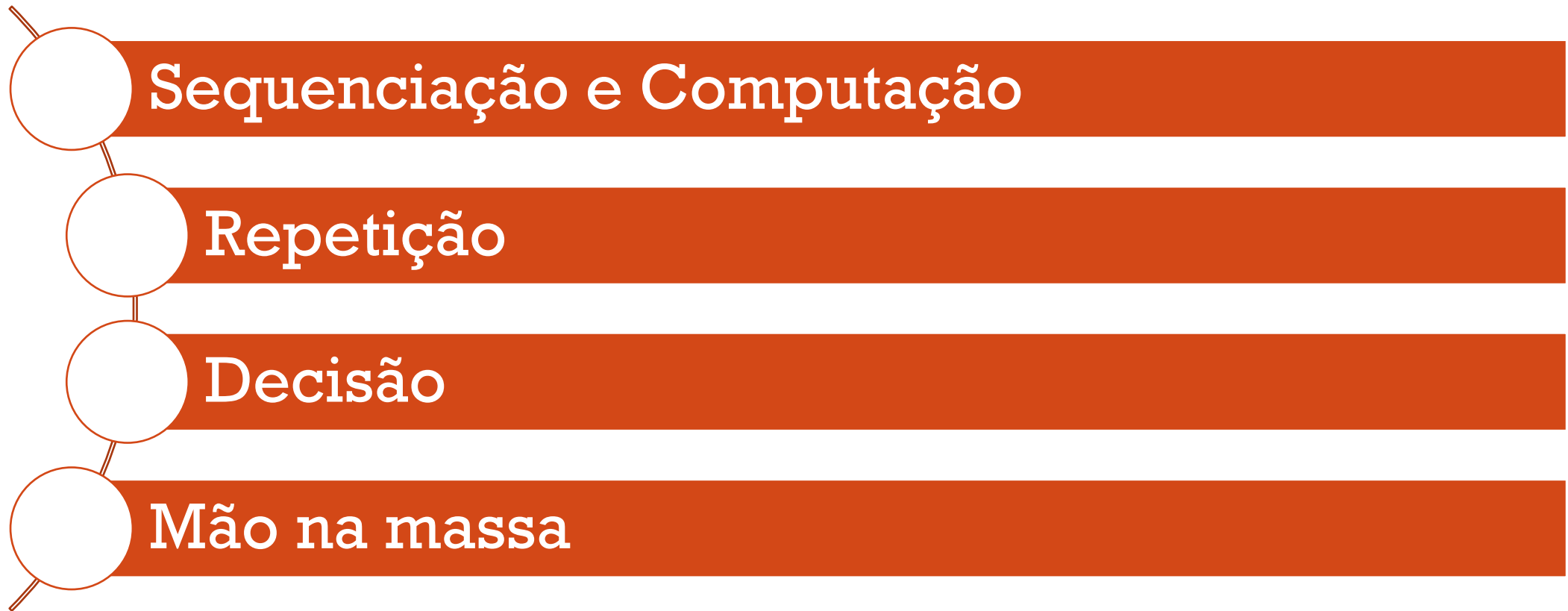


PYTHON BÁSICO

Prof. Peter Jandl Junior

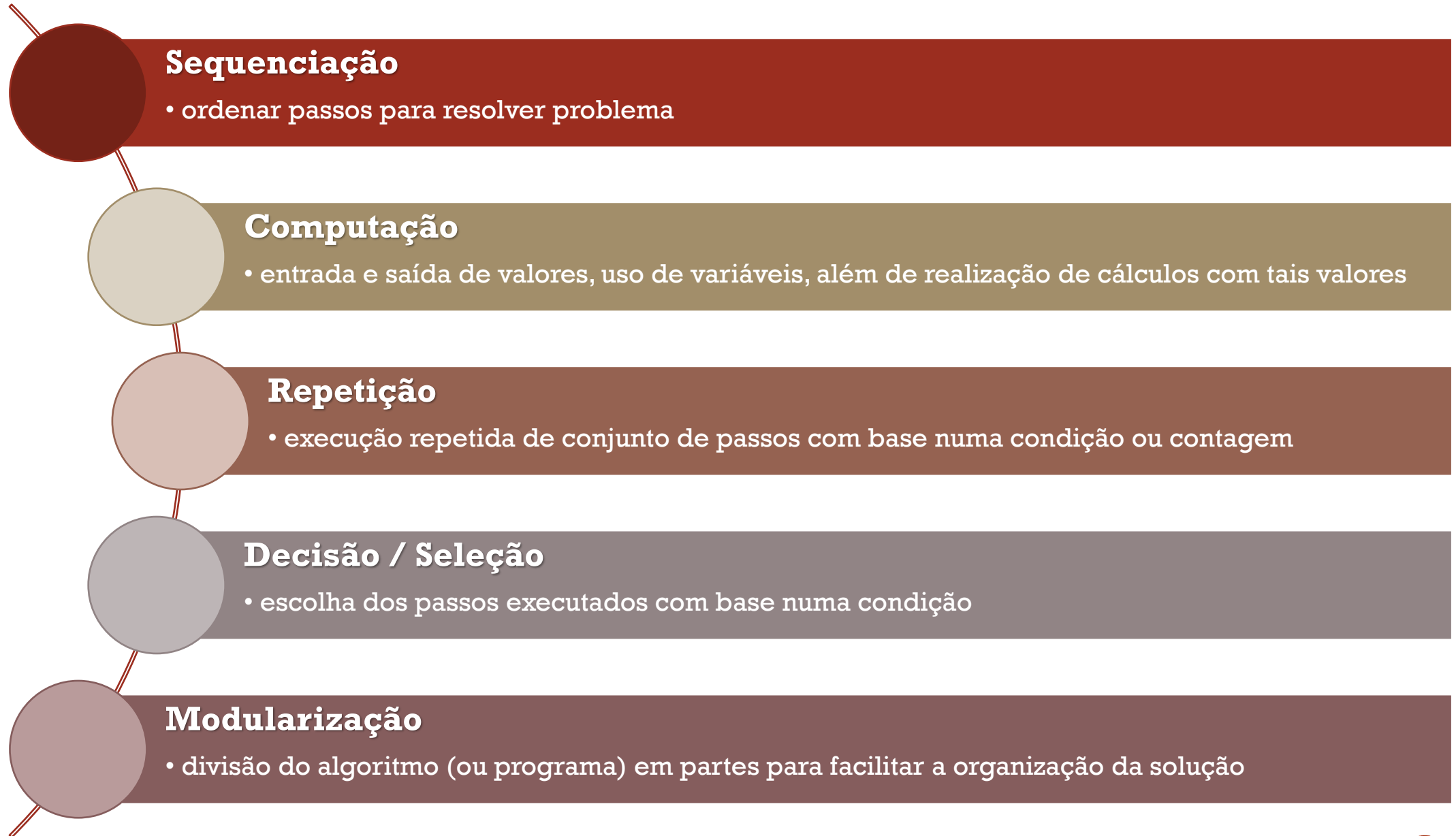
2

DIA 2 [ROTEIRO]



Not so Long ago
In a galaxy that isn't very far away....

HABILIDADES PARA CONSTRUÇÃO DE ALGORITMOS E PROGRAMAS





5



SEQUENCIACÃO | COMPUTAÇÃO

(C) 1999-2020, Jandl.

22/10/2020

PYTHON::MODO INTERATIVO

Console Python.

Modo interativo permite executar comandos um a um, no que se denomina: REPL (Read-Evaluate-Print-Loop)

Tecla Windows, digite:
Python ENTER.

Abra um promp de comandos:
Tecla Windows, digite:
Prompt Enter.
Digite:
python ENTER

```
Command Prompt - python
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pjand>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50)
[MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> print('Oi')
Oi
>>> print("Oficina Python")
Oficina Python
>>> print("Oficina Python\n")
Oficina Python

>>> print("Oficina\nPython")
Oficina
Python
>>> print(2020)
2020
>>> print(7.654321)
7.654321
>>> print('Ano',2020)
Ano 2020
>>> _
```

```
Python 3.8 (64-bit)
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50)
Type "help", "copyright", "credits" or "license" for more
>>> print('Oi')
Oi
>>> print("Oficina Python")
Oficina Python
>>> print("Oficina Python\n")
Oficina Python

>>> print("Oficina\nPython")
Oficina
Python
>>> print(2020)
2020
>>> print(7.654321)
7.654321
>>> print('Ano',2020)
Ano 2020
>>>
```

Para sair em ambos:
exit() ENTER
ou apenas CTRL+Z.

SAÍDA SIMPLES

- `print`
- Comando de saída simples.
- Escreve no console.
- Pode exibir:
 - Uma string (cadeia de caracteres/texto)
 - Um número
 - Uma variável (seu conteúdo)
 - Uma combinação de variáveis diversas e literais (strings e números)

- `print('Oi!')`
- `print("Oficina Python")`
- `print("Oficina Python\nDia 2")`
- `print("Mais\num dia\nno paraíso!")`
- `print(2020)`
- `print(7.654321)`
- `print('Ano', 2020)`

Toda string é delimitada por aspas simples ou aspas duplas (sem misturar)!

DEFINIÇÃO DE VARIÁVEIS

- Sintaxe de definição
 - nome = valorInicial
- Exemplos:
 - i = 0
 - valor = 0.23
 - nome = 'Peter Jandl Jr'
 - oficina = "Python"
 - legal = True

Uma variável é um símbolo ao qual associamos um valor.

