Desvendando programação com Python

 Carlos ROLAND

python™

O que é um programa?

Qual a palavra mais usada no texto?

We are surrounded in our daily lives with computers ranging from laptops to cell phones. We can think of these computers as our “personal assistants” who can take care of many things on our behalf. The hardware in our current-day computers is essentially built to continuously ask us the question, “What would you like me to do next?”
Programmers add an operating system and a set of applications to the hardware and we end up with a Personal Digital Assistant that is quite helpful and capable of helping us do many different things. Our computers are fast and have vast amounts of memory and could be very helpful to us if we only knew the language to speak to explain to the computer what we would like it to “do next”. If we knew this language, we could tell the computer to do tasks on our behalf that were repetitive. Interestingly, the kinds of things computers can do best are often the kinds of things that we humans find boring and mind-numbing.
For example, look at the first three paragraphs of this chapter and tell me the most commonly used word and how many times the word is used. While you were able to read and understand the words in a few seconds, counting them is almost painful because it is not the kind of problem that human minds are designed to solve. For a computer the opposite is true, reading and understanding text from
a piece of paper is hard for a computer to do but counting the words and telling you how many times the most used word was used is very easy for the computer: python word

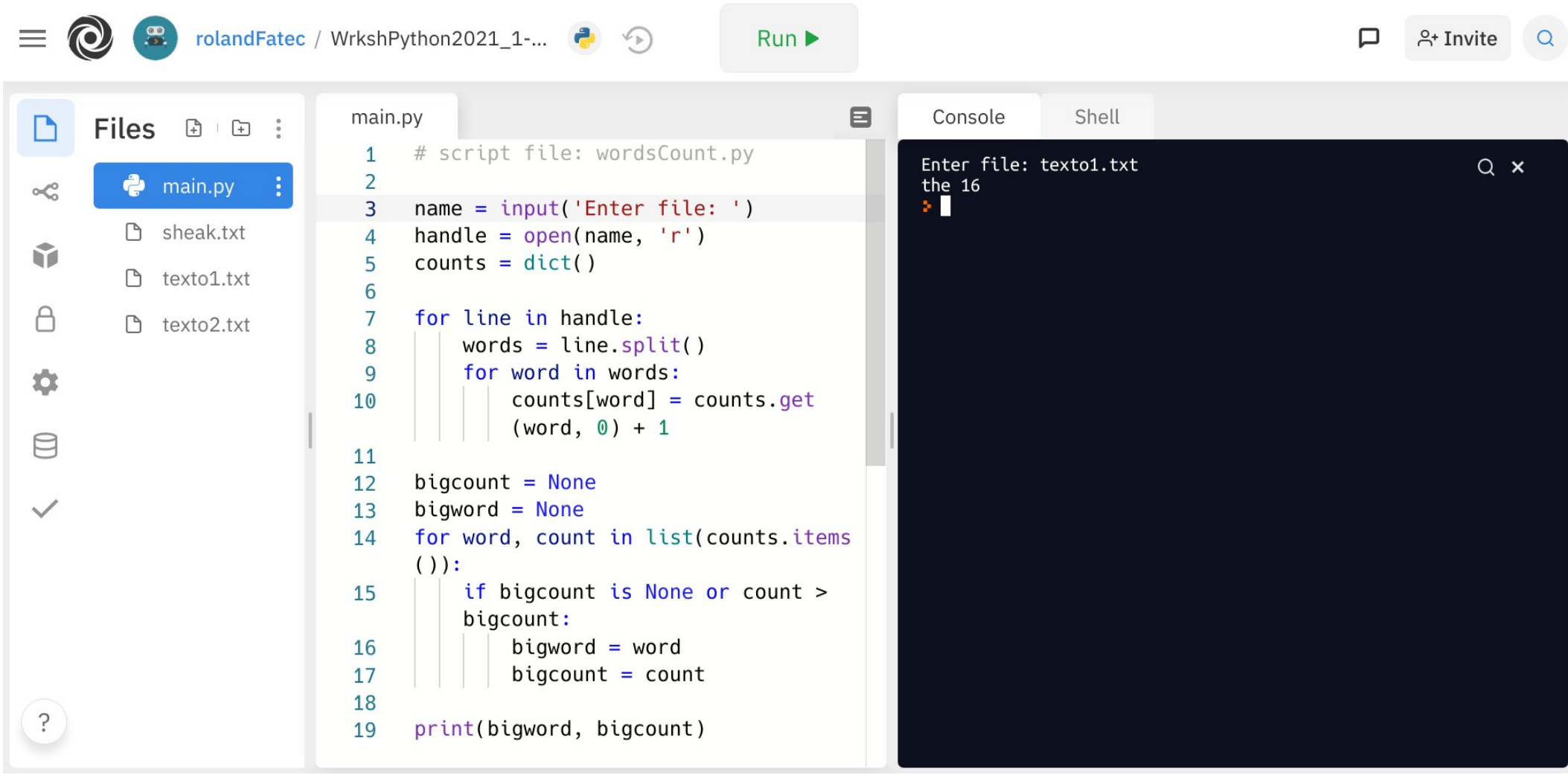
O que é um programa?

```
# script file: wordsCount.py

name = input('Enter file:')
handle = open(name, 'r')
counts = dict()

for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Executando o script

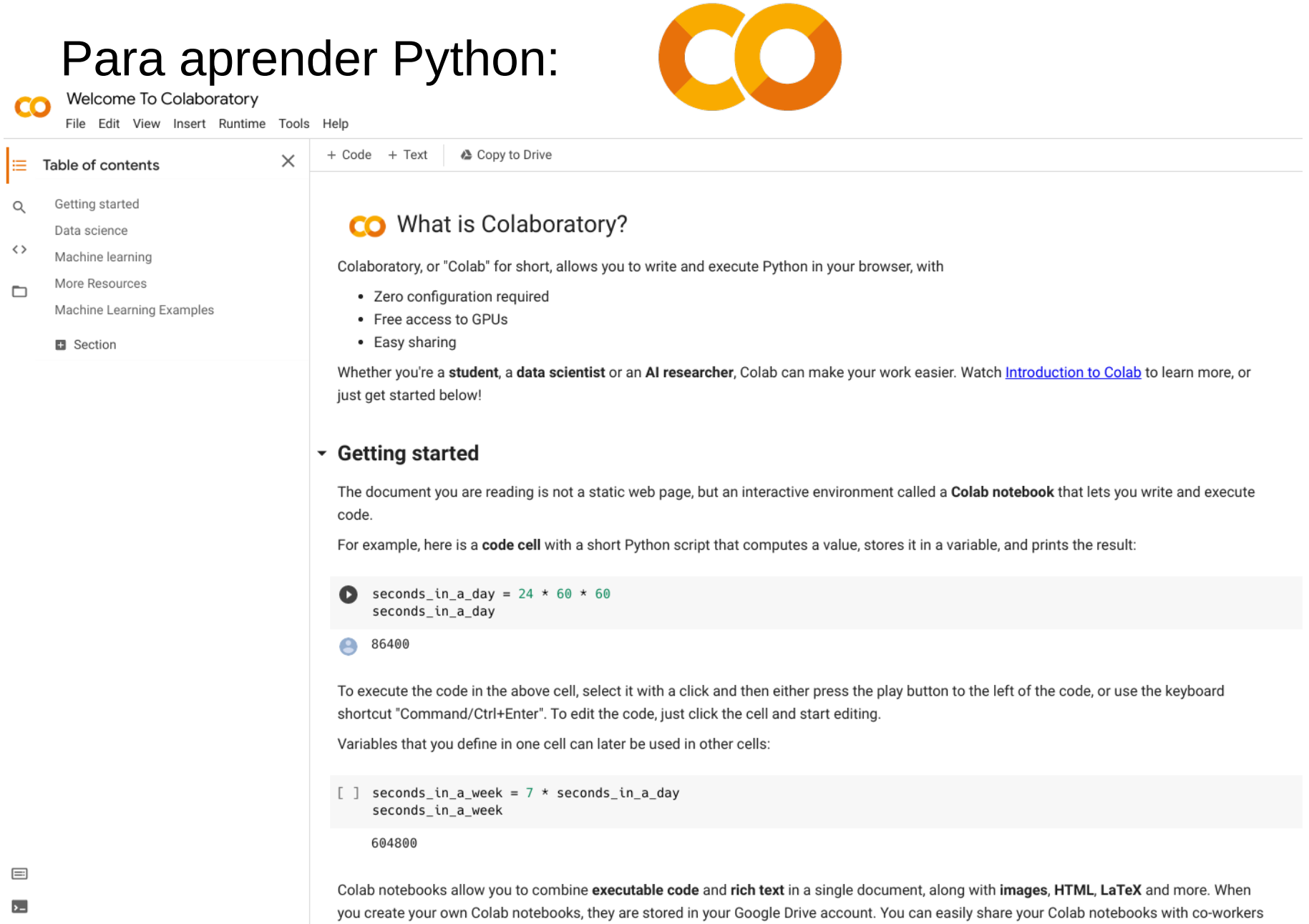
```
$python3 wordsCount.py
Enter file:words.txt
the 16
$
```

Executando o script



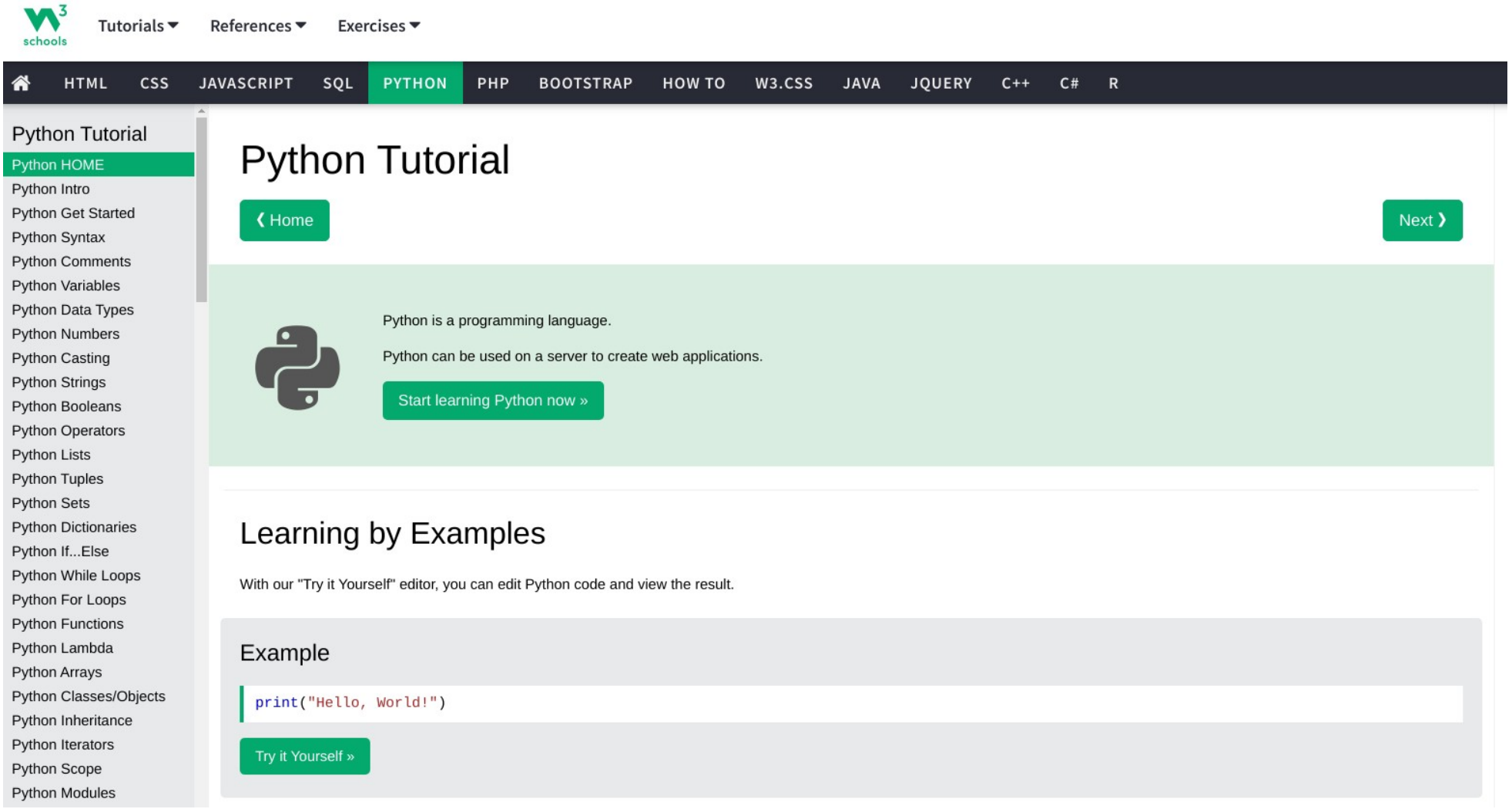
Hello, World!

Para aprender Python:



Hello, World!

Para aprender Python:



Variáveis

Nome para um valor armazenado em memória

Variáveis

Nome para um valor armazenado em memória

Criadas por comandos de atribuição (=)

