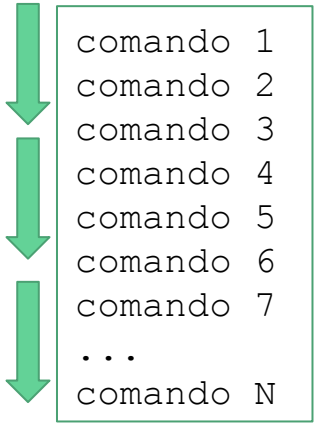
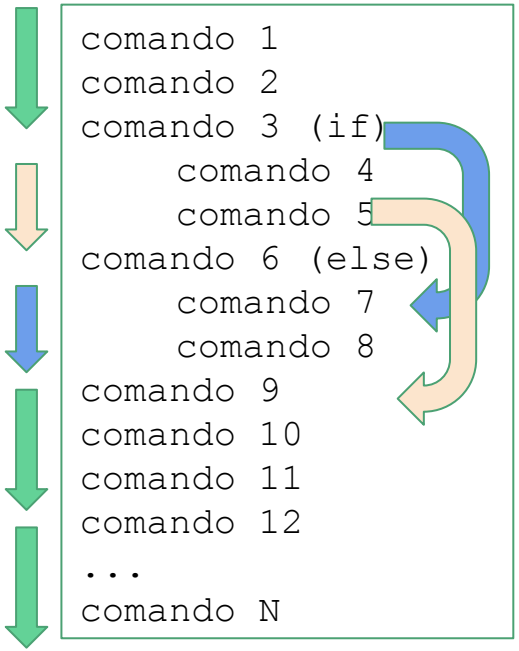
MC102 - Algoritmos e Programação de Computadores

Turma Z - Segundo Semestre de 2019

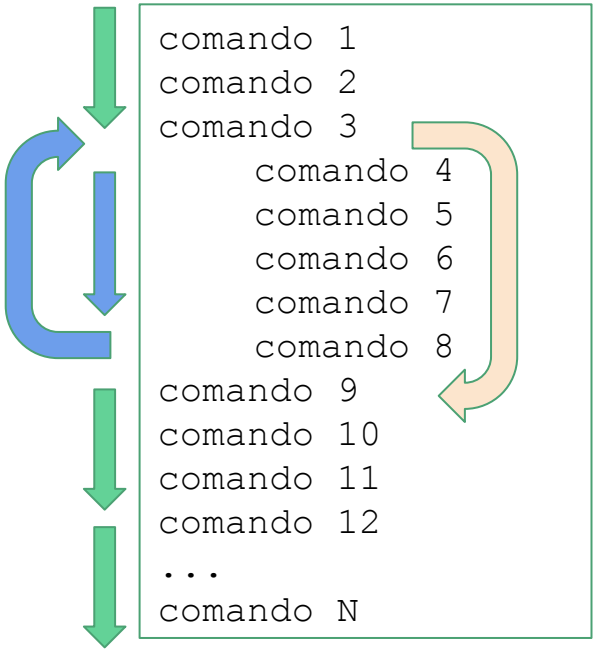
1ª semana



2ª semana



Agora



A partir desse slide, os slides com fundo cinza claro são parte do material desenvolvido pela professora Sandra Avila e disponível em http://www.ic.unicamp.br/~sandra/

Comando `while`

Comando `while`

- Executa um bloco de comando(s) enquanto a condição é verdadeira (`True`).

```
while condicao:
    comando(s)
```

Comando `while`

- Programa que imprime os *n* primeiros números.

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
numero = 1
while numero <= n:
    print(numero)
    numero = numero + 1
```

Comando `for`

Comando `for`

- É a estrutura de repetição mais usada no Python.
- Para cada elemento da lista, em ordem de ocorrência, é atribuído este elemento à variável e então é executado o(s) comando(s).

```
for variável in lista:
    comando(s)
```

Comando `for` e a função `range`

- Programa que imprime os *n* primeiros números.

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
for numero in range(1,n+1):
    print(numero)
```

while e for

- Programa que imprime os *n* primeiros números.