Dar valor inicial é o chamamos de inicialização.

Nome de variável (identificador):

- Começa com letra,
- Pode usar números;
- Não pode conter espaços, símbolos, exceto `_` (sublinhado)

Você pode conhecer o tipo de uma variável com **type()**.

TIPOS PRÉ-DEFINIDOS (BUILT-INS)

- **int** → inteiro
→ -2, 0, 9, 28, 2020
- **float** → reais ou decimais
→ -93, -0.757, 1.23, 2938
- **bool** → lógico ou booleano
→ False, True
- **str** → string ou texto
'Exemplo', "Python",
"Peter Jandl"
- **complex** → complexos
→ -2.3 + 3.7j

Separador decimal é o ponto

Iniciados com maiúsculas

Delimitado por aspas simples ou duplas

O j simboliza a parte imaginária



OPERADORES

Aritméticos

+ - * /
// %
**


Relacionais

< <=
> >=
== !=

Lógicos

and is
or not
in

PYTHON: MODO INTERATIVO



```
Command Prompt - python
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pjand>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019,
Type "help", "copyright", "credits" or "license()"
>>> 1 + 2
3
>>> x = 1 + 2 * 3
>>> print(x)
7
>>> type(x)
<class 'int'>
>>> y = (1 + 2) * 3
>>> print("x=", x, "y=", y)
x= 7 y= 9
>>> print(x/y)
0.7777777777777778
>>> print(x//y)
0
>>> print(x%y)
7
>>> z = (x * 2) / (y - 3)
>>> print(z)
2.3333333333333335
>>> ■
```

- $1 + 2$
- $x = 1 + 2 * 3$
- `print(x)`
- `type(x)`
- $y = (1 + 2) * 3$
- `print("x=", x, "y=", y)`
- `print(x / y)`
- `print(x // y)`
- `print(x % y)`
- $z = (x * 2) / (y - 3)$
- `print(z)`

Você pode conhecer o tipo de uma variável com **type()**.

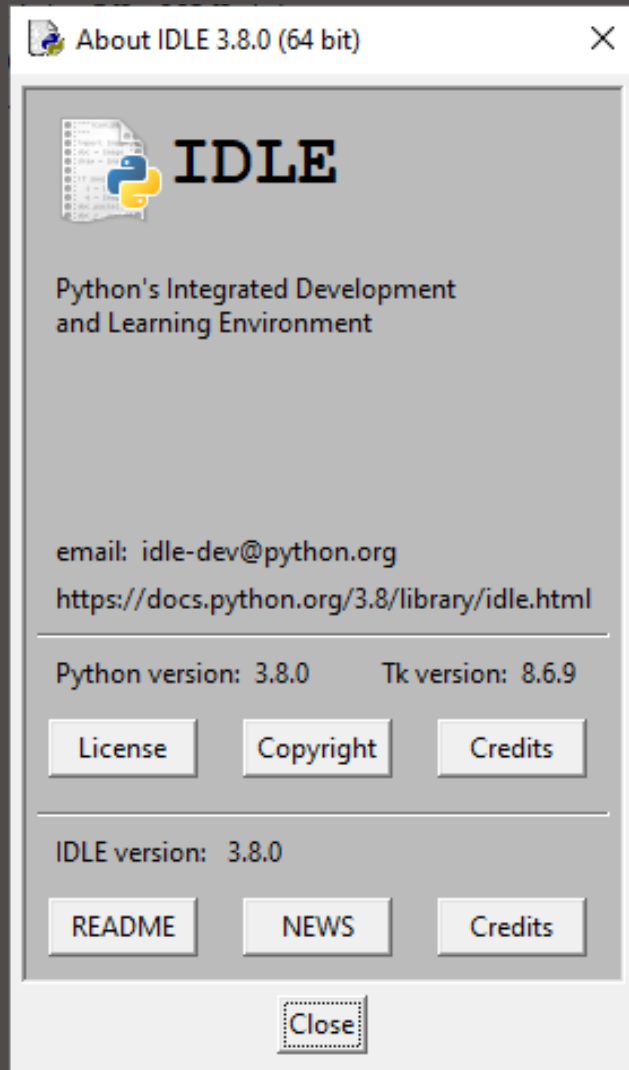
Teste com o console Python!

10/22/2020

PYTHON IDLE

Integrated Development and Learning Environment

- Ambiente integrado padrão das instalações de Python 3 (Python Shell).
- Oferece:
 - Um console
 - Um editor de programa
 - Browser (caminho, módulos)
 - Depurador



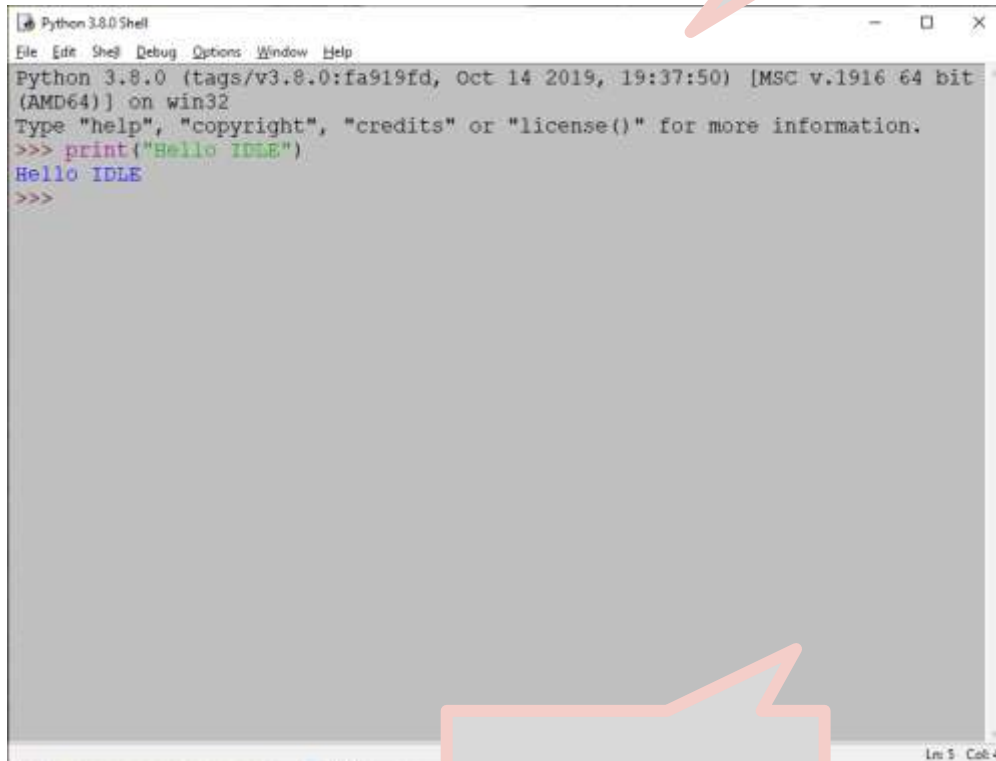
PYTHON IDLE

1

Tecla Windows.
Digite IDLE, Enter.

2

File | New File
Ctrl + N

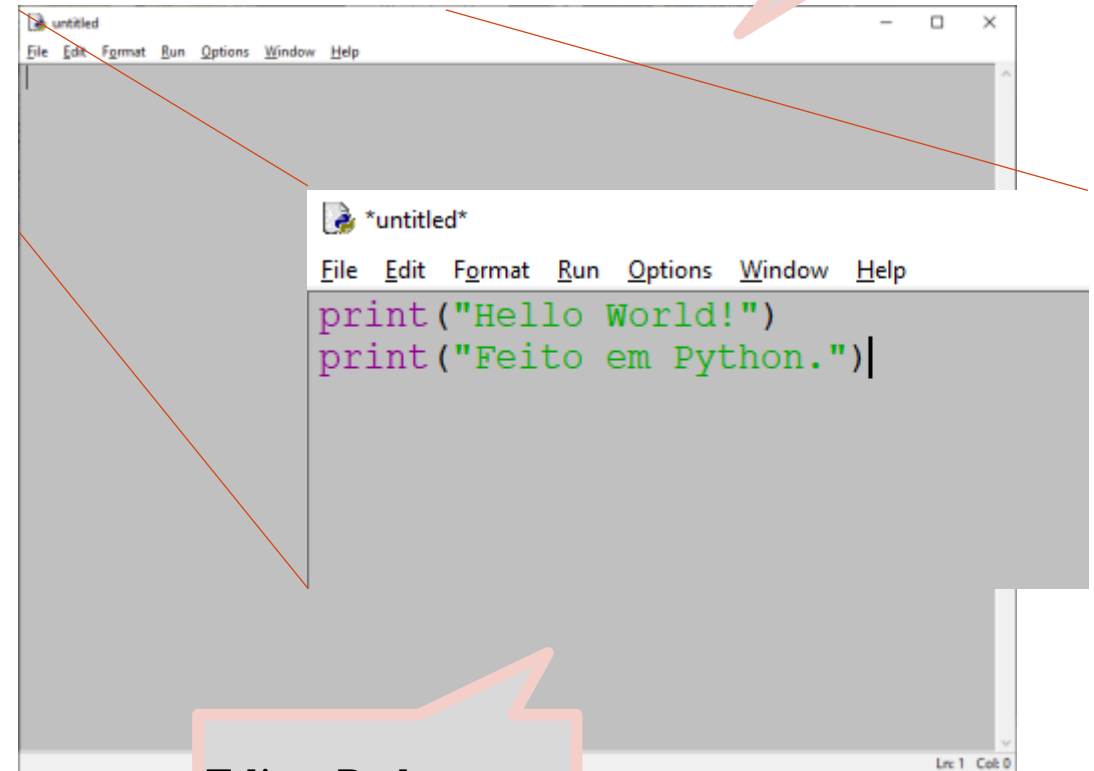


The screenshot shows the Python 3.8.0 Shell window. The title bar reads "Python 3.8.0 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The console text is as follows:

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> print("Hello IDLE")  
Hello IDLE  
>>>
```

The status bar at the bottom right indicates "Ln: 5 Col: 4".

Console Python



The screenshot shows the Python IDLE editor window with a new file open. The title bar reads "untitled". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". A smaller window titled "*untitled*" is overlaid on top, showing the following code:

```
print("Hello World!")  
print("Feito em Python.")|
```

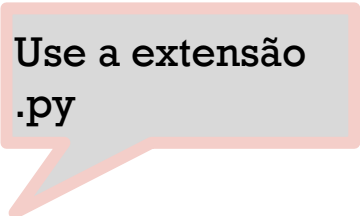
The status bar at the bottom right of the main window indicates "Ln: 1 Col: 0".

Editor Python

PYTHON IDLE [PROGRAMAÇÃO]

Novos programas

- File | New File (Ctrl + N)
Para abrir um novo editor
- Edite seu código.
Observe o destaque da sintaxe para:
 - Comandos
 - Strings e Números
 - Pontuação
- File | Save (Ctrl + S)
Para salvar o código num arquivo.
- Run | Run module (F5)
Para executar o programa no console.



Use a extensão
.py

Programas existentes

- File | Open ... (Ctrl + O)
Para abrir um arquivo existente.
- File | Save as ... (Ctrl + Shift + S)
Para salvar com novo nome.
- File | Close (Alt+ F4)
Fecha a janela atual.
- File | Quit (Ctrl + Q)
Encerra o IDLE.

COMENTÁRIOS

- # Comentário de uma linha
- # Comentário de múltiplas linhas simulado, pois
- # Python só tem um tipo de comentário!
- """
- Strings multilinhas não atribuídas funcionam como
- comentários de documentação.
- """

10/22/2020

ENTRADA DE DADOS

- É feita com a função embutida `input`, que é capaz de:
 - Exibir uma mensagem no prompt de comandos (ou shell)
 - Ler o texto (string) que foi digitado pelo usuário
- Exemplo:
 - `planeta = input('Digite seu planeta: ')`
 - `cidade = input('Digite sua cidade: ')`



```

imc.py - C:\Users\pjand\Desktop\Oficina Python\código-fonte\Dia_2\imc.py (3.8.0)
File Edit Format Run Options Window Help

# Cálculo do IMC (índice de massa corpórea)
print('IMC::índice de massa corpórea')

# Entrada de dados
altura = float(input('Digite sua altura em metros (m): '))
massa = float(input('Digite sua massa em quilogramas (kg): '))

# Processamento
imc = massa / (altura**2)

#Saída de dados
print('IMC =', imc)
print('IMC = {:.2f}'.format(imc))
print('IMC({:.2f},{:.1f}) = {:.2f}'.format(altura, massa, imc))
#
#
#
#

```

Impressão formatada

Exibe a variável imc com formato real com duas casas decimais `{:.2f}`

Impressão formatada

Exibe as variáveis altura, massa e imc com formato real com uma `{:.1f}` ou duas casas decimais `{:.2f}`

ENTRADA DE DADOS

- Texto, uso direto de:
 - Input.
- `nome = input("Nome? ")`
- Números, uso combinado de:
 - input e
 - funções de conversão:
- `inteiro = int(input("Inteiro? "))`
- `real = float(input("Real? "))`

Resultado no IDLE

```
>>>
=== RESTART: C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/imc.py ===
IMC::índice de massa corpórea
Digite sua altura em metros (m): 1.69
Digite sua massa em quilogramas (kg): 85
IMC = 29.76086271489094
IMC = 29.76
>>> |
```



Teste no IDLE!!!

Resultado no Prompt de Comandos
Navegue até o diretório adequado.
Execute:

> **python imc.py**

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pjand\Desktop\Oficina Python\código-fonte\Dia_2>python imc.py
IMC::índice de massa corpórea
Digite sua altura em metros (m): 1.78
Digite sua massa em quilogramas (kg): 74.5
IMC = 23.513445272061606
IMC = 23.51

C:\Users\pjand\Desktop\Oficina Python\código-fonte\Dia_2>_
```




19

PYTHON::REPETIÇÃO

(C) 1999-2020, Jandl.

22/10/2020

REPETIÇÃO

Condicional: while

- Permite a repetição de um comando ou de um conjunto de comandos.
- **Repetição condicional** é realizada conforme a avaliação de uma condição:
 - Se avaliada como **True**:
Realiza repetição do(s) comando(s)
 - Se avaliada como **False**
Finaliza o comando de repetição (interrompe a repetição).

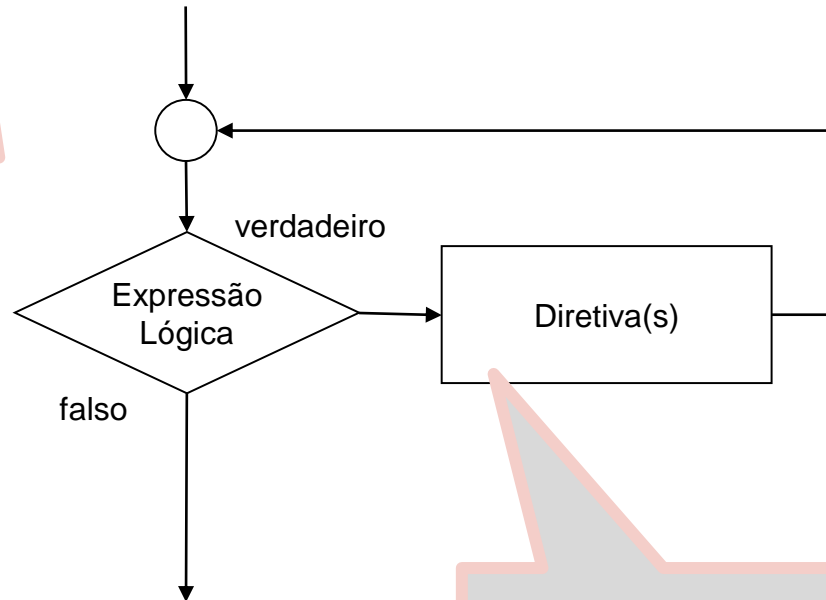
Automática: for

- Permite a repetição de um comando ou de um conjunto de comandos.
- **Repetição automática** aplica uma lista de valores a uma variável:
 - Variável pode ser usada no(s) comando(s) associado(s).
 - Se a lista possui um valor, os comandos são executados uma vez.
 - Se a lista possui dois valores, os comandos são executados uma vez.
- A repetição se encerra quando não existe mais valores a serem aplicados.

REPETIÇÃO CONDICIONAL

Testa-e-Faz

Testa primeiro, faz se condição **True**.



Comando = Diretiva

Observe o dois-pontos!

▪ Sintaxe:

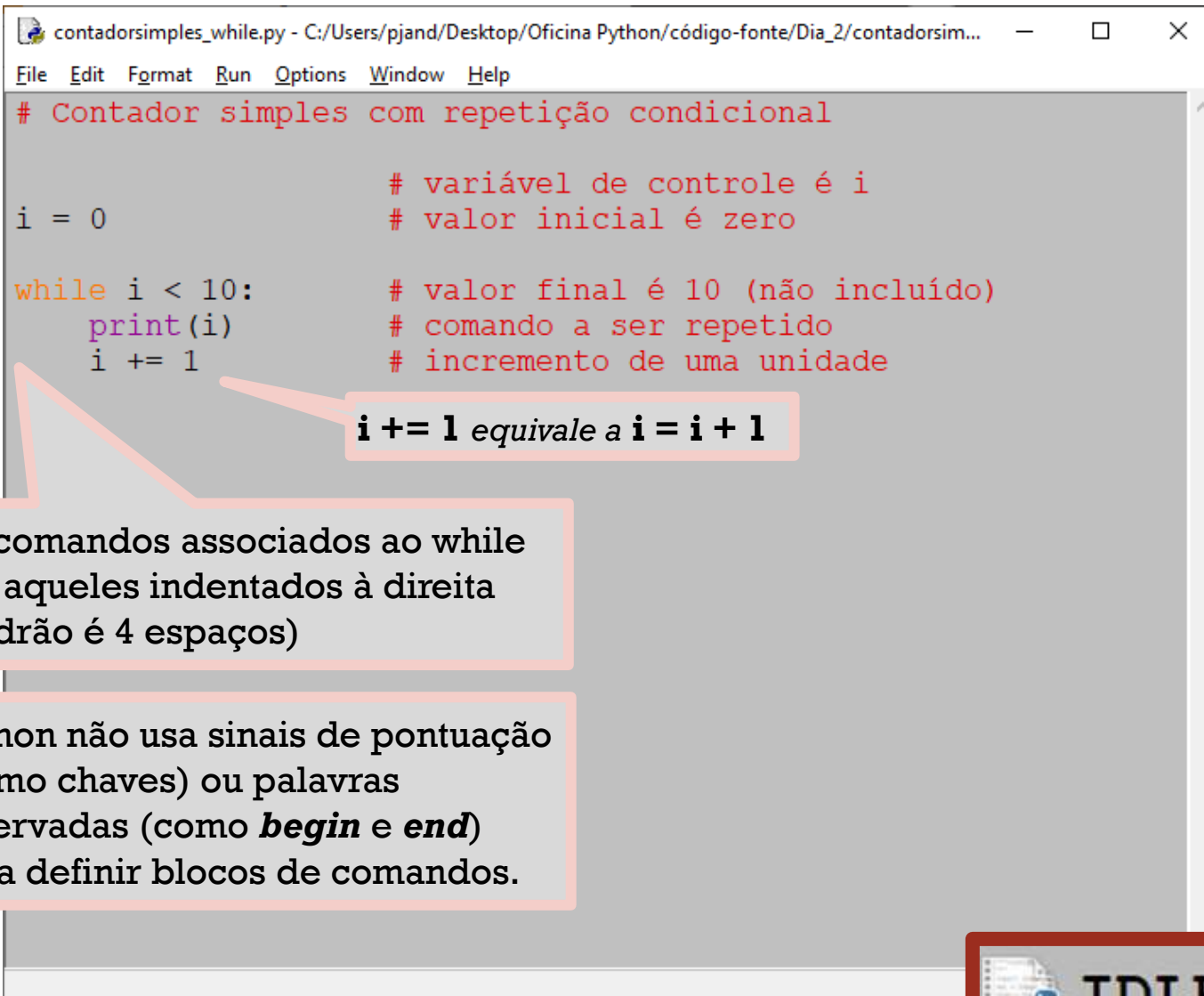
while condição :

