Funções

prof. Guilherme Rey

Revisão

Programação

Python

Variáveis

Condicionais

Loops (while e for)

Listas

operadores.py

```
''''
Operadores Aritméticos:
+, -, /, *, %, **

Operadores para comparação:
==, <, <=, >, >=, !=

Operadores lógicos:
and, or, not

Operadores de atribuição:
=, +=, -=, *=, /=, %=, **=, //=
''''
```

comments.py

```
# Comentário
# Comentário 2
# Comentário 3

''' Comentário
Comentário 2
Comentário 3
'''
```

Guido van Rossum @gvanrossum · Sep 10, 2011

Replying to @BSUCSClub

@BSUCSClub Python tip: You can use multi-line strings as multi-line comments. Unless used as docstrings, they generate no code! :-)

9 70 130

variaveis.py

```python
# Principais tipos
texto = 'Sou um texto'
inteiro = 20     # integer
decimal = 3.14   # float
verdade = True   # booleano (True/False)
```

input_output.py

```
mensagem = input('Qual mensagem?')

print(mensagem)
```

LET'S CODE

while_com_condicionais.py

```
while True:
    area = input('Qual é a área mais legal?')
    if area == 'Computação':
        print('Acertou! Está livre!')
        break
    else:
        print('Poxa, tenta de novo!')
```

while_exemplo2_contador.py

```
contador = 0
N = 1
t = 1
while contador < N:
    print(t)     # imprime o valor de t no momento
    t *= 2       # equivalente a t = t * 2
    contador += 1
```

exercicio_revisao.py

```
# Escrever um programa que receba N
# notas, até que o usuário insira
# 'compute'. Depois de receber as
# notas, imprimir a média delas
```

while_exemplo2_contador.py

```
total = 0
contador = 0
while True:
    inp = input('Digite uma nota (ou compute) ')
    if inp == 'compute':
        break
    valor = float(inp)
    total += valor
    contador += 1
print(total / contador)
```

while_exemplo2_contador.py

```
notas = []
while True:
    inp = input('Digite uma nota (ou compute)')
    if inp == 'compute':
        break
    valor = float(inp)
    notas.append(valor)
print(sum(notas) / len(notas))
```

Como somar dois números?

somando.py

```
a = 1
b = 1
soma = a + b
print('a soma é: ', soma)
# a soma é 2
```

Como somar outros dois números?

somando.py

```
a = 1
b = 1
soma = a + b
print('a soma é: ', soma)
# ------------------------------
a = 3
b = 2
soma = a + b
print('a soma é: ', soma)
```

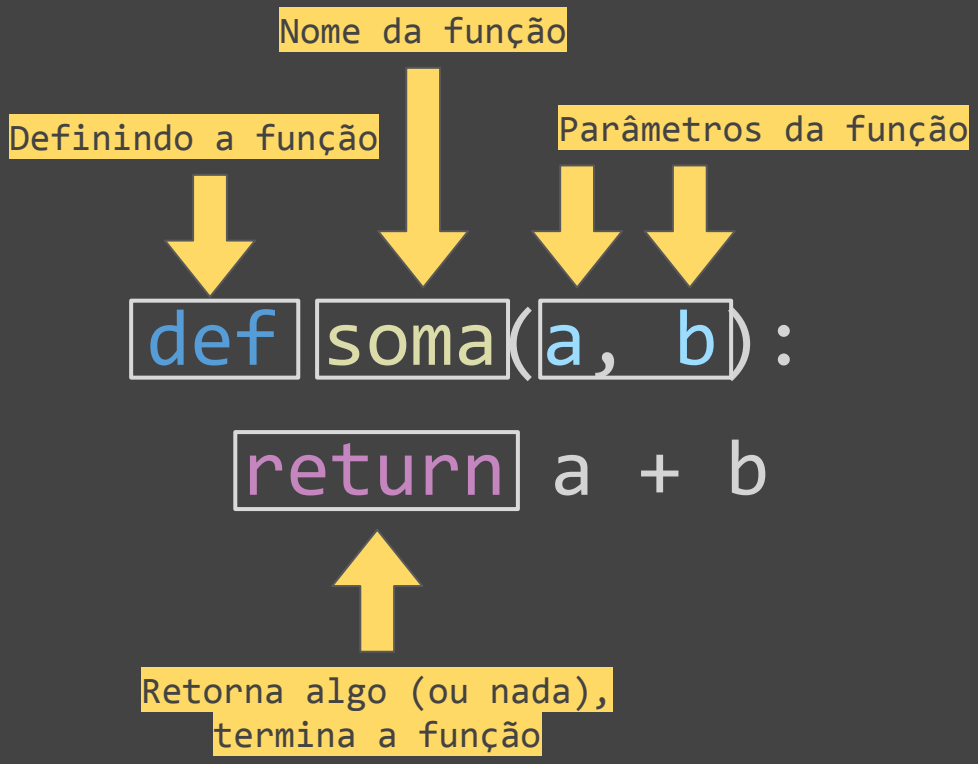
Como escrever algo que
sempre some?