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
numero = 1
while numero <= n:
    print(numero)
    numero = numero + 1
```

```
# Imprime os n primeiros números
n = int(input("Digite um número: "))
for numero in range(1,n+1):
    print(numero)
```

Agenda

- Variável indicadora
- Variável contadora

Comandos de Repetição

- Vimos quais são os comandos de repetição em Python.
- Veremos mais alguns exemplos de sua utilização.

```
while condicao:
    comando(s)
```

```
for variável in lista:
    comando(s)
```

Variável Indicadora

Variável Indicadora

- Um uso comum de laços é para a verificação se um determinado objeto, ou conjunto de objetos, satisfaz uma propriedade ou não.
- Um padrão que pode ser útil na resolução deste tipo de problema é o uso de uma **variável indicadora**.
  - Assumimos que o objeto satisfaz a propriedade (**indicadora = `True`**).
  - Com um laço verificamos se o objeto realmente satisfaz a propriedade.
  - Se em alguma iteração descobrirmos que o objeto não satisfaz a propriedade, então fazemos **indicadora = `False`**.

Exemplo: Número Primo

- Problema: Determinar se um número *n* é primo ou não.

- Um número é primo se seus únicos divisores são 1 e ele mesmo.

Exemplo: Número Primo

- Problema: Determinar se um número *n* é primo ou não.

- Um número é primo se seus únicos divisores são 1 e ele mesmo.
- Dado um número *n* como detectar se este é ou não primo?
  - Leia o número *n*.
  - Teste se nenhum dos números entre 2 e (*n* - 1) divide *n*.

- Lembre-se que o operador % retorna o resto da divisão.
- Portanto (a % b) é zero se e somente se b divide a.

Exemplo: Número Primo

- Dado um número *n* como detectar se este é ou não primo?
  - Leia o número *n*.
  - Faça a variável **indicadora = `True`**, assumindo que é primo.
  - Teste se nenhum dos números entre 2 e (*n* - 1) divide *n*.
  - Se o resto da divisão for igual a zero então faça **indicadora = `False`**. Com isto descobrimos que não é primo.

Exemplo: Número Primo

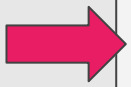
```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
primo = True # primo é a variável indicadora

while (numero <= n-1) and (primo):
    if (n % numero == 0): # se n é divisível por numero
        primo = False
    numero = numero + 1

if (primo):
    print("É primo.")
else:
    print("Não é primo.")
```

Exemplo: Número Primo



```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
primo = True # primo é a variável indicadora

while (numero <= n-1) and (primo):
    if (n % numero == 0): # se n é divisível por numero
        primo = False
    numero = numero + 1

if (primo):
    print("É primo.")
else:
    print("Não é primo.")
```

Exemplo: Número Primo (com `break`)

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
primo = True # primo é a variável indicadora

while (numero <= n-1):
    if (n % numero == 0): # se n é divisível por numero
        primo = False
        break
    numero = numero + 1

if (primo):
    print("É primo.")
else:
    print("Não é primo.")
```

Exemplo: Números em Ordem Crescente

- Problema: Fazer um programa que lê *n* números inteiros do teclado, e no final informa se os números lidos estão ou não em ordem crescente.
- Usaremos uma variável indicadora na resolução deste problema.

Exemplo: Números em Ordem Crescente

- Um laço principal será responsável pela leitura dos números.
- Vamos usar duas variáveis, uma que guarda o número lido na iteração atual, e uma que guarda o número lido na iteração anterior.
- Os números estarão ordenados se a condição (anterior <= atual) for válida durante a leitura de todos os números.

```
n = int(input("Digite um número: "))
anterior = int(input())

i = 1 # leu um número
ordenado = True # ordenado é a variável indicadora

while (i < n) and (ordenado):
    atual = int(input())
    i = i + 1 # leu mais um número
    if (atual < anterior):
        ordenado = False
    anterior = atual

if (ordenado):
    print("Sequência está ordenada.")
else:
    print("Sequência não está ordenada.")
```