comandol

:

comandoN

- A condição **deve** ser um expressão que resulte o tipo bool (lógico).
- Podem ser associados um ou mais comandos à diretiva **while**.
- Os comandos devem estar identados 04 (quatro) espaços a direita (**padrão Python**).



```
contadorsimples_while.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/contadorsim...
File Edit Format Run Options Window Help
# Contador simples com repetição condicional


i = 0                # variável de controle é i
                    # valor inicial é zero

while i < 10:        # valor final é 10 (não incluído)
    print(i)         # comando a ser repetido
    i += 1           # incremento de uma unidade

i += 1 equivale a i = i + 1
```

Os comandos associados ao while são aqueles indentados à direita (padrão é 4 espaços)

Python não usa sinais de pontuação (como chaves) ou palavras reservadas (como **begin** e **end**) para definir blocos de comandos.



CONTAGEM

- Toda contagem envolve:
 - Um valor **inicial**
 - Um valor **final**
 - Um **passo**:
 - Incremento se início < fim
 - Decremento se início > fim
 - Uma **variável** de controle, cujo valor evolui do início ao fim da contagem, passo a passo.
- Cada valor assumido pela variável de controle define uma **iteração** da estrutura de repetição.



```
contadorregressivo_while.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/contadorr...
File Edit Format Run Options Window Help
# Contador regressivo com repetição condicional

i = 10                # variável de controle é i
                     # valor inicial é dez

while i >= 0:         # valor final é 0 (incluído)
    print(i)          # comando a ser repetido
    i -= 1            # decremento de uma unidade
```

$i -= 1$ equivale a $i = i - 1$


Os comandos associados ao while são aqueles indentados à direita (padrão é 4 espaços)

Python não usa sinais de pontuação (como chaves) ou palavras reservadas (como ***begin*** e ***end***) para definir blocos de comandos.



CONTADORES

- Requerem uma ***variável de controle*** para contagem.
- Contagem pode ser:
 - **Progressiva, crescente** ou ascendente;
 - **Regressiva, decrescente** ou descendente.
- O valor final pode ou não ser incluído na contagem:
 - É a condição da repetição que define a inclusão.



```
contadorgenerico_while.py - C:\Users\pjand\Desktop\Oficina Python\código-fonte\Dia_2\contadorgeneri...
File Edit Format Run Options Window Help
# Contador genérico com repetição condicional

# Entrada de dados
inicio = float(input('Valor real inicial da contagem: '))
final = float(input('Valor real final da contagem: '))
passo = float(input('Valor real do passo da contagem: '))

# variável de controle é i
i = inicio # valor inicial é inicio

# processamento de dados
print('início')
while i <= final: # fim da contagem (não incluído) é final
    print('|', i) # comando a ser repetido
    i += passo # incremento determinado por passo

# saída de dados
print('fim')

|
```

$i += \text{passo}$ equivale a $i = i + \text{passo}$

A repetição está generalizada com o uso das variáveis ***inicio***, ***final*** e ***passo***.

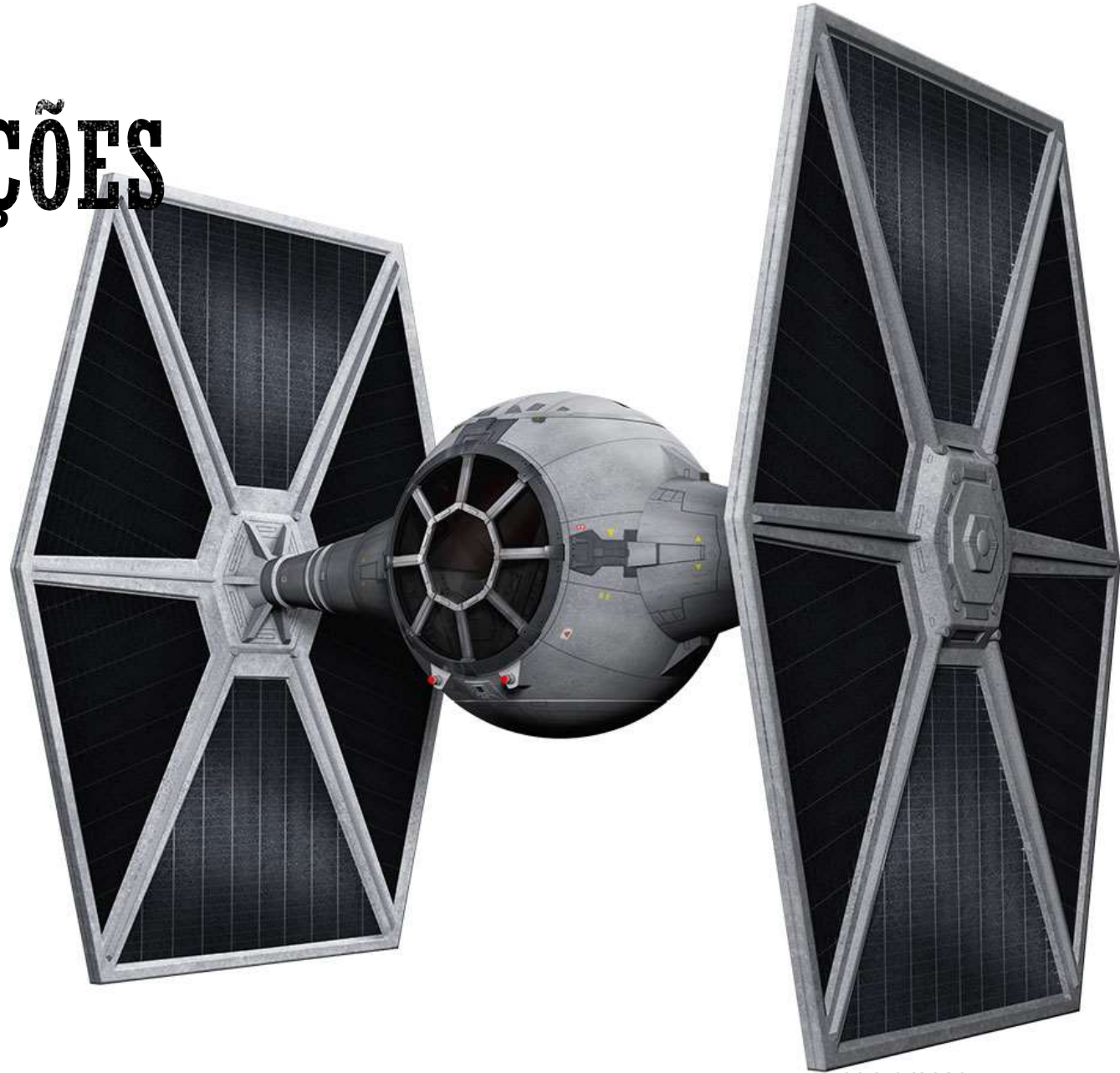


CONTADORES

- Além da variável de controle, use variáveis para armazenar:
 - Início,
 - Final
 - Passo
- *É uma boa prática de programação independente da linguagem utilizada!*

REPETIÇÃO::APLICAÇÕES

- O uso da repetição permite:
 - **Contar** a quantidade de repetições de uma ou mais ações desejadas;
 - **Agregar** valores: somatórios, produtórios, determinação de máximo, mínimo e média;
 - **Indexar** (percorrer toda uma estrutura de dados ou uma parte dela).




```
entradavalidada.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/entradavalidada.py (3.8.0)
File Edit Format Run Options Window Help
# Validação de entrada
# Quando é necessário ler um valor DENTRO de uma faixa de valores
MINIMO = 0.0 # valor mínimo da faixa (incluso)
MAXIMO = 10.0 # valor máximo da faixa (incluso)

# Entrada de dados
notal = float(input('Digite 1a nota [0.0, 10.0]: '))
# Validação: repete se notal FORA da faixa
while notal < MINIMO or notal > MAXIMO:
    # Informa usuário sobre seu erro
    print('Êita! Vamos tentar novamente!')
    # Repete leitura
    notal = float(input('Digite 1a nota [0.0, 10.0]: '))
# Comando executado após laço, então notal ESTÁ na faixa

nota2 = float(input('Digite 2a nota [0.0, 10.0]: '))
# Validação: repete se valor FORA da faixa
while nota2 < MINIMO or nota2 > MAXIMO:
    # Informa usuário sobre seu erro
    print('Êita! Vamos tentar novamente!')
    # Repete leitura
    nota2 = float(input('Digite 2a nota [0.0, 10.0]: '))
# Comando executado após laço, então nota2 ESTÁ na faixa

# Saída de dados
media = (notal + nota2) / 2
print('Notal: {:.1f} | Nota2: {:.1f} | Media {:.2f}'
      .format(notal, nota2, media))
```

Boa prática!