Escreva uma função!

def_soma.py

```
def soma(a, b):
    return a + b
```

def_soma.py


Nome da função
Definindo a função
Parâmetros da função
def soma(a, b):
return a + b
Retorna algo (ou nada),
termina a função

somando_ruim.py

```python
a = 1
b = 1
soma = a + b
print('a soma é: ', soma)
# ------------------------------
a = 3
b = 2
soma = a + b
print('a soma é: ', soma)
```

somando_nice.py

```python
def soma(a, b):
    return a + b


soma1 = soma(1, 1)
soma2 = soma(2, 3)

print(soma1)
print(soma2)
```

Como descobrir o maior valor de uma lista de inteiros?

loop_2.py

```
inteiros = [4, 2, 10, 25, 4]
print(max(inteiros))
```

Função max, retorna o maior valor encontrado numa lista de inteiros

loop_2.py

max(*iterable*, *[, *key*, *default*])
max(*arg1*, *arg2*, **args*[, *key*])

Return the largest item in an iterable or the largest of two or more arguments.

If one positional argument is provided, it should be an iterable. The largest item in the iterable is returned. If two or more positional arguments are provided, the largest of the positional arguments is returned.

There are two optional keyword-only arguments. The *key* argument specifies a one-argument ordering function like that used for `list.sort()`. The *default* argument specifies an object to return if the provided iterable is empty. If the iterable is empty and *default* is not provided, a `ValueError` is raised.

If multiple items are maximal, the function returns the first one encountered. This is consistent with other sort-stability preserving tools such as `sorted(iterable, key=keyfunc, reverse=True)[0]` and `heapq.nlargest(1, iterable, key=keyfunc)`.

New in version 3.4: The *default* keyword-only argument.

LET'S CODE

Como descobrir o
maior valor par de uma lista
de inteiros?

entendendo_conceito.py

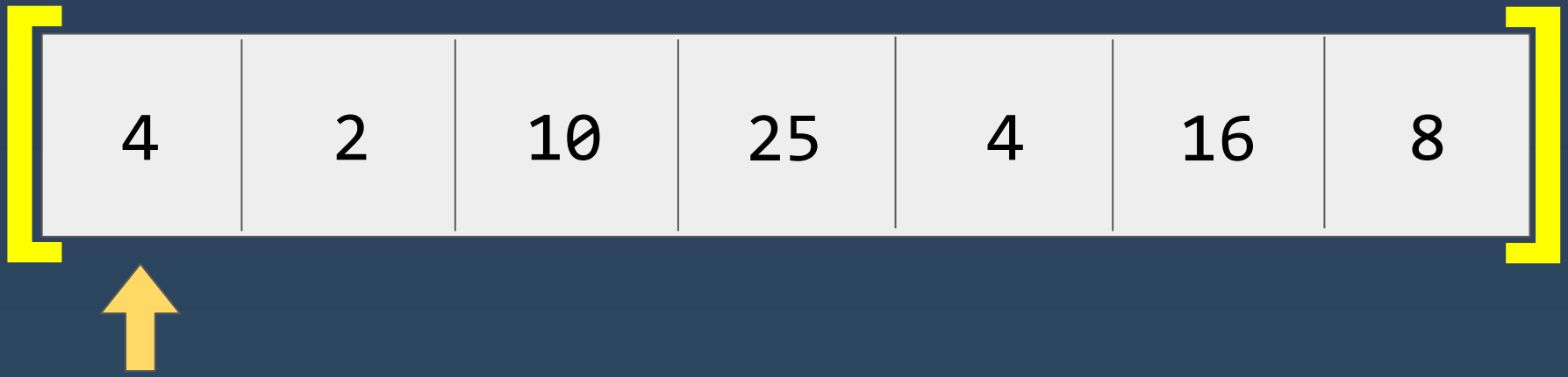
```
lista = [4, 2, 10, 25, 4, 16, 8]
```


```
maior_valor = 4
```

