Algoritmos e Estrutura de Dados I

Aula 06

Estrutura Sequencial

Prof. Dr. Dilermando Piva Jr

1º Semestre - CDN

Zen of Python

Conjunto de “princípios orientadores” para melhor utilização da linguagem...
Criado por Tim Peters 1999

Zen do Python

1. Bonito é melhor que feio.
2. Explícito é melhor que implícito.
3. Simples é melhor que complexo.
4. Complexo é melhor que complicado.
5. Linear é melhor do que aninhado.
6. Esparso é melhor que denso.
7. Legibilidade conta.
8. Casos especiais não são especiais o bastante para quebrar as regras.
9. Ainda que praticidade vença a pureza.
10. Erros nunca devem passar silenciosamente.
11. A menos que sejam explicitamente silenciados.
12. Diante da ambiguidade, recuse a tentação de adivinhar.
13. Dever haver um — e preferencialmente apenas um — modo óbvio para fazer algo.
14. Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.
15. Agora é melhor que nunca.
16. Apesar de que nunca normalmente é melhor do que *exatamente* agora
17. Se a implementação é difícil de explicar, é uma má ideia
18. Se a implementação é fácil de explicar, pode ser uma boa ideia
19. Namespaces são uma grande ideia — vamos ter mais dessas!

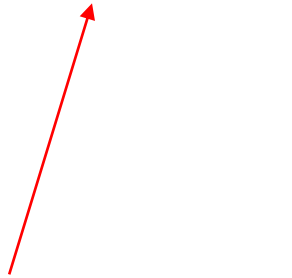
20º - Em aberto “para Guido preencher”

Zen do Python

1. Bonito é melhor que feio.
2. Explícito é melhor que implícito.
3. Simples é melhor que complexo.
4. Complexo é melhor que complicado.
5. Linear é melhor do que aninhado.
6. Esparso é melhor que denso.
7. Legibilidade conta.
8. Casos especiais não são especiais o bastante para quebrar as regras.
9. Ainda que praticidade vença a pureza.
10. Erros nunca devem passar silenciosamente.
11. A menos que sejam explicitamente silenciados.
12. Diante da ambiguidade, recuse a tentação de adivinhar.
13. Dever haver um — e preferencialmente apenas um — modo óbvio para fazer algo.
14. Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.
15. Agora é melhor que nunca.
16. Apesar de que nunca normalmente é melhor do que *exatamente* agora
17. Se a implementação é difícil de explicar, é uma má ideia
18. Se a implementação é fácil de explicar, pode ser uma boa ideia
19. Namespaces são uma grande ideia — vamos ter mais dessas!

Acessando o Zen no Python...