Variável Contadora

Variável Contadora

- Considere ainda o uso de laços para a verificação se um determinado objeto, ou conjunto de objetos, satisfaz uma propriedade ou não.
- Um outro padrão que pode ser útil é o uso de uma **variável contadora**.
    - Esperamos que um objeto satisfaça x vezes uma sub-propriedade. Usamos um laço e uma variável que **conta** o número de vezes que o objeto tem a sub-propriedade satisfeita.
    - Ao terminar o laço, se a variável contadora for igual à x então o objeto satisfaz a propriedade.

Exemplo: Número Primo

- Problema: Determinar se um número *n* é primo ou não.

- Um número n é primo se nenhum número de 2 até (*n* - 1) dividi-lo.
  - Podemos usar uma variável que **conta** quantos números dividem *n*.
  - Se o número de divisores for 0, então *n* é primo.

Exemplo: Número Primo

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
divisores = 0 # divisores é a variável contadora

while (numero <= n-1):
    if (n % numero == 0): # se n é divisível por numero
        divisores = divisores + 1
    numero = numero + 1

if (divisores == 0):
    print("É primo.")
else:
    print("Não é primo.")
```

Exemplo: Número Primo

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
divisores = 0 # divisores é a variável contadora

while (numero <= n-1) and (divisores == 0):
    if (n % numero == 0): # se n é divisível por numero
        divisores = divisores + 1
    numero = numero + 1

if (divisores == 0):
    print("É primo.")
else:
    print("Não é primo.")
```

É melhor terminar o laço assim que descobrirmos algum divisor de *n*.

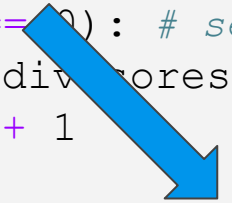
Exemplo: Número Primo

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
divisores = 0 # divisores é a variável contadora

while (numero <= n-1) and (divisores == 0):
    if (n % numero == 0): # se n é divisível por numero
        divisores = divisores + 1
    numero = numero + 1

if (divisores == 0):
    print("É primo.")
else:
    print("Não é primo.")
```



Basta testarmos até *n*/2. Por que?

Faça um programa que lê um número *n* e imprima os valores entre 2 e *n*, que são divisores de *n*.

```
n = int(input("Digite um número inteiro positivo: "))

numero = 2
while numero <= n:
    if (n % numero == 0): # se n é divisível por numero
        print(numero, end=" ")
    numero = numero + 1
```

Fizemos esse programa na aula passada

Resumo

- O uso de variáveis **acumuladora**, **indicadora** e **contadora** são úteis em várias situações.
- Mas não existem fórmulas para a criação de soluções para problemas.
- Em outros problemas, o uso destes padrões pode aparecer em conjunto, ou nem mesmo aparecer como parte da solução.

Agenda

- Laços encaixados
- Exercício

Laços Encaixados

- Para resolver alguns problemas, é necessário implementar um laço dentro de outro laço.
- Estes são laços encaixados.

```
for i in range(1,11):
    for j in range(1,6):
        print(i, j)
```

- O que será impresso por este programa?

Laços Encaixados

```
for i in range(1,11):
    for j in range(1,6):
        print(i, j)
```

- Fixado um valor para **i** no primeiro laço **for**, começa-se o segundo laço **for**, que varia o valor de **j** entre 1 e 5.

Laços Encaixados

```
for i in range(1,11):
    for j in range(1,6):
        print(i, j)
```

- Fixado um valor para **i** no primeiro laço **for**, começa-se o segundo laço **for**, que varia o valor de **j** entre 1 e 5.
- No final deste segundo laço **for**, voltamos para o primeiro laço onde a variável **i** assumiria seu próximo valor. Fixado este valor de **i** começa-se novamente o segundo laço **for**.

Laços Encaixados

