MC-102 — Aula 09
Comandos Repetitivos

Instituto de Computação – Unicamp

15 de Setembro de 2016

Roteiro

1. Laços Encaixados
   - Números Primos
   - Dados
   - Mega-Sena
2. Exercícios

Laços Encaixados: Primos

- A geração de números primos é uma parte fundamental em sistemas criptográficos como os utilizados em *internetbanking*.
- Já sabemos testar se um determinado número é ou não primo.
- Imagine que agora queremos imprimir os n primeiros números primos.
- O que podemos fazer?

Laços Encaixados: Primos

- O programa abaixo verifica se o valor na variável **candidato** corresponde a um primo:

```
divisor = 2
eprimo = True
while (divisor <= candidato/2) and eprimo:
  if(candidato % divisor == 0):
     eprimo = False
  divisor = divisor + 1

if(eprimo):
  print(candidato)
```

Laços Encaixados: Primos

- Criamos um laço externo e usamos uma variável contadora **primosImpressos**, que contará o número de primos impressos durante a execução deste laço.

```
while primosImpressos < n:

  //trecho do código anterior que
  //checa se candidato é ou não é primo

  if(eprimo):
    print(candidato)
    primosImpressos = primosImpressos + 1

  candidato = candidato + 1 #Testa próximo número candidato a primo
```

Laços Encaixados: Primos

- Incluímos uma parte inicial de código para leitura de **n** e inicialização de variáveis.
- Para finalizar, basta incluir o trecho de código que checa se um número é primo ou não.

```
n = int(input("Digite quantidade de primos: "))
candidato = 2
primosImpressos = 0
while primosImpressos < n:

    #trecho do código que checa
    #se candidato é ou não é primo

    if(eprimo):
        print(candidato)
        primosImpressos = primosImpressos + 1
    candidato = candidato + 1
```

Laços Encaixados: Primos

Código completo:

```
n = int(input("Digite quantidade de primos: "))
candidato = 2
primosImpressos = 0
while primosImpressos < n:
  divisor = 2
  eprimo = True
  while (divisor <= candidato/2) and eprimo :
    if(candidato % divisor == 0):
      eprimo = False
    divisor = divisor + 1

  if(eprimo):
    print(candidato)
    primosImpressos = primosImpressos + 1
  candidato = candidato + 1
```

Laços Encaixados: Primos

- O que acontece se mudarmos a variável indicadora **eprimo** para fora do primeiro laço **while**? Faz diferença?

```
n = int(input("Digite quantidade de primos: "))
candidato = 2
primosImpressos = 0
eprimo = True    # **************************** <--Saiu do laço, faz diferença?
while primosImpressos < n:
  divisor = 2
  while (divisor <= candidato/2) and eprimo :
    if(candidato % divisor == 0):
      eprimo = False
    divisor = divisor + 1

  if(eprimo):
    print(candidato)
    primosImpressos = primosImpressos + 1
  candidato = candidato + 1
```

Laços Encaixados: Primos

- O que acontece se mudarmos a variável indicadora **eprimo** para fora do primeiro laço **while**? Faz diferença?
- Resposta: Quando testarmos um **candidato** que não é primo, a variável **eprimo** será setada para **False** e nunca mais será setada para **True**.
- Logo nenhum outro **candidato** posterior será identificado como primo.

```
n = int(input("Digite quantidade de primos: "))
candidato = 2
primosImpressos = 0
eprimo = True    # *************************** <--Saiu do laço, faz diferença?
while primosImpressos < n:
  divisor = 2
  while (divisor <= candidato/2) and eprimo :
    if(candidato % divisor == 0):
      eprimo = False
    divisor = divisor + 1

  if(eprimo):
    print(candidato)
    primosImpressos = primosImpressos + 1
  candidato = candidato + 1
```

Laços Encaixados: Primos

- Note que o número 2 é o único número par que é primo.
- Podemos alterar o programa para sempre imprimir o número 2:

```
n = int(input("Digite quantidade de primos: "))
if(n>0):
    print(2)
    primosImpressos = 1
    .
    .
    .
```

Laços Encaixados: Primos

- Podemos alterar o programa para testar apenas números ímpares como candidatos a primo:

```
n = int(input("Digite quantidade de primos: "))
if(n>0):
    print(2)
    primosImpressos = 1
    candidato = 3 #primeiro candidato a primo
    while primosImpressos < n:
        divisor = 2
        eprimo = True
        while (divisor <= candidato/2) and eprimo :
            if(candidato % divisor == 0):
                eprimo = False
            divisor = divisor + 1
        if(eprimo):
            print(candidato)
            primosImpressos = primosImpressos + 1
        candidato = candidato + 2 #testa próximo candidato a primo
```

Laços Encaixados: Primos

Além disso sabendo que **candidato** é sempre um número ímpar:

- Não precisamos mais testar os divisores que são pares.
- Se **candidato** é sempre um número ímpar, ele não pode ser divisível por um número par, pois seria divisível por 2 também.
- Portanto basta testar divisores ímpares.