import this



easter egg

Prompt de Comando - python

```
C:\Users\Piva>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Zen do Python ---> PEP20

- Zen of Python foi incluído na posição 20 das PROPOSTAS OFICIAIS DE APRIMORAMENTO DO PYTHON PYTHON ENHANCEMENT PROPOSAL (PEP)
- PEP20

- Dessas muitas propostas... PEP-8
- O Estilo de codificação Python

PEP-8

PEP 8 – Guia de Estilo para Codificação Python

https://peps.python.org/pep-0008/

RESUMO

Resumo do PEP-008

- Use 4 espaços para indentação
- Nunca misture Tabs e Espaços
- Tamanho máximo de linha é de 79 caracteres
- `lower_case_with_underscore` p/ nomes de variáveis
- `CamelCase` para Classes
- Um `import` por linha

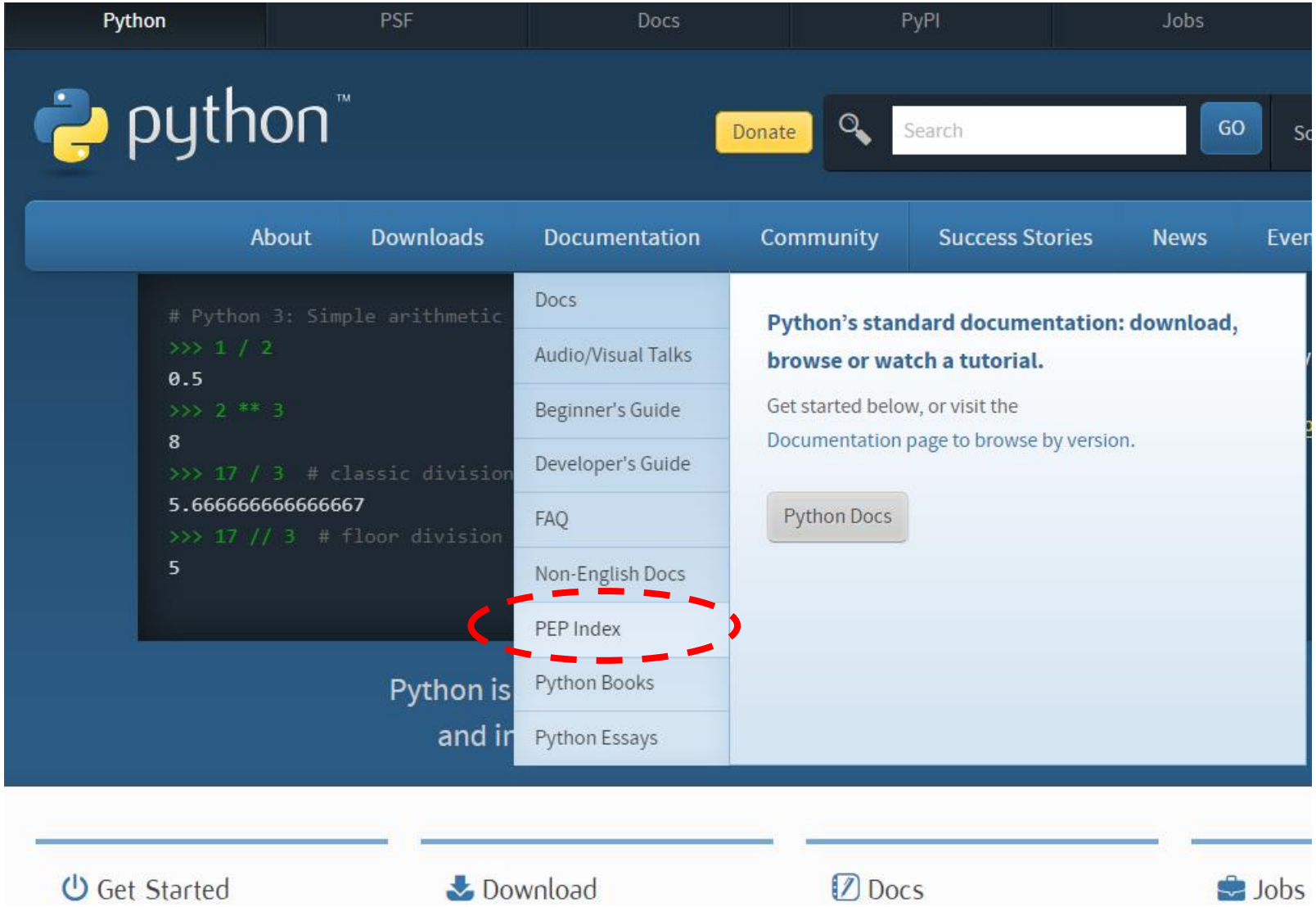
PEP

Python Enhancement Proposal

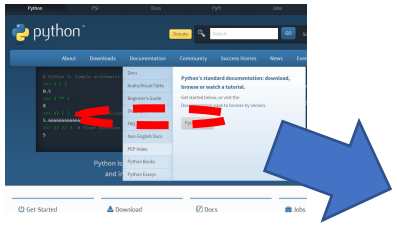
Propostas Oficiais de Aprimoramento do Python

https://peps.python.org

PEP Index


Python
PSF
Docs
PyPI
Jobs
python
Donate
Search
GO
About
Downloads
Documentation
Community
Success Stories
News
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3 # classic division
5.666666666666667
>>> 17 // 3 # floor division
5
Docs
Audio/Visual Talks
Beginner's Guide
Developer's Guide
FAQ
Non-English Docs
PEP Index
Python Books
Python Essays
Python's standard documentation: download, browse or watch a tutorial.
Get started below, or visit the Documentation page to browse by version.
Python Docs
Python is
and in
Get Started
Download
Docs
Jobs

PEP Index



Python Enhancement Proposals | Python » PEP Index » PEP 0

Contents

PEP 0 – Index of Python Enhancement Proposals (PEPs)

Author: python-dev <python-dev at python.org>
Status: Active
Type: Informational
Created: 13-Jul-2000

▶ **Table of Contents**

Introduction

This PEP contains the index of all Python Enhancement Proposals, known as PEPs. PEP numbers are assigned by the PEP editors, and once assigned are never changed. The version control history of the PEP texts represent their historical record. The PEPs are indexed by topic for specialist subjects.

Index by Category

Meta-PEPs (PEPs about PEPs or Processes)

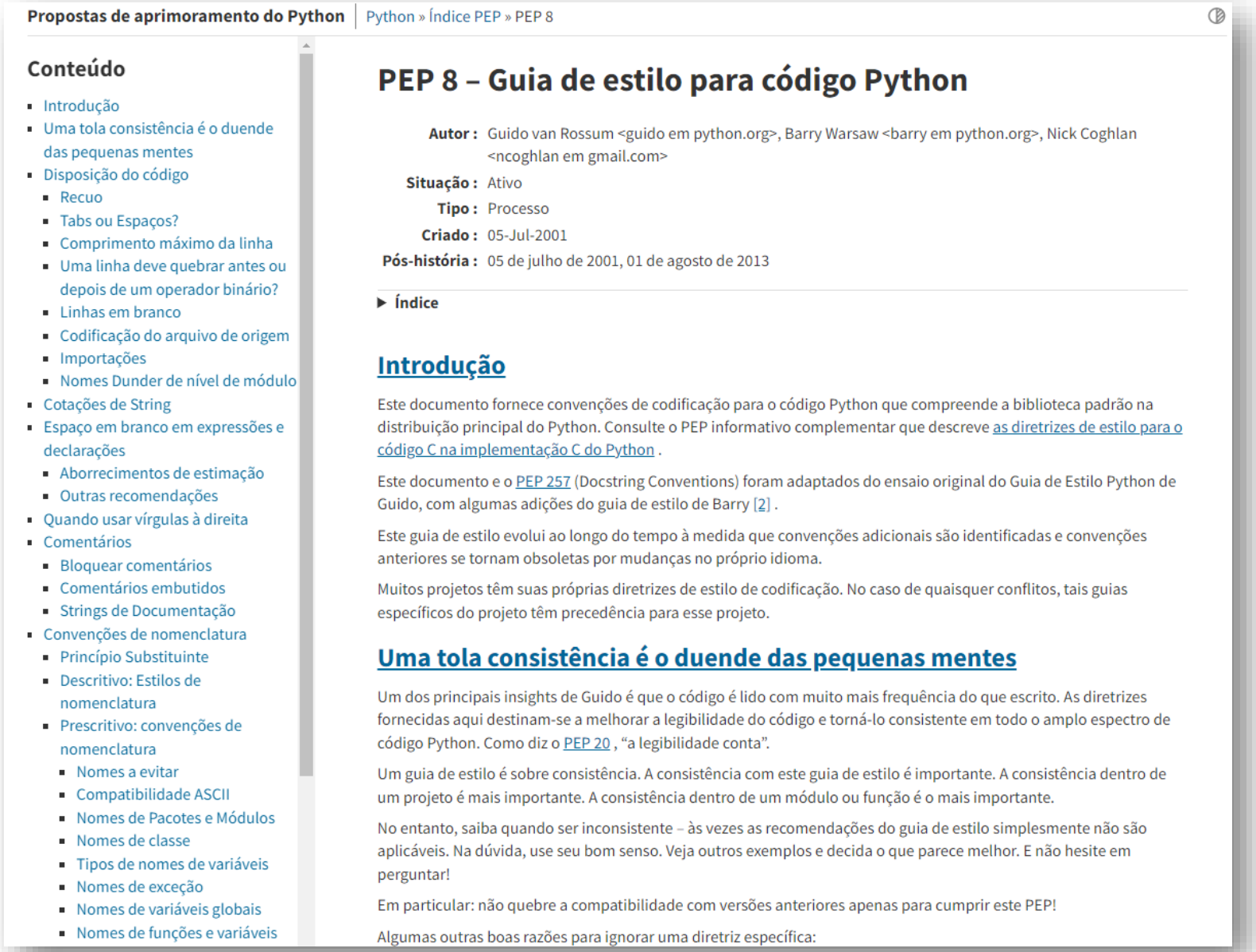
	PEP	Title	Authors
P	1	PEP Purpose and Guidelines	Warsaw, Hylton, Goodger, Coghlan
P	4	Deprecation of Standard Modules	Cannon, von Löwis
P	5	Guidelines for Language Evolution	Prescod
P	6	Bug Fix Releases	Aahz, Baxter
P	7	Style Guide for C Code	GvR, Warsaw
P	8	Style Guide for Python Code	GvR, Warsaw, Coghlan
P	10	Voting Guidelines	Warsaw
P	11	CPython platform support	von Löwis, Cannon
P	12	Sample reStructuredText PEP Template	Goodger, Warsaw, Cannon
P	387	Backwards Compatibility Policy	Peterson
PA	581	Using GitHub Issues for CPython	Mariatta
P	609	Python Packaging Authority (PyPA) Governance	Ingram, Gedam, Harihareswara

PEP-8

PEP 8 – Guia de Estilo para Codificação Python

https://peps.python.org/pep-0008/

PEP 8 – Style Guide for Python Code



Propostas de aprimoramento do Python | Python » Índice PEP » PEP 8

