera exatamente a mesma saída esperada para o conjunto de entrada apresentado na tabela anterior? Em outras palavras, se for digitado, por exemplo, `1` no programa, a saída gerada será exatamente `1 é positivo`? Corrija o código para que ele gere a saída esperada para todas as entradas.

Exercícios da OBI

A Olimpíada Brasileira de Informática (OBI), organizada pela Sociedade Brasileira de Computação (SBC) e pelo Instituto de Computação da UNICAMP, é uma competição organizada nos moldes das outras olimpíadas científicas brasileiras, como Matemática, Física e Astronomia. O objetivo da OBI é despertar nos alunos o interesse por uma ciência importante na formação básica hoje em dia (no caso, ciência da computação), através de uma atividade que envolve desafio, engenhosidade e uma saudável dose de competição. Podem participar da OBI estudantes do ensino médio e estudantes que estejam no primeiro ano da graduação. Para saber mais detalhes sobre a OBI, acesse o site https://olimpiada.ic.unicamp.br.

Resolva os seguintes exercícios da OBI.

1. https://olimpiada.ic.unicamp.br/pratique/pj/2018/f1/basquete/
2. https://olimpiada.ic.unicamp.br/pratique/pj/2017/f2/cartas/
3. https://olimpiada.ic.unicamp.br/pratique/p1/2018/f1/xadrez/

Obs.: O objetivo desses exercícios é fixar o uso dos comandos condicionais. Então, não use funções integradas do Python (como `min` ou `max`).