Laço teimoso: só sai se der certo

É incômodo esse código da entrada da segunda nota igual ao usado anteriormente.

Observe o uso de format!



VALIDAÇÃO

- Uso de repetição permite a validação de valores fornecidos pelo usuário.
- Validar significa garantir que o valor fornecido está dentro de uma faixa de valores, i.e., *entre de um mínimo e um máximo*.

S

C

R

Laço
contador

Boa prática!

Observe a acumulação e a contagem.

Observe o uso de format!

```
acumulacao.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/acumulacao.py (3.8.0)
File Edit Format Run Options Window Help
# Acumulação de valores de entrada
# Leitura de N valores a serem somados, com indicação da média
N = 10 # número de valores

# Entrada de dados
soma = 0 # variável para acumulação, recebe valor nêutro
contador = 0 # variável de controle da repetição
while contador < N:
    # Solicita valor
    valor = float(input('Digite valor real #{:d}: '.format(contador+1)))

    # Acumula valor
    soma += valor # o mesmo que: soma = soma + valor
    # Incrementa variável de controle
    contador += 1 # o mesmo que: contador = contador + 1

# Saída de dados
media = soma / N
print('Soma: {:.2f} | Media {:.2f}'.format(soma, media))
```



ACUMULAÇÃO

- Uso de repetição permite a acumular de valores fornecidos pelo usuário.
- A acumulação pode:
 - Receber um número **pré-determinado** de vezes (neste exemplo, valor de N);
 - Ser realizada até o atingimento de um objetivo, i.e., que o total acumulado iguale (se *aproxime ou supere*) uma **meta pré-estabelecida**.

REPETIÇÃO



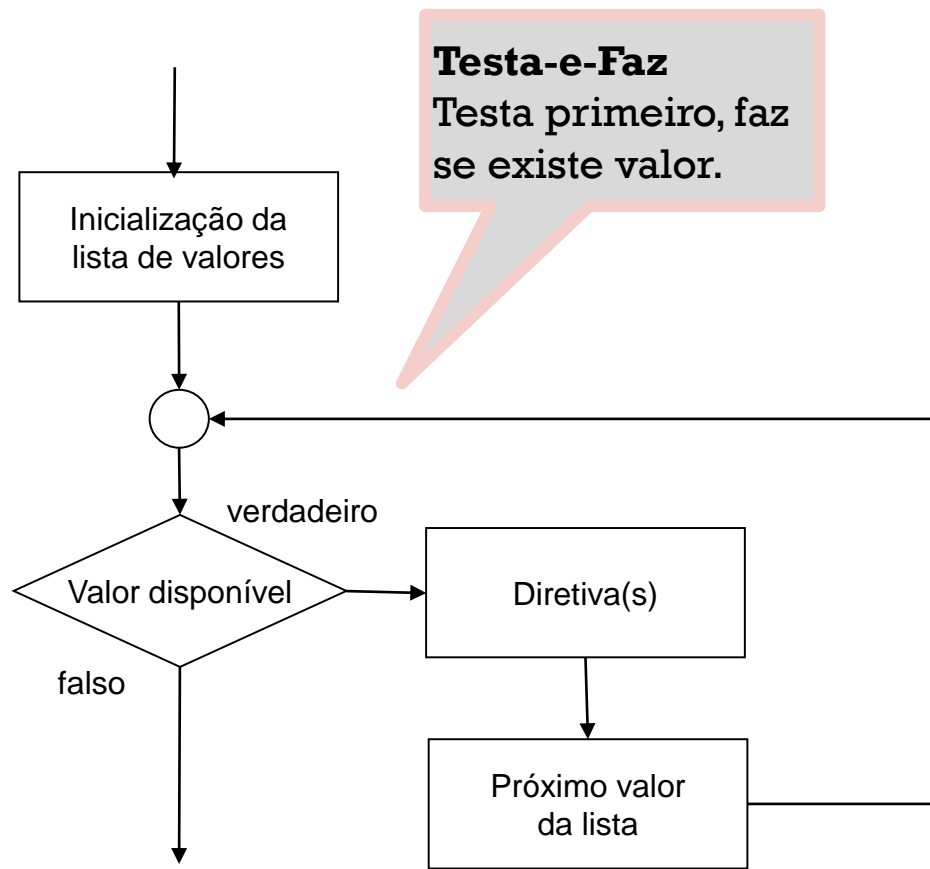
Condicional: while

- Permite a repetição de um comando ou de um conjunto de comandos.
- **Repetição condicional** é realizada conforme a avaliação de uma condição:
 - Se avaliada como **True**:
Realiza repetição do(s) comando(s)
 - Se avaliada como **False**
Finaliza o comando de repetição (interrompe a repetição).

Automática: for

- Permite a repetição de um comando ou de um conjunto de comandos.
- **Repetição automática** aplica uma lista de valores a uma variável:
 - Variável pode ser usada no(s) comando(s) associado(s).
 - Se a lista possui um valor, os comandos são executados uma vez.
 - Se a lista possui dois valores, os comandos são executados uma vez.
- A repetição se encerra quando não existe mais valores a serem aplicados.

REPETIÇÃO AUTOMÁTICA



Definição implícita da variável

Lista de valores (de qualquer tipo)

É como um **foreach!**

- Sintaxe:

for variável **in** listaValores :

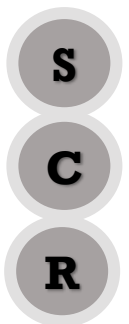
comando1

:

comandoN

Observe o dois-pontos!

- A listaValores **deve** resultar/conter zero ou mais valores de qualquer tipo.
- Podem ser associados um ou mais comandos à diretiva **for**.
- Os comandos devem estar identados 04 (quatro) espaços a direita (**padrão Python**).



```
contadorsimples_for.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/contadorsimples_for...
File Edit Format Run Options Window Help
# Contador simples com repetição automática
# variável de controle é definida no próprio
for i in range(0, 10):
    print(i)          # comando a ser repetido
                     # incremento implícito de uma unidade
```

A função range() requer inteiros

Os comandos associados ao **for** são aqueles indentados à direita (padrão é 4 espaços)

Python não usa sinais de pontuação (como chaves) ou palavras reservadas (como **begin** e **end**) para definir blocos de comandos.

```
>>>
= RESTART: C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/contadorsimples_for.py
0
1
2
3
4
5
6
7
8
9
>>>
```



CONTAGEM

- Toda contagem envolve:
 - Um valor **inicial**
 - Um valor **final**
 - Um **passo**:
 - Incremento se início < fim
 - Decremento se início > fim
 - Uma **variável** de controle, cujo valor evolui do início ao fim da contagem, passo a passo.
- A função **range(ini,fim)** gera uma lista com valor inicial **ini**, até o valor final **fim** (não-incluso), com passo 1.

S

C

R

contadorregressivo_for.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/contadorregressi...