Conteúdo

PEP 8 – Guia de estilo para código Python

Autor : Guido van Rossum <guido em python.org>, Barry Warsaw <barry em python.org>, Nick Coghlan <ncoghlan em gmail.com>

Situação : Ativo

Tipo : Processo

Criado : 05-Jul-2001

Pós-história : 05 de julho de 2001, 01 de agosto de 2013

▶ Índice

Introdução

Este documento fornece convenções de codificação para o código Python que compreende a biblioteca padrão na distribuição principal do Python. Consulte o PEP informativo complementar que descreve as diretrizes de estilo para o código C na implementação C do Python .

Este documento e o PEP 257 (Docstring Conventions) foram adaptados do ensaio original do Guia de Estilo Python de Guido, com algumas adições do guia de estilo de Barry [2] .

Este guia de estilo evolui ao longo do tempo à medida que convenções adicionais são identificadas e convenções anteriores se tornam obsoletas por mudanças no próprio idioma.

Muitos projetos têm suas próprias diretrizes de estilo de codificação. No caso de quaisquer conflitos, tais guias específicos do projeto têm precedência para esse projeto.

Uma tola consistência é o duende das pequenas mentes

Um dos principais insights de Guido é que o código é lido com muito mais frequência do que escrito. As diretrizes fornecidas aqui destinam-se a melhorar a legibilidade do código e torná-lo consistente em todo o amplo espectro de código Python. Como diz o PEP 20 , "a legibilidade conta".

Um guia de estilo é sobre consistência. A consistência com este guia de estilo é importante. A consistência dentro de um projeto é mais importante. A consistência dentro de um módulo ou função é o mais importante.

No entanto, saiba quando ser inconsistente – às vezes as recomendações do guia de estilo simplesmente não são aplicáveis. Na dúvida, use seu bom senso. Veja outros exemplos e decida o que parece melhor. E não hesite em perguntar!

Em particular: não quebre a compatibilidade com versões anteriores apenas para cumprir este PEP!

Algumas outras boas razões para ignorar uma diretriz específica:

DIR

Lista todos os membros de um objeto ou escopo especificado

dir(<parâmetro>)

__builtins__

Todos os módulos “embutidos” da linguagem ou tipo de dado ou contexto específico

dir(__builtins__)

HELP

Ajuda para algum objeto, método, tipo de dado ou contexto em específico.

help(<parâmetro>)

PRINCÍPIO DE FUNCIONAMENTO DE SISTEMAS COMPUTACIONAIS

PRINCÍPIO DE FUNCIONAMENTO DE SISTEMAS COMPUTACIONAIS

`print()`

PRINCÍPIO DE FUNCIONAMENTO DE SISTEMAS COMPUTACIONAIS

`input()`

PRINT()

Mostrar algo na tela

print(“Olá Mundo!!”)

PRINT()

Mostrar algo na tela

print(“Olá Mundo!!”)

Olá Mundo!!

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem as com ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

PRINT()

- Semelhança dos comandos de uma linguagem com as ações que fazemos no nosso dia a dia...
- Esquentar algo no Micro-ondas:

print (" Olá Mundo! ")

PRINT()