```
>>> message = 'Dado inválido! Tecle <ENTER>.'
>>> n = 17
>>> pi = 3.1415926535897931
```

Variáveis

Executar os comandos. Qual o resultado? Porque?

```
>>> message = 'Dado inválido! Tecle <ENTER>.'
>>> n = 17
>>> pi = 3.1415926535897931
```

Variáveis

Executar os comandos. Qual o resultado? Porque?

```
>>> message = 'Dado inválido! Tecle <ENTER>.'
>>> n = 17
>>> pi = 3.1415926535897931
```

Como visualizar os valores?

```
>>>print(message)
>>>print(n)
>>>print(pi)
```

Variáveis

Executar os comandos. Qual o resultado? Porque?

```
>>> message = 'Dado inválido! Tecle <ENTER>.'
>>> n = 17
>>> pi = 3.1415926535897931
```

Como visualizar os valores?

```
>>>print(message)
>>>print(n)
>>>print(pi)
```

Como visualizar os tipos?

```
>>>type(message)
>>>type(n)
>>>type(pi)
```


Time for
a Break

Nomes de variáveis e palavras-chave

Significativas e descritivas

Nomes de variáveis e palavras-chave

Significativas e descritivas

Sem limitação de tamanho

Nomes de variáveis e palavras-chave

Significativas e descritivas

Sem limitação de tamanho

Podem conter letras e números

Nomes de variáveis e palavras-chave

Significativas e descritivas

Sem limitação de tamanho

Podem conter letras e números

Não podem começar com números

Nomes de variáveis e palavras-chave

Significativas e descritivas

Sem limitação de tamanho

Podem conter letras e números

Não podem começar com números

Sensíveis a caixa

Boas práticas (BP): iniciar com minúsculas

Nomes de variáveis e palavras-chave

Significativas e descritivas

Sem limitação de tamanho

Podem conter letras e números

Não podem começar com números

Sensíveis a caixa

- Boas práticas (BP): iniciar com minúsculas

Podem usar sublinhado (_)

- **BP:** usar _ no início para variáveis de bibliotecas

Readability Counts

https://youtu.be/knMg6G9_XCg

Nomes de variáveis e palavras-chave

Testar:

```
>>>76trombones = 'big parade'

>>>more@ = 1000000

>>>class = 'Advanced Theoretical Zymurgy'

>>>precoUnitario = 3456.80

>>>pr_unit = 3456.80

>>>lado1 = 10

>>>taxaJuroAnual = 546.68
```

Palavras reservadas - revisão

and
as
assert
break
class
continue
def
del
elif
else
except
False
finally
for
from
global
if
import
in
is
lambda
None
nonLocal
not
or
pass
raise
return
True
try
while
with
yield

>import keyword
>keyword.kwlist

Comandos

Unidade de código executável pelo interpretador

Comandos

Unidade de código executável pelo interpretador

Exemplos:

- print()
- =
- type()
- int()
- str()
- float()

Comandos

Unidade de código executável pelo interpretador

Exemplos:

print()
=
type()
int()
str()
float()

No modo interativo são executados quando teclado <ENTER>

Comandos

Unidade de código executável pelo interpretador

Exemplos:

print()
=
type()
int()
str()
float()

No modo interativo são executados quando teclado <ENTER>

No modo script todos são executados na sequência de escrita

Nomes de variáveis e palavras-chave

Testar (modos interativo e programado):

```
>>>firstname = ‘Rick'

>>>lastname = ‘Riordan’

>>>series = ‘Percy Jackson’

>>>book = ‘Percy Jackson e os Olimpianos’

>>>precoUnitario = 34.90

>>>pr_unit = 34.90
```

Nomes de variáveis e palavras-chave

Testar (modos interativo e programado):

```
>>>print(firstname, lastname)

>>>print(book, precoUnitario)