```
for i in range(1,11):
    for j in range(1,6):
        print(i, j)
```

Laços Encaixados

```
for i in range(1,11):
    for j in range(1,6):
        print(i, j)
```

```
1 1
1 2
1 3
1 4
1 5
2 1
…
10 5
```

Exercício: Corrida de Lesmas (Lab. 2018/1)

https://www.ic.unicamp.br/~mc102/mc102-1s2018/labs/roteiro-lab05.html

Corrida de Lesmas

- A tarefa de capturar lesmas velozes não é uma tarefa muito fácil, pois praticamente todas as lesmas são muito lentas. Cada lesma é classificada em um nível dependendo de sua velocidade:
  - **Nível 1**: Se a velocidade é menor que 10 cm/h.
  - **Nível 2**: Se a velocidade é maior ou igual a 10 cm/h e menor que 20 cm/h.
  - **Nível 3**: Se a velocidade é maior ou igual a 20 cm/h.
- Sua tarefa é identificar qual nível de velocidade da lesma mais veloz de um grupo de lesmas.

Corrida de Lesmas

- A **entrada** consiste várias linhas:
  - A primeira linha contém um inteiro L ($1 \leq L \leq 100$) representando o número de lesmas do grupo.
  - As outras L linhas contêm inteiros Vi ($1 \leq Vi \leq 50$) representando as velocidades de cada lesma do grupo.

Corrida de Lesmas

- Para a saída, imprima uma única linha indicando o nível de velocidade da lesma mais veloz do grupo.
- Caso algum dos valores esteja fora dos intervalos estabelecidos, uma mensagem de erro deve ser emitida indicando a linha em que o erro ocorreu:

  `Valor inválido na linha <l>.`

Corrida de Lesmas

Entrada	Saída
5 12 9 8 7 6	2

Entrada	Saída
10 1 5 2 9 5 5 8 4 4 3	1

Entrada	Saída
5 10 90 8 7 6	Valor inválido na linha 3.

```
lesmas = int(input("Digite o número de lesmas do grupo: " ))
maior = 0
erro = False

if (1 <= lesmas <= 100):  # condição do número de lesmas
    for linha in range(lesmas):
        velocidade = int(input())
        if (1 <= velocidade <= 50):  # condição da velocidade da lesma
            if (maior < velocidade):
                maior = velocidade
        else:
            print("Valor inválido na linha "  + str(linha+2) + ".")
            erro = True

    if (not erro):
        if (maior < 10):
            print("A lesma mais veloz está no nível 1." )
        elif (maior < 20):
            print("A lesma mais veloz está no nível 2." )
        else:
            print("A lesma mais veloz está no nível 3." )
else:
    print("Valor inválido na linha 1." )
```

```
lesmas = int(input())
maior = 0
erro = False

if (1 <= lesmas <= 100):  # condição do número de lesmas
    for linha in range(lesmas):
        velocidade = int(input())
        if (1 <= velocidade <= 50):  # condição da velocidade da lesma
            if (maior < velocidade):
                maior = velocidade
        else:
            print("Valor inválido na linha " + str(linha+2) + ".")
            erro = True

    if (not erro):
        if (maior < 10):
            print("1")
        elif (maior < 20):
            print("2")
        else:
            print("3")
else:
    print("Valor inválido na linha 1.")
```

Para funcionar no SuSy =)

Exercício

Imprima o primeiro número primo maior que Y para:

- Y = 3
- Y = 20
- Y qualquer.

Postulado de Bertrand: Para *n* natural maior que 1, existe ao menos um número primo entre *n* e *2n*

Mais Exercícios =)

- https://wiki.python.org.br/EstruturaDeRepeticao: 51 exercícios \o/

- Curso de Python:
  - https://www.codecademy.com/learn/learn-python-3

Referências & Exercícios

- Os slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)

- https://wiki.python.org.br/EstruturaDeRepeticao: 51 exercícios \o/
- https://panda.ime.usp.br/pensepy/static/pensepy/07-Iteracao/maisiteracao.html