```
print("Olá Mundo!")
```

```
Olá Mundo!
```

PRINT()

```
print("Olá Mundo!")
Olá Mundo!

print(6+5)
11
```

PRINT()

```
print("Olá Mundo!")
Olá Mundo!

print(6+5)
11

print("6"+"5")
65
```

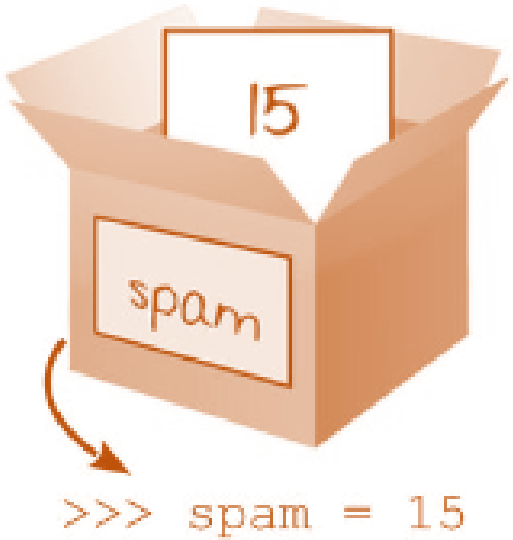
VARIÁVEIS...

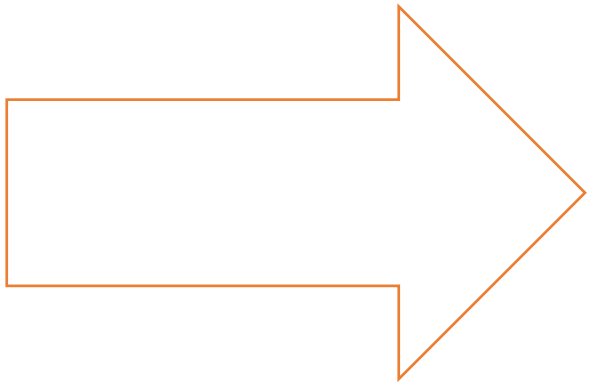
Mesmo conceito que aprendemos em matemática

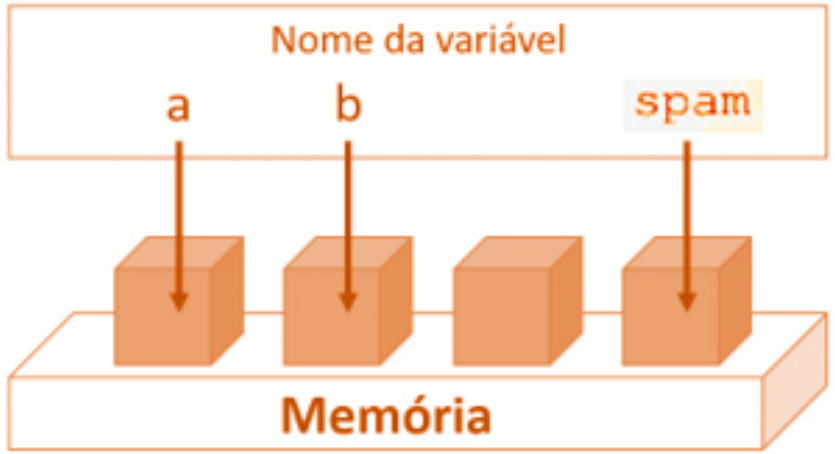
$x + 4 = 5 \rightarrow x = 1$

VARIÁVEIS...

Variáveis são espaços em memória, utilizados para “guardar” uma determinada informação







VARIÁVEIS...