File Edit Format Run Options Window Help

Contador regressivo com repetição automática

início = 10 # valor inicial é dez

final = 0 # valor final é zero

passo = -2 # valor do passo é negativo (contagem regressiva)

função range parametrizada com as variáveis acima

for i in range(início, final, passo):

print(i) # comando a ser repetido

função range parametrizada com as variáveis acima

final-1 inclui o valor final (numa contagem regressiva)

for i in range(início, final-1, passo):

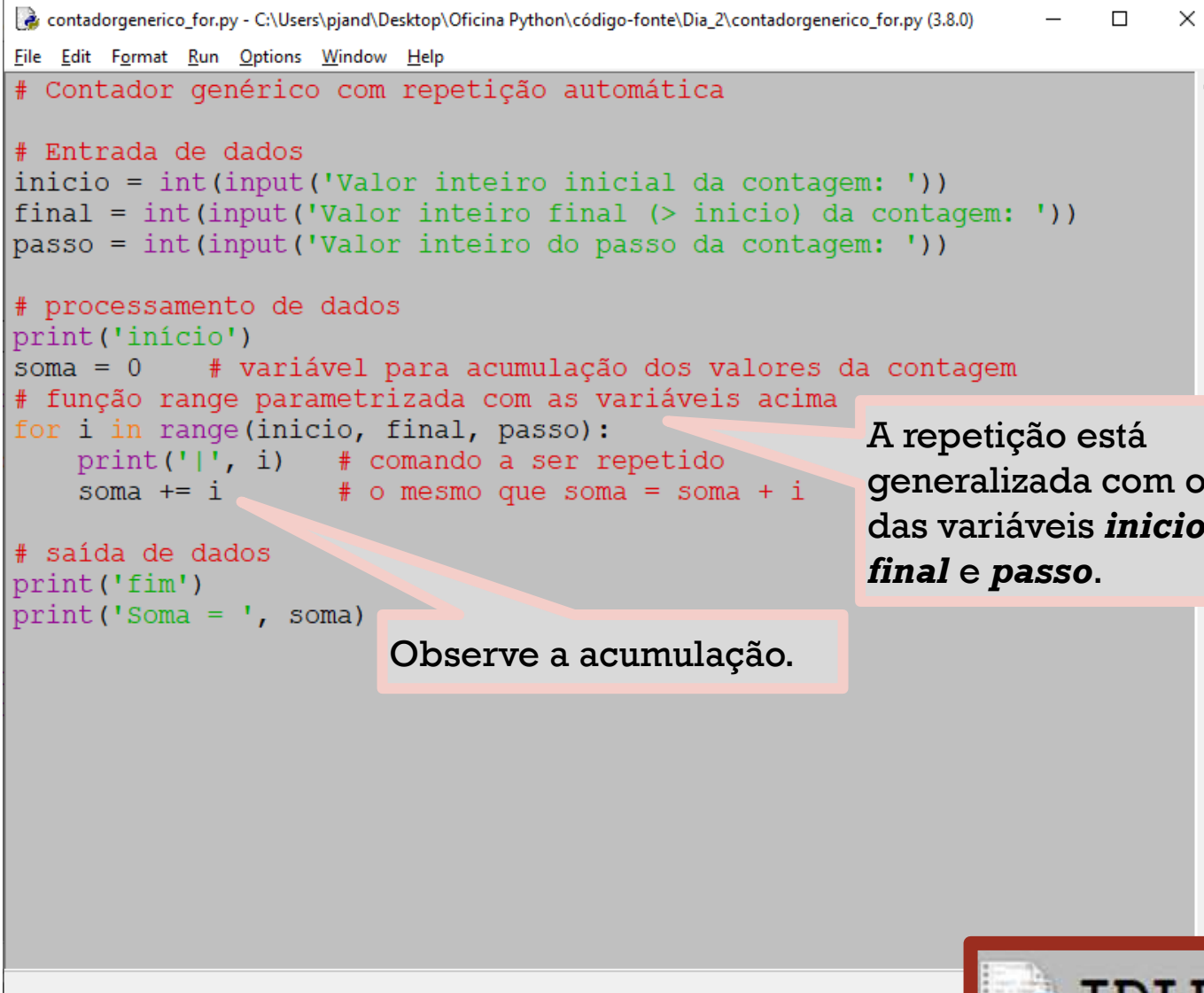
print(i) # comando a ser repetido

A função **range(ini, fim, passo)**
é fácil de usar, mas requer atenção
por conta do **valor final**
efetivamente desejado!



CONTADORES

- Contagem pode ser:
 - **Progressiva, crescente** ou ascendente;
 - `range(0, 10) → [0, 9]`
 - `range(1, 11, 2) → [1, 3, 5, 7, 9]`
 - **Regressiva, decrescente** ou descendente.
 - `range(100, -1, -10) → [100, 90..0]`
 - `range(20, 1, -5) → [20, 15, 10, 5]`



```
contadorgenerico_for.py - C:\Users\pjand\Desktop\Oficina Python\código-fonte\Dia_2\contadorgenerico_for.py (3.8.0)
File Edit Format Run Options Window Help
# Contador genérico com repetição automática

# Entrada de dados
inicio = int(input('Valor inteiro inicial da contagem: '))
final = int(input('Valor inteiro final (> inicio) da contagem: '))
passo = int(input('Valor inteiro do passo da contagem: '))

# processamento de dados
print('início')
soma = 0      # variável para acumulação dos valores da contagem
# função range parametrizada com as variáveis acima
for i in range(inicio, final, passo):
    print('|', i)    # comando a ser repetido
    soma += i        # o mesmo que soma = soma + i

# saída de dados
print('fim')
print('Soma = ', soma)
```

A repetição está generalizada com o uso das variáveis **inicio**, **final** e **passo**.

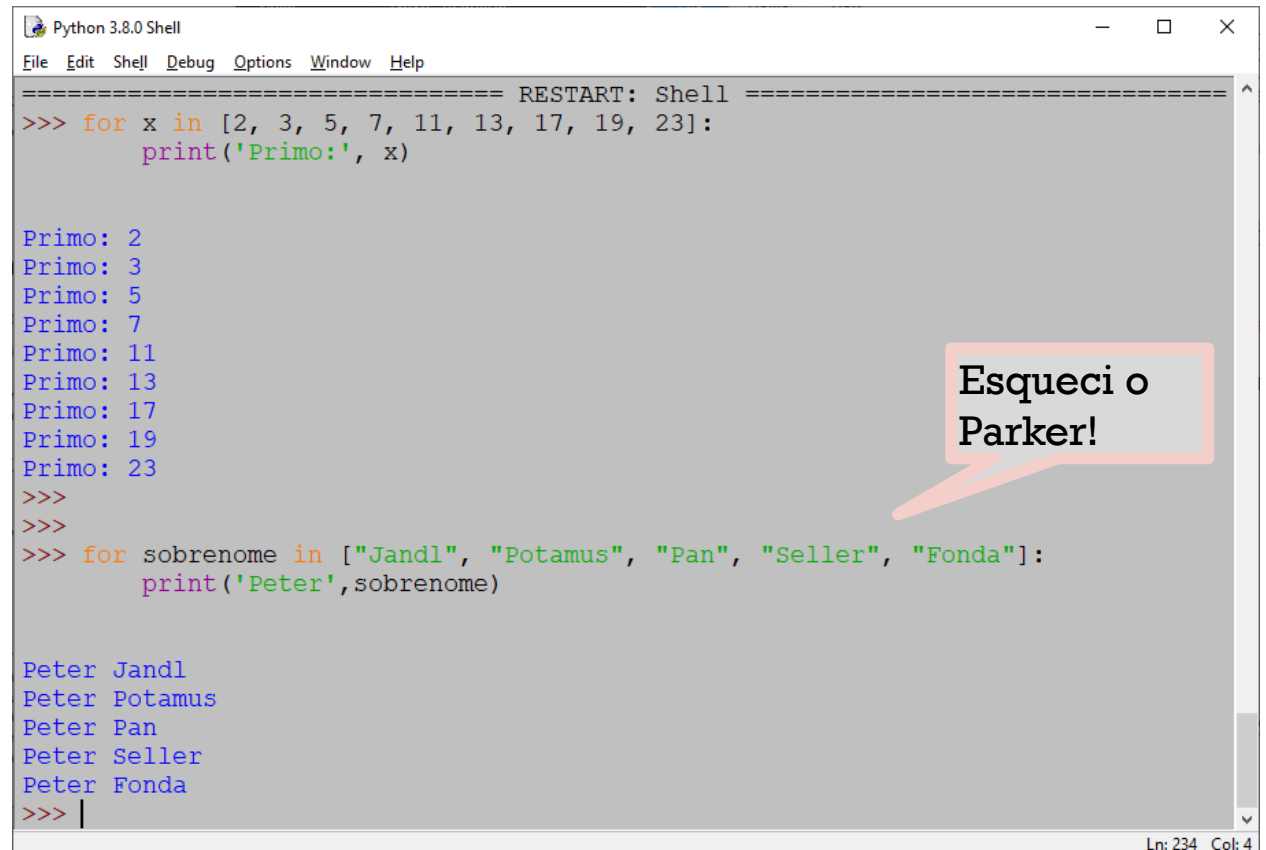
Observe a acumulação.

CONTADORES

- Contar e acumular requer variáveis de controle separadas
- Além das variáveis de controle, use outras para armazenar:
 - Início,
 - Final e
 - Passo.
- *É uma boa prática de programação independente da linguagem utilizada!*

REPETIÇÃO AUTOMÁTICA COM LISTAS PRÉ-DEFINIDAS

- Listas:
 - de inteiros,
 - de reais,
 - de string etc.
- podem ser definidas diretamente no código para uso num for.



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: Shell =====
>>> for x in [2, 3, 5, 7, 11, 13, 17, 19, 23]:
    print('Primo:', x)

Primo: 2
Primo: 3
Primo: 5
Primo: 7
Primo: 11
Primo: 13
Primo: 17
Primo: 19
Primo: 23
>>>
>>>
>>> for sobrenome in ["Jandl", "Potamus", "Pan", "Seller", "Fonda"]:
    print('Peter', sobrenome)

Peter Jandl
Peter Potamus
Peter Pan
Peter Seller
Peter Fonda
>>> |
```

Esqueci o Parker!

Ln: 234 Col: 4

SEQUENCIÇÃO

- Uso de repetição permite criar sequências de valores a partir de relações matemáticas requeridas pelos problemas.
- A sequenciação envolve:
 - Calcular (ou acumular) valores;
 - Determinar os termos da sequência baseados em valores anteriores.