Laços Encaixados: Primos

```
n = int(input("Digite quantidade de primos: "))
if(n>0):
    print(2)
    primosImpressos = 1
    candidato = 3 #primeiro candidato a primo
    while primosImpressos < n:
        divisor = 3 #primeiro divisor ímpar
        eprimo = True
        while (divisor <= candidato/2) and eprimo :
            if(candidato % divisor == 0):
                eprimo = False
            divisor = divisor + 2 #próximo divisor ímpar
        if(eprimo):
            print(candidato)
            primosImpressos = primosImpressos + 1
        candidato = candidato + 2 #testa próximo candidato a primo
```

Laços Encaixados: Dados

Problema

Imprimir todas as possibilidades de resultados ao se jogar 4 dados de 6 faces.

- Para cada possibilidade do primeiro dado, devemos imprimir todas as possibilidades dos 3 dados restantes.
- Para cada possibilidade do primeiro e segundo dado, devemos imprimir todas as possibilidades dos 2 dados restantes....
- Você consegue pensar em uma solução com laços aninhados?

Laços Encaixados: Dados

```
print("D1 D2 D3 D4")
for d1 in range(1, 7):
    for d2 in range(1, 7):
        for d3 in range(1, 7):
            for d4 in range(1, 7):
                print(d1,d2,d3,d4)
```

Laços Encaixados: Mega-Sena

- Na Mega-Sena, um jogo consiste de 6 números distintos com valores entre 1 e 60.

Problema

Imprimir todos os jogos possíveis da Mega-Sena

Laços Encaixados: Mega-Sena

- Partimos da mesma idéia dos dados: gerar todos os possíveis valores para cada um dos 6 números do jogo.

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

- Qual a saída deste programa? Ele está correto?

Laços Encaixados: Mega-Sena

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

- As primeiras linhas impressas por este programa serão:

```
1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 2
1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 4
1, 1, 1, 1, 1, 5
1, 1, 1, 1, 1, 6
1, 1, 1, 1, 1, 7
1, 1, 1, 1, 1, 8
1, 1, 1, 1, 1, 9
```

Laços Encaixados: Mega-Sena

- O programa anterior repete números, portanto devemos remover repetições.

```
for d1 in range(1, 61):
    for d2 in range(1, 61):
        for d3 in range(1, 61):
            for d4 in range(1, 61):
                for d5 in range(1, 61):
                    for d6 in range(1, 61):
                        if(d1 != d2 and d2 != d3 ... and d5!=d6)
                            print(d1,d2,d3,d4,d5,d6)
```

- Após incluir todos os testes para garantir que os números são distintos, temos a solução?

Laços Encaixados: Mega-Sena

- Não temos uma solução válida, pois o programa irá imprimir jogos como:

```
12, 34, 8, 19, 4, 45
34, 12, 8, 19, 4, 45
34, 12, 19, 8, 4, 45
```

- Na verdade, todos estes jogos são um único jogo: 4, 8, 12, 19, 34, 45.
- Podemos assumir que um jogo é sempre apresentado com os números em ordem crescente.
- Dado que fixamos o valor de **d1**, **d2** necessariamente é maior que **d1**. E com **d1** e **d2** fixados, d3 é maior que d2 etc.

Laços Encaixados: Mega-Sena

Solução correta:

```
for d1 in range(1, 61):
    for d2 in range(d1+1, 61):
        for d3 in range(d2+1, 61):
            for d4 in range(d3+1, 61):
                for d5 in range(d4+1, 61):
                    for d6 in range(d5+1, 61):
                        print(d1,d2,d3,d4,d5,d6)
```

Exercício

- Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

Exercício

- Faça um programa que leia um número n e imprima n linhas na tela com o seguinte formato (exemplo se $n = 6$):

```
+ * * * * *
* + * * * *
* * + * * *
* * * + * *
* * * * + *
* * * * * +
```

Exercício

- Um jogador da Mega-Sena é supersticioso, e só faz jogos em que o primeiro número do jogo é par, o segundo é ímpar, o terceiro é par, o quarto é ímpar, o quinto é par e o sexto é ímpar. Faça um programa que imprima todas as possibilidades de jogos que este jogador supersticioso pode jogar.