```
x = 5

print(x)
5
```

VARIÁVEIS...

```
x = 5

print(x)
5
y = 6

print(x+y)
11
```

VARIÁVEIS...

ATENÇÃO!

```
x = 5
y = 6
print("x+y")
x+y
```

VARIÁVEIS...

```
x = 5

y = 6

print("Resultado da Soma: ", x+y)
Resultado da Soma:  11
```

VARIÁVEIS...

```
nome = “Maria”
idade = 25
print(nome, “ tem ”, idade, “ anos!”)
Maria tem 25 anos!
```

INPUT()

Recebendo alguma informação do usuário... E guardando em algum lugar!

```
nome = input(“Digite seu nome:”)
```

VARIÁVEIS...

~~nome = "Maria"~~

```
nome = input("Digite seu nome: ")
```

~~idade = 25~~

```
idade = input("Digite sua idade: ")
print(nome, " tem ", idade, " anos!")
```

VARIÁVEIS...

```
x = input("Digite o valor de x: ")
y = input("Digite o valor de y: ")
print("Resultado da Soma: ", x+y)
```

Resultado da Soma: **?**

VARIÁVEIS...

```
x = input("Digite o valor de x: ")
y = input("Digite o valor de y: ")
print("Resultado da Soma: ", x+y)
```

Resultado da Soma: **?**

PRIMEIRO DESAFIO

RESPOSTA AO DESAFIO 1

INPUT()

DESAFIO 1...

```
x = input("Digite o valor de x: ")
y = input("Digite o valor de y: ")
print("Resultado da Soma: ", x+y)
```

Resultado da Soma: **?**

DESAFIO 1...

```
x = input("Digite o valor de x: ")
y = input("Digite o valor de y: ")
print("Resultado da Soma: ", x+y)
```

5

6

Resultado da Soma: **56**

?

11

DESAFIO 1...

```
x = int(input("Digite o valor de x: "))
y = int(input("Digite o valor de y: "))
print("Resultado da Soma: ", x+y)
```

Resultado da Soma: !

Exemplo 1

Calcular a área de um quadrilátero, com dois lados iguais. Para isso faz a leitura de dois números inteiros, representando os lados e imprime os valores lidos e a área calculada.

base, altura, area representam os nomes das variáveis – podem ser do tipo inteiro ou do tipo real.

Exemplo 1

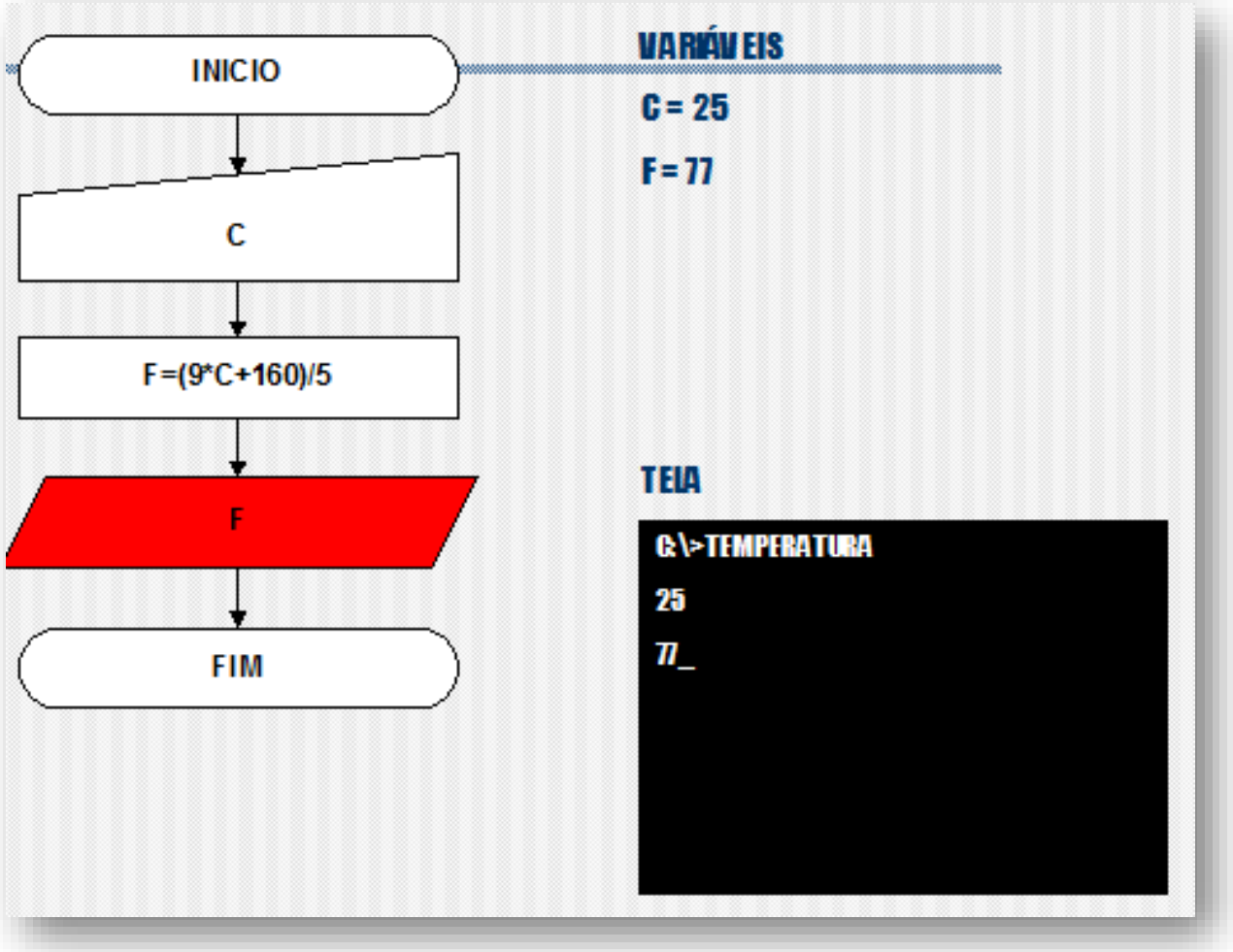
Calcular a área de um quadrilátero, com dois lados iguais. Para isso faz a leitura de dois números inteiros, representando os lados e imprime os valores lidos e a área calculada.

base, altura, area representam os nomes das variáveis – podem ser do tipo inteiro ou do tipo real.