```
fibonacci.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/fibonacci.py (3.8.0)
File Edit Format Run Options Window Help

# Fibonacci
# Sequência (série) matemática dada por  $F(n) = F(n-1) + F(n-2)$ 
# que significa: cada elemento é a soma dos dois valores anteriores
# Assim se definem os dois primeiros elementos como:  $F(0) = 0$  e  $F(1) = 1$ 
# Assim: 0 1 1 2 3 5 8 13 21 34 55 89

n = int(input('Digite o número do elemento: '))

v_2 = 0      # valor inicial do 2o anterior
v_1 = 1      # valor inicial do 1o anterior

print("F(0) = 0\nF(1) = 1")

for i in range(2, n+1):
    v = v_1 + v_2    # o termo atual é a soma dos dois anteriores
    print('F({:d}) = {:d} + {:d} = {:d}'.format(i, v_1, v_2, v))

    v_2 = v_1        # o 2a anterior se torna o 1o anterior
    v_1 = v          # o 1a anterior se torna atual

print("-----")
print("F({:d}) = {:d}".format(n, v))
```


Acumulação

Troca de valores

Observe o uso de format!

Laço
contador



A detailed view of the interior of the Millennium Falcon's cockpit. The scene is filled with various control panels, dials, and switches, all illuminated with green and blue lights. In the center, there are two brown leather seats. The background shows a large, circular window with a starry space view. The overall atmosphere is that of a classic science fiction movie set.

Aaaããann
ââaaaa?

Ainda não
decidi
Chewie!

A detailed view of the International Space Station (ISS) in space, showing its complex structure and solar panels against a starry background.

36

PYTHON::DECISÃO

(C) 1999-2020, Jandl.

22/10/2020

DECISÃO

Decisão simples: if

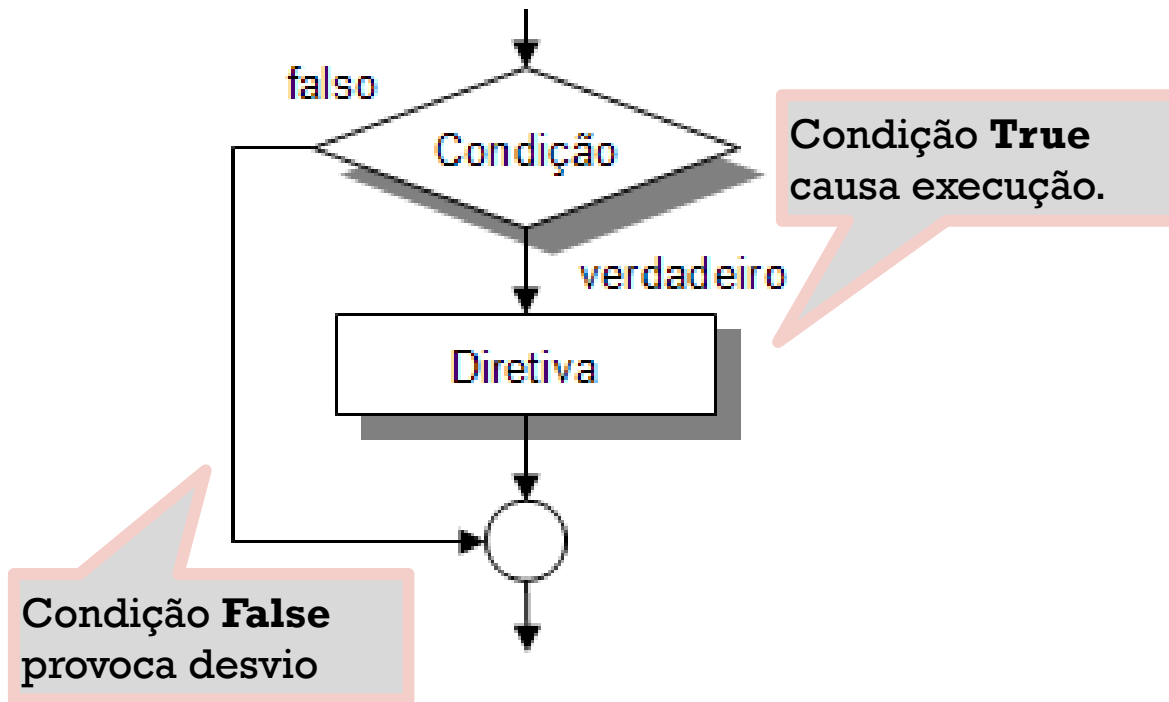
- Permite selecionar um comando ou um conjunto de comandos para execução.
- A **decisão simples** usa uma condição, cuja avaliação define:
 - Condição resulta **True**: Executa o(s) comando(s).
 - Condição resulta **False** **Não** executa o(s) comando(s).

A condição falsa provoca um desvio na execução!

Decisão completa: if/else

- Permite selecionar um comando (ou conjunto de comandos) **dentre** dois comandos (ou conjuntos de comandos).
- A **decisão completa** usa uma condição, cuja avaliação define a escolha:
 - Condição resulta True: Apenas o(s) comando(s) associado(s) ao próprio **if** é(são) executado(s).
 - Condição resulta False: Apenas o(s) comando(s) associado(s) à cláusula **else** é(são) executado(s).

DECISÃO SIMPLES



Observe o dois-pontos!

- Sintaxe:

if condição :

 # comandol
 :

 # comandoN

- A condição **deve** ser um expressão que resulte o tipo bool (lógico).
- Podem ser associados um ou mais comandos à diretiva **if**.
- Os comandos devem estar identados 04 (quatro) espaços a direita (**padrão Python**).


```
desconto_if.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/desconto_if.py (3.8.0)
File Edit Format Run Options Window Help

# Uso de decisão simples
precoIngresso = 22.00    # Preço do ingresso
descontoPadrao = 0.5    # Desconto padrão para estudantes (50%)

print('Compra de Ingressos')

quantidade = int(input('Quantos ingressos deseja comprar? '))
totalCompra = quantidade * precoIngresso

resp = input('Você é estudante [S|N]? ')

descontoCompra = 0      # Precisa definir variáveis para
totalFinal = totalCompra # garantir execução OK nos dois casos

if resp=='S' or resp=='s':
    # Aplica desconto SE É estudante
    descontoCompra = descontoPadrao*totalCompra
    totalFinal = totalCompra - descontoCompra