LET'S CODE

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```



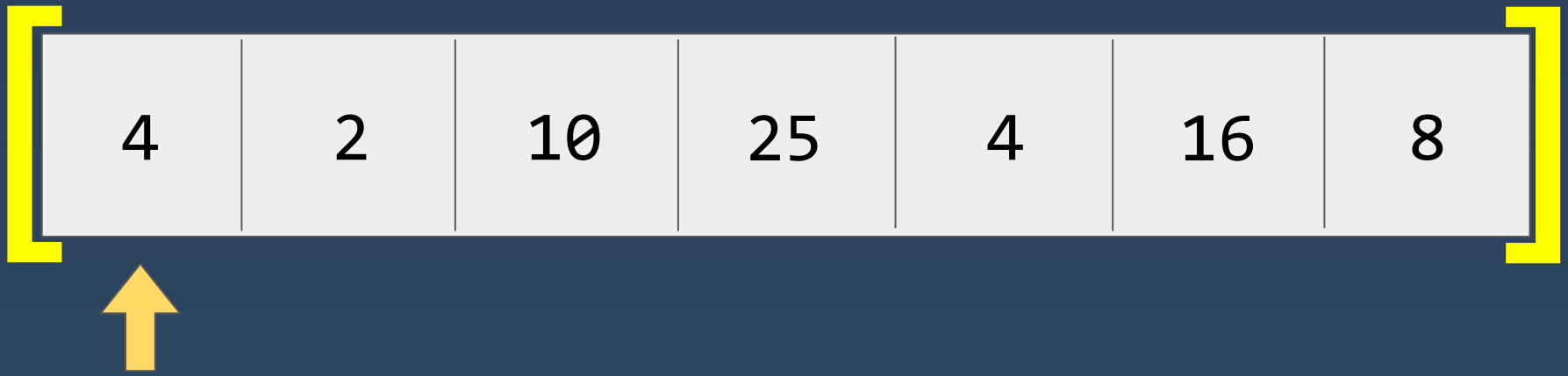
é par?

maior_valor = 4

LET'S CODE

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```



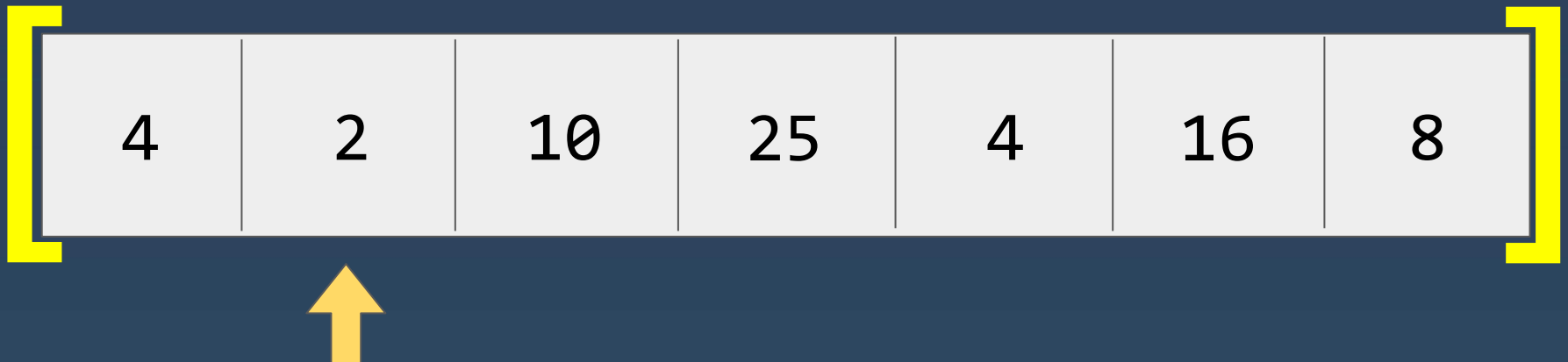
é par?

é maior que maior_valor?

maior_valor = 4

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```



é par?

é maior que maior_valor?

maior_valor = 4

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```

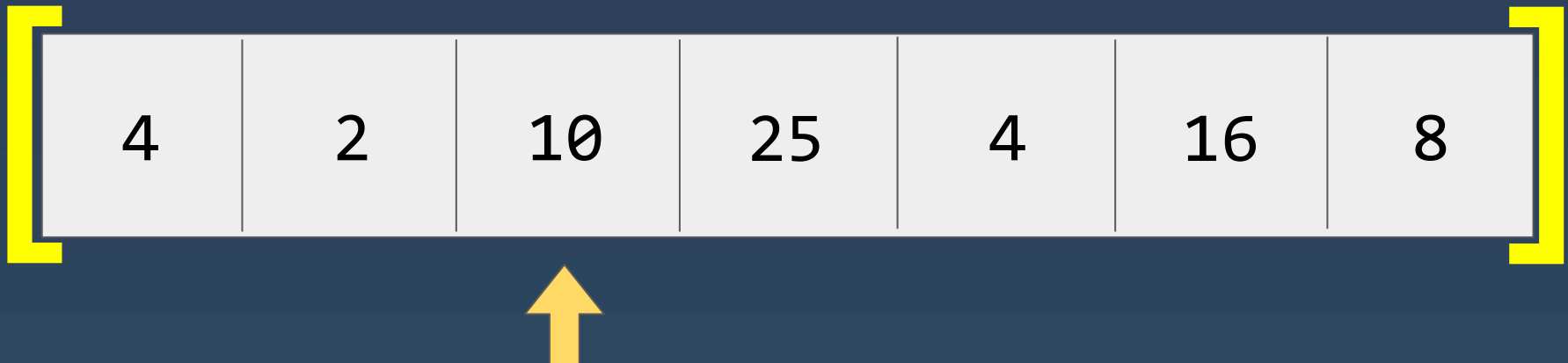
é par?