```
base = float(input("Entre com a base:"))
altura = float(input("Entre com a altura:"))
area = base * altura
print("A área é = ", area)
```

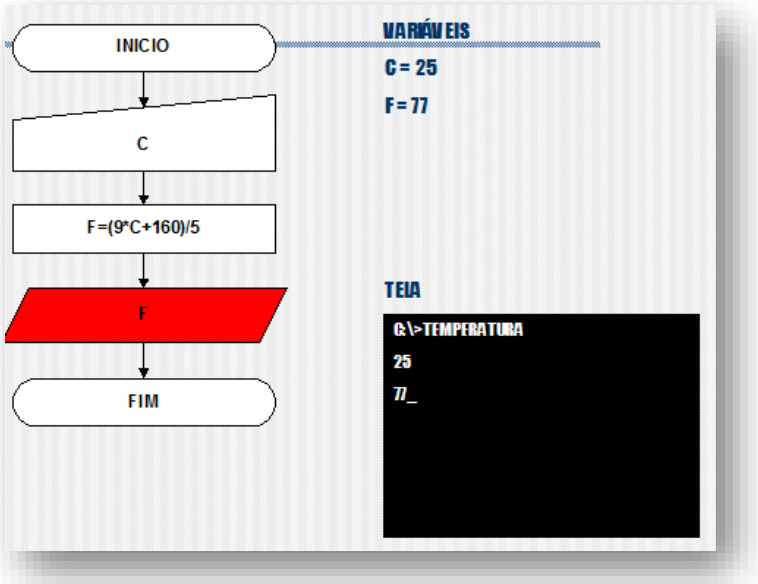
Exemplo 2

Ler a temperatura em °C e apresentar em °F.
Dados : F=(9*C+160)/5.



Exemplo 2

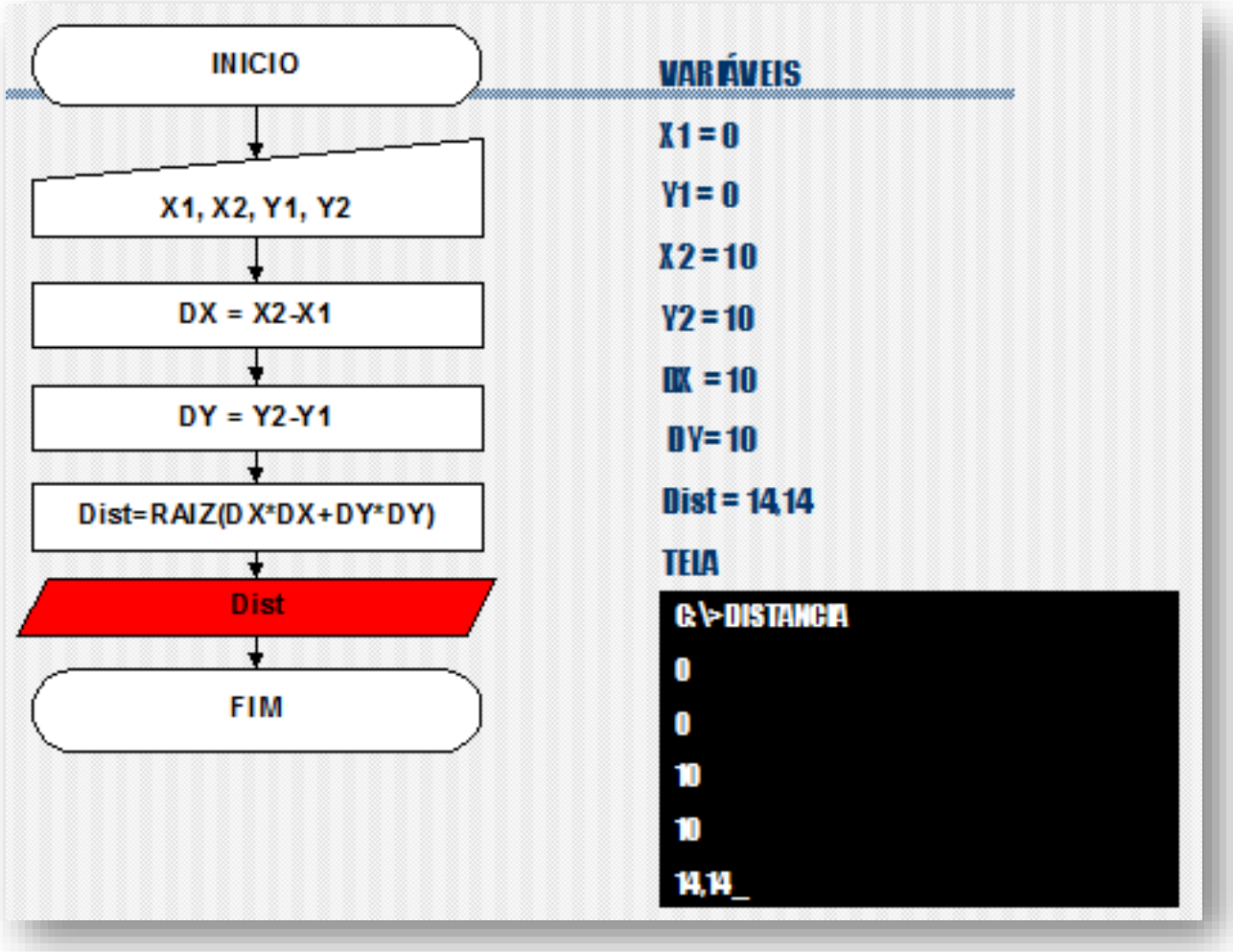
Ler a temperatura em ◦C e apresentar em ◦F.
Dados : F=(9*C+160)/5.



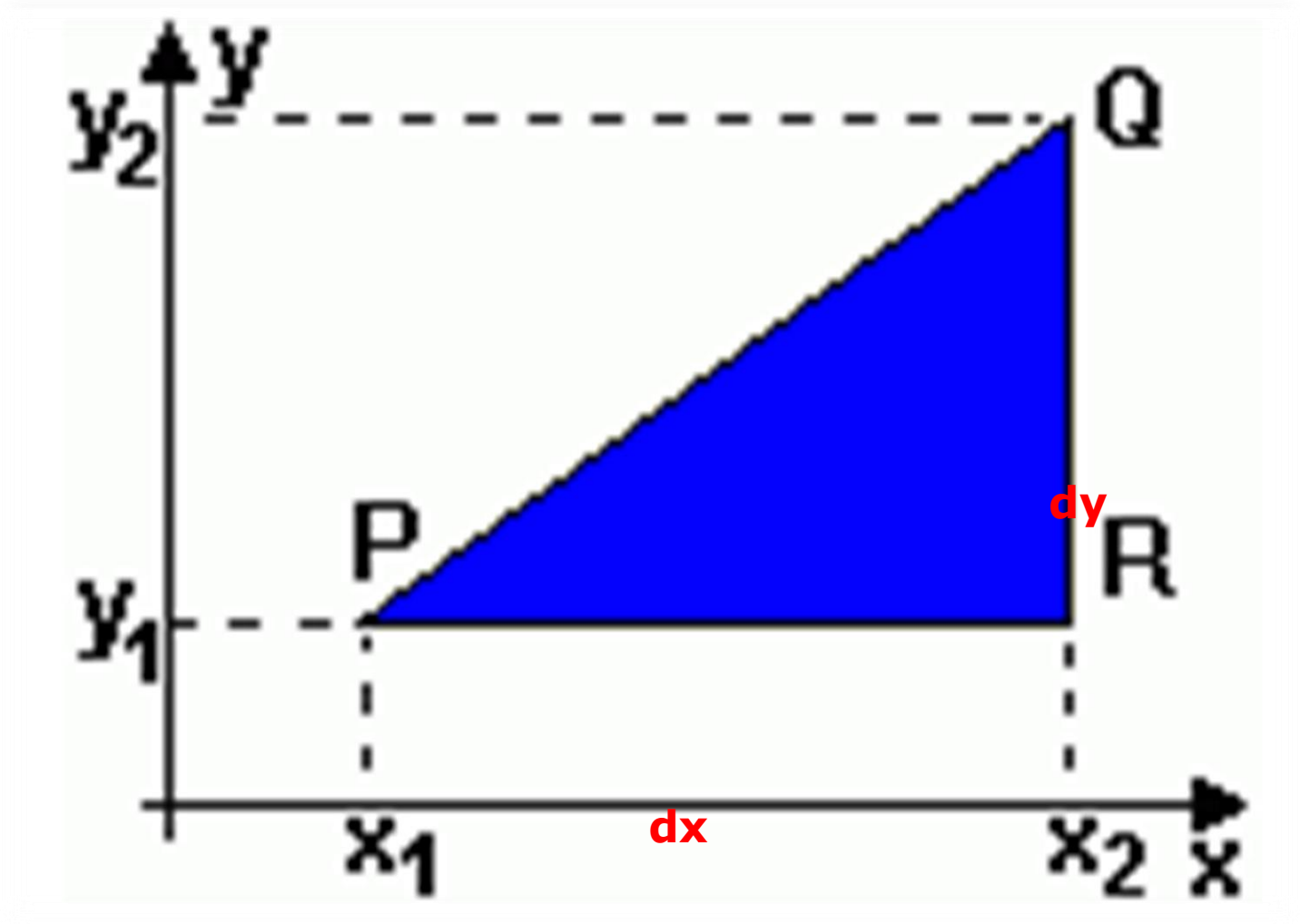