print('-----')
print('Quant ingressos =', quantidade)
print(' Preço ingresso = R$ {:.2f}'.format(precoIngresso))
print('Total da compra = R$ {:.2f}'.format(totalCompra))
print('      Desconto = R$ {:.2f}'.format(descontoCompra))
print('      Total Final = R$ {:.2f}'.format(totalFinal))
print('-----')
```

Variáveis para definir valores contantes

Decisão simples

Aplicação do desconto é condicional

Observe o uso de format!

DECISÃO

- Uso de decisão permite que um mesmo programa realizar operações diferentes.
- A **decisão simples** permite escolher (selecionar) se ações são executadas ou não.
- É muito importante prestar atenção na maneira com que a condição é expressa no programa.
- Aqui a aplicação do desconto só ocorre **se** a resposta é 's' ou 'S'.



DECISÃO



Decisão simples: if

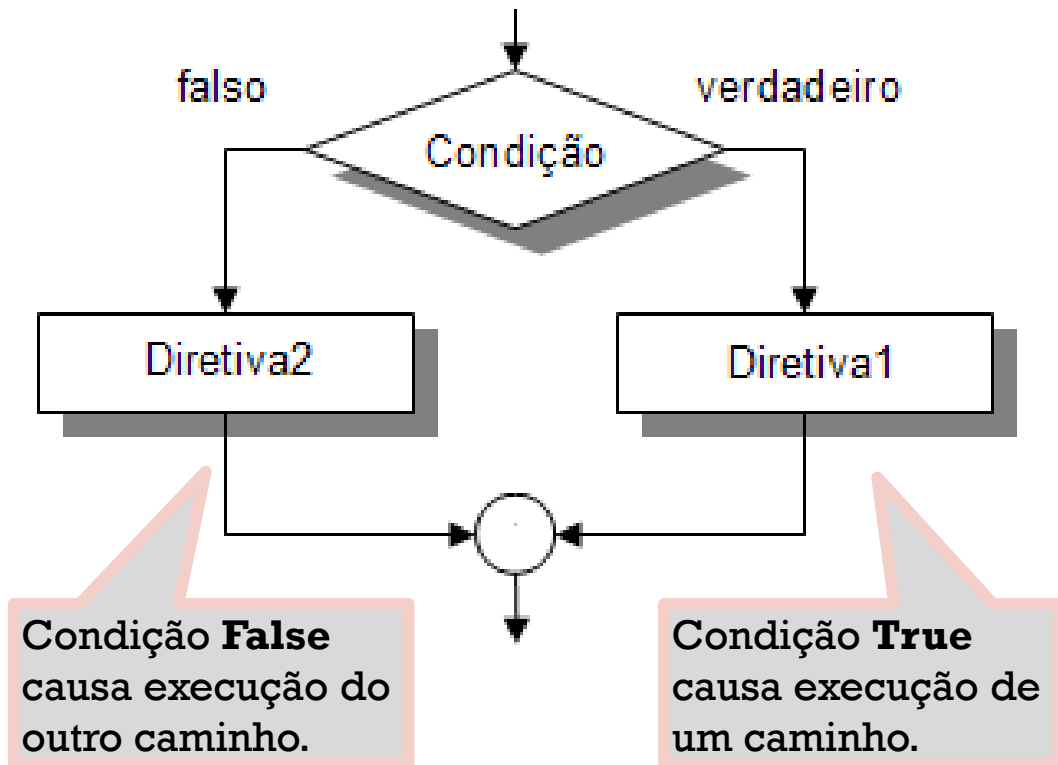
- Permite selecionar um comando ou um conjunto de comandos para execução.
- A **decisão simples** usa uma condição, cuja avaliação define:
 - Condição resulta **True**: Executa o(s) comando(s).
 - Condição resulta **False** **Não** executa o(s) comando(s).

É como uma bifurcação no caminho do programa, onde apenas um lado é executado!

Decisão completa: if/else

- Permite selecionar um comando (ou conjunto de comandos) dentre dois comandos (ou conjuntos de comandos).
- A **decisão completa** usa uma condição, cuja avaliação define a escolha:
 - Condição resulta **True**: Apenas o(s) comando(s) associado(s) ao próprio **if** é(são) executado(s).
 - Condição resulta **False**: Apenas o(s) comando(s) associado(s) à cláusula **else** é(são) executado(s).

DECISÃO COMPLETA



Observe o dois-pontos!

▪ Sintaxe:

if condição :

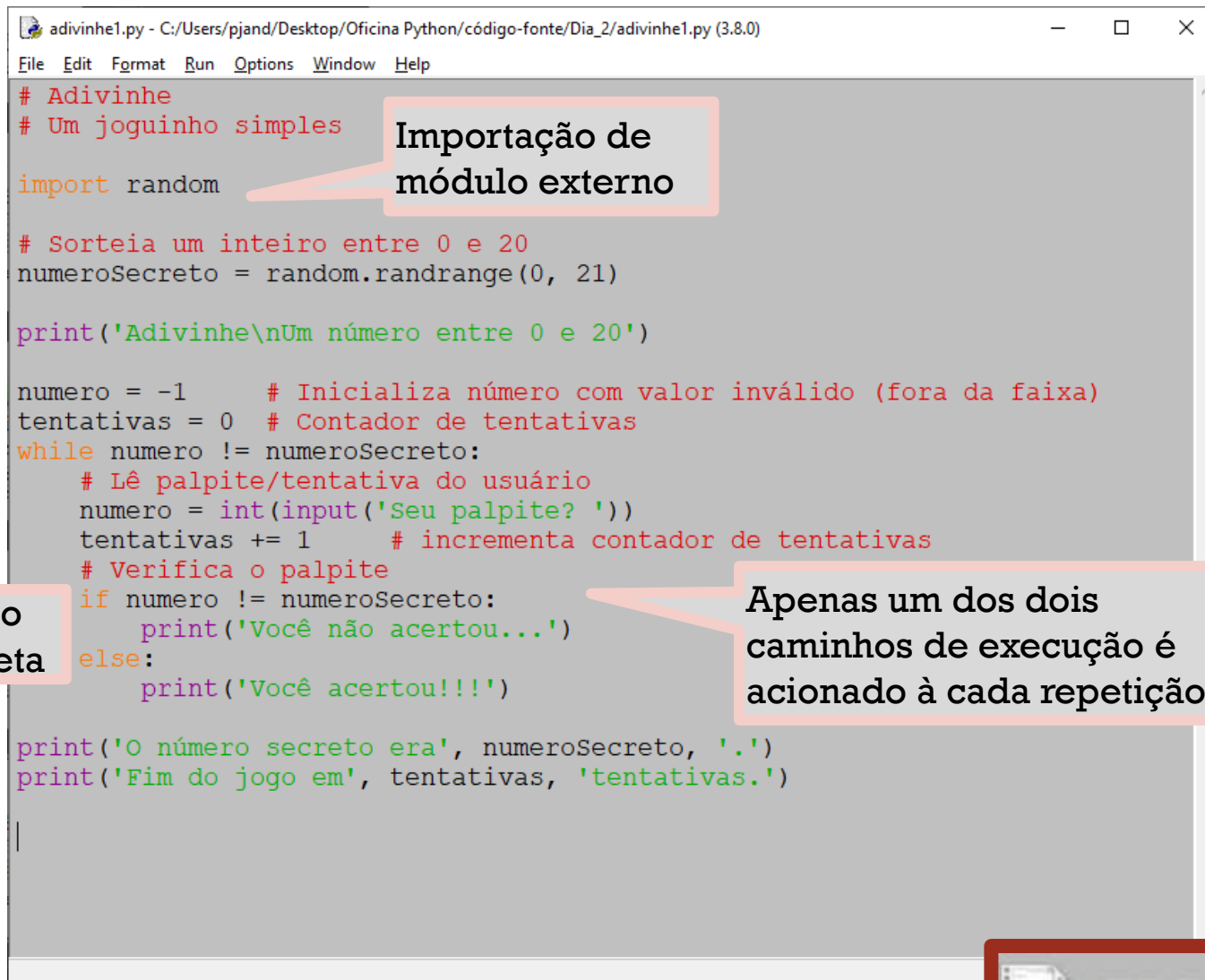
 # comando1(s)

else:

 # comando2(s)

- A condição **deve** ser um expressão que resulte o tipo bool (lógico).
- Podem ser associados um ou mais comandos à diretiva **if**.
- Os comandos devem estar identados 04 (quatro) espaços a direita (**padrão Python**).

Decisão
completa



```
adivinhe1.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/adivinhe1.py (3.8.0)
File Edit Format Run Options Window Help
# Advinhe
# Um joguinho simples
import random

# Sorteia um inteiro entre 0 e 20
numeroSecreto = random.randrange(0, 21)

print('Advinhe\nUm número entre 0 e 20')

numero = -1      # Inicializa número com valor inválido (fora da faixa)
tentativas = 0   # Contador de tentativas
while numero != numeroSecreto:
    # Lê palpite/tentativa do usuário
    numero = int(input('Seu palpite? '))
    tentativas += 1    # incrementa contador de tentativas
    # Verifica o palpite
    if numero != numeroSecreto:
        print('Você não acertou...')
    else:
        print('Você acertou!!!')

print('O número secreto era', numeroSecreto, '.')
print('Fim do jogo em', tentativas, 'tentativas.')
```



DECISÃO

- Uso de decisão permite que um mesmo programa realizar operações diferentes.
- A **decisão completa** permite escolher (selecionar) um conjunto de ações entre duas possibilidades.
- Preste muita atenção em como as condições de um programa são expressas.
- Aqui, a cada repetição do laço while, o palpite do usuário é avaliado para indicar se ele acertou ou se não acertou.

Decisão
em
cadeia

Importação de
módulo externo

Agora existem três
caminhos de execução!

```
adivinhe2.py - C:/Users/pjand/Desktop/Oficina Python/código-fonte/Dia_2/adivinhe2.py (3.8.0)
File Edit Format Run Options Window Help
# Adivinhe
# Um joguinho simples
import random

# Sorteia um inteiro entre 0 e 100
numeroSecreto = random.randrange(0, 101)

print('Adivinhe\nUm número entre 0 e 100')

numero = -1      # Inicializa número com valor inválido (fora da faixa)
tentativas = 0   # Contador de tentativas
while numero != numeroSecreto:
    # Lê palpite/tentativa do usuário
    numero = int(input('Seu palpite? '))
    tentativas += 1      # incrementa contador de tentativas
    # Verifica o palpite
    if numero < numeroSecreto:
        print('Baixo...')
    elif numero > numeroSecreto:
        print('Alto...')
    else:
        print('Você acertou!!!')

print('O número secreto era', numeroSecreto, '.')
print('Fim do jogo em', tentativas, 'tentativas.')
```



DECISÃO

- Uso de decisão permite que um mesmo programa realizar operações diferentes.
- **A decisão encadeada** permite combinar duas ou mais decisões completas de maneira a determinar 3, 4 ou mais possibilidades distintas.
- Preste muita atenção em como as condições de um programa são expressas.
- Aqui, a cada repetição do laço while, o palpite do usuário é avaliado para indicar se o valor dado é baixo, alto ou o correto.



O QUE JÁ SABEMOS FAZER?

- Computar: entrada, saída e cálculos;
- Repetir: condicional e automático;
- Decidir;
- Sequenciar, combinando tudo isso!



COMPUTAÇÃO

- Entrada de dados com **input()** e as funções de conversão **int()** e **float()**.
- Saída de dados **print()**.
- Definição de **variáveis** e realização de cálculos combinando **valores literais**, **variáveis** e **operadores**.



REPETIÇÃO

- Repetição condicional com **while**.
- Repetição automática com **for**.
- Uso da repetição para **contar**, **agregar** e **acumular**.



DECISÃO

- Seleção de comandos com uso de **if/else**.
- Encadeamento de seleção com **elif**.
- Permite validar valores e também flexibilizar o uso dos programas.

SEQUENCIAÇÃO

- Combinação de **computação** (entrada, saída e cálculos), **repetição** e **decisão** para resolver muitos tipos de problemas.



MÃOS NA MASSA

Outra lista!!!

- Resolver a Lista **DOIS**.
- Como pensar como um Cientista da Computação.
Projeto Panda | IME | USP.
<https://panda.ime.usp.br/pensepy/static/pensepy/index.html>
- Python e Orientação a Objetos
Curso Py-14 | Caelum.
<https://www.caelum.com.br/apostila/apostila-python-orientacao-a-objetos.pdf>