é maior que
maior_valor?

maior_valor = 4

entendendo_conceito.py

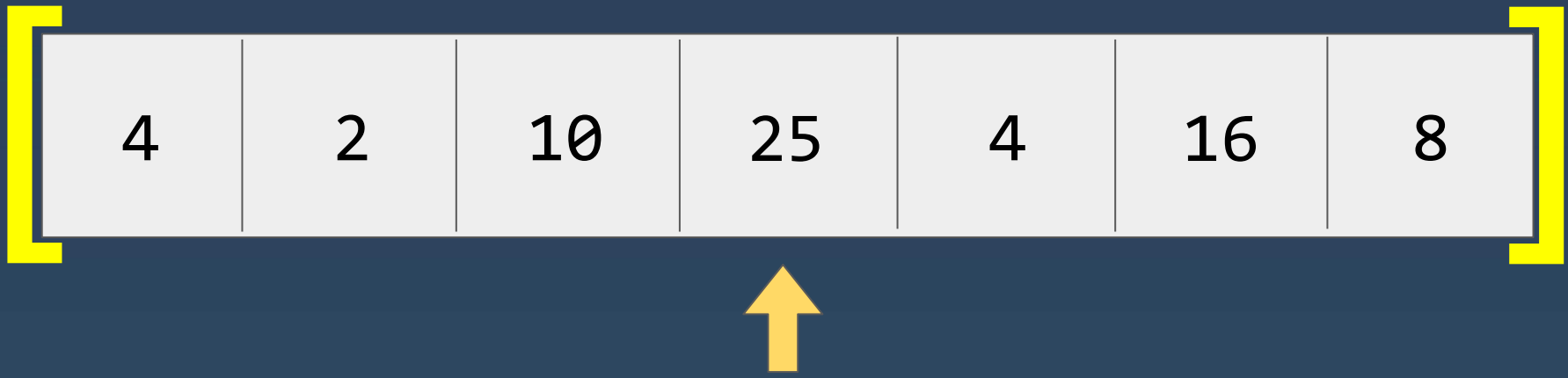
```
lista = [4, 2, 10, 25, 4, 16, 8]
```



LET'S CODE

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```




```
maior_valor = 10
```

LET'S CODE

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```

é par?

é maior que
maior_valor?

maior_valor = 10

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```

é par?

é maior que maior_valor?

maior_valor = 10

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```

é par?

é maior que maior_valor?

maior_valor = 16

entendendo_conceito.py

```
lista = [4, 2, 10, 25, 4, 16, 8]
```

é par?

é maior que maior_valor?

maior_valor = 16

LET'S CODE

def_max_par.py

```
def max_par(inteiros):
    maior_valor = inteiros[0]
    for num in inteiros:
        if num % 2 == 0 and num >= maior_valor:
            maior_valor = num

    return maior_valor
```

def_max_par.py

```
inteiros = [4, 2, 10, 25, 4, 16, 8]
print(max_par(inteiros))
```

Chama o código que escrevemos dentro de **max_par**

def_max_par.py

```
'''
Recebe uma lista de inteiros e devolve
o maior valor par encontrado na lista
'''
def max_par(inteiros):
    ... #(aqui vai o codigo)
```

def_max_par.py

```
'''
... #(aqui vai o codigo)
```

"Comments are, at best, a necessary evil.
The proper use of comments is to compensate
for our failure to express ourself in code [...]
Truth can only be found in one place: the code."
— Robert C. Martin, *Clean Code*

def_max_par.py

```
def max_par(inteiros):
    maior_valor = inteiros[0]
    for num in inteiros:
        if num % 2 == 0 and num >= maior_valor:
            maior_valor = num

    return maior_valor
```

def_max_par.py

```
def max_par(inteiros):
    maior_valor = inteiros[0]
    for num in inteiros:
        if eh_par(num) and num >= maior_valor:
            maior_valor = num

    return maior_valor
```

LET'S CODE

def_max_par.py

```
'''
Recebe um inteiro num e retorna True se
é par, False caso contrário
'''
def eh_par(num):
    return num % 2 == 0
```

Bibliotecas

libraries

aula += 1