```
celsius = float(input("Entre com a temperatura em C:"))
fare = (9 * celsius + 160) / 5
print("A temperatura em F = ", fare)
```

Exemplo 3

CALCULAR A DISTÂNCIA ENTRE 2 PONTOS.



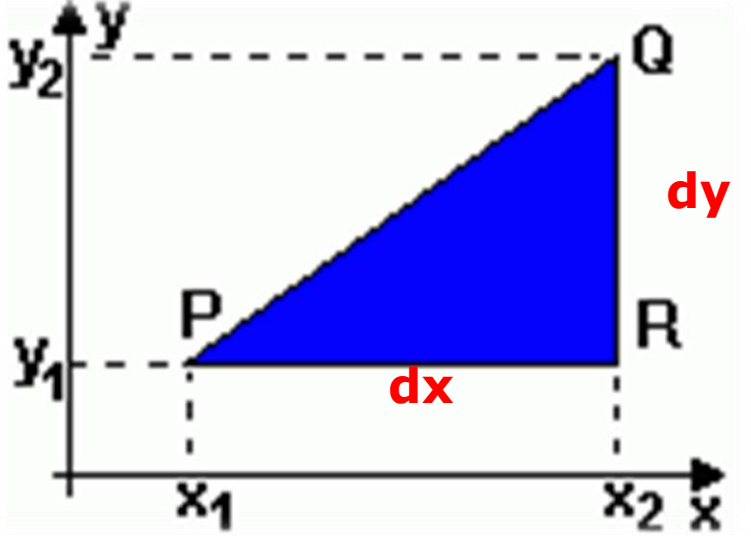
Exemplo 3



Exemplo 3