>>>print(book, pr_unit*2)

>>>print(book,”:“,pr_unit*2)
```

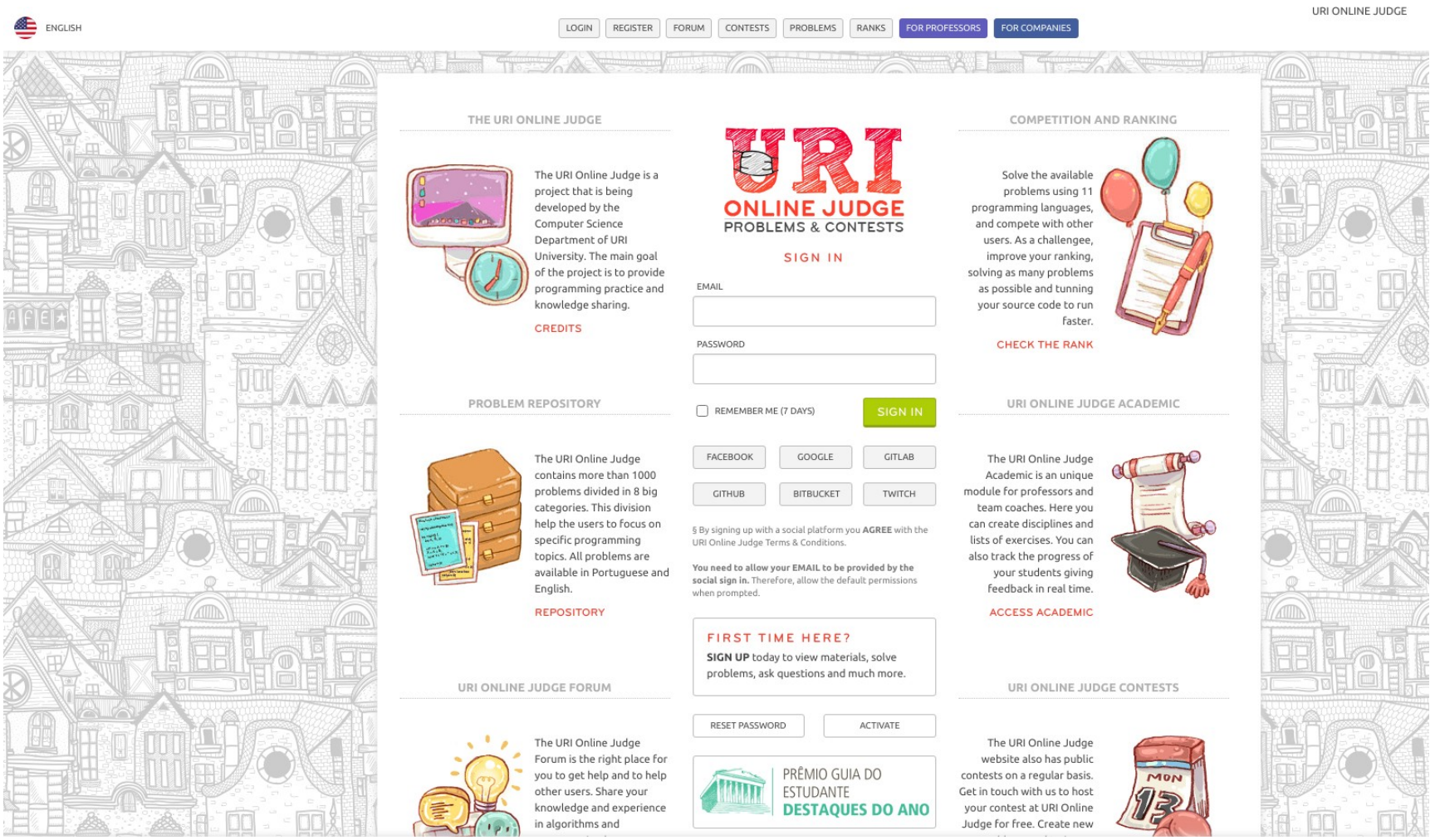
Concatenando – juntando - strings:

```
firstname = ‘Rick’

lastname =  ’Riordan’

print(firstname + lastname)

print(firstname + “ ” + lastname)
```

VAMOS PROGRAMAR!

Para treinar:

URI Online Judge - Problems & Contests




That's all Folks!