```
from math import pow, sqrt

x1 = float(input("Entre com x1: "))
y1 = float(input("Entre com y1: "))
x2 = float(input("Entre com x2: "))
y2 = float(input("Entre com y2: "))
dx = x2 - x1
dy = y2 - y1
d = sqrt(pow(dx,2)+pow(dy,2))
print("Distância = ", d)
```



VAMOS PARA A PRÁTICA ?!!!

Programando – Exercício 0

- Faça um programa para ler e escrever na tela o seu nome e a sua idade.

Exercício 1

Escreva um algoritmo que receba o nome do usuário e lhe deseje “Bom-dia!”.

Fonte: Slides da Aula 4 do Prof. Dr. José Nelson Falavinha Junior

58

Exercício 2

Faça um algoritmo que receba o ano de nascimento de uma pessoa e o ano atual e mostre:

a) A idade dessa pessoa em anos;
b) A idade dessa pessoa em meses;
c) A idade dessa pessoa em dias;
d) A idade dessa pessoa em semanas.

Fonte: Slides da Aula 4 do Prof. Dr. José Nelson Falavinha Junior

59

Exercício 3

Faça um algoritmo que receba o salário de um funcionário, calcule e mostre o novo salário, sabendo-se que este sofreu um aumento de 25%.

Exercício 4

Faça um algoritmo que receba o salário de um funcionário e o percentual de aumento, calcule e mostre o valor do aumento e o novo salário.

Exercício 5

Faça um algoritmo que receba o valor de um depósito e o valor da taxa de juros, calcule e mostre o valor do rendimento e o valor total depois do rendimento.

Exercício 6

Faça um algoritmo que calcule e mostre a área de um triângulo. Sabe-se que Área = (base * altura)/2

Exercício 7

Faça um algoritmo que calcule e mostre a área de um círculo. Sabe-se que

$$Area = \pi * R^2$$

$$\pi = 3,1415$$

Exercício 8

Sabe-se que:

- 1 pé = 12 polegadas
- 1 jarda = 3 pés
- 1 milha = 1760 jardas

Faça um algoritmo que receba uma medida em pés, faça as conversões a seguir e mostre os resultados:

a) Polegadas
b) Jardas
c) milhas

Exercício 9

Cada degrau de uma escada tem Xcm de altura. Faça um algoritmo que receba a altura de cada degrau em cm e a altura que o usuário deseja alcançar subindo a escada (em metros). Faça as conversões, calcule e mostre quantos degraus o usuário deverá subir para atingir seu objetivo.

Obs: não se preocupe com a altura do usuário!