Python: do básico à autonomia intermediária
`aula 1`

Bernardo Alves Furtado

February 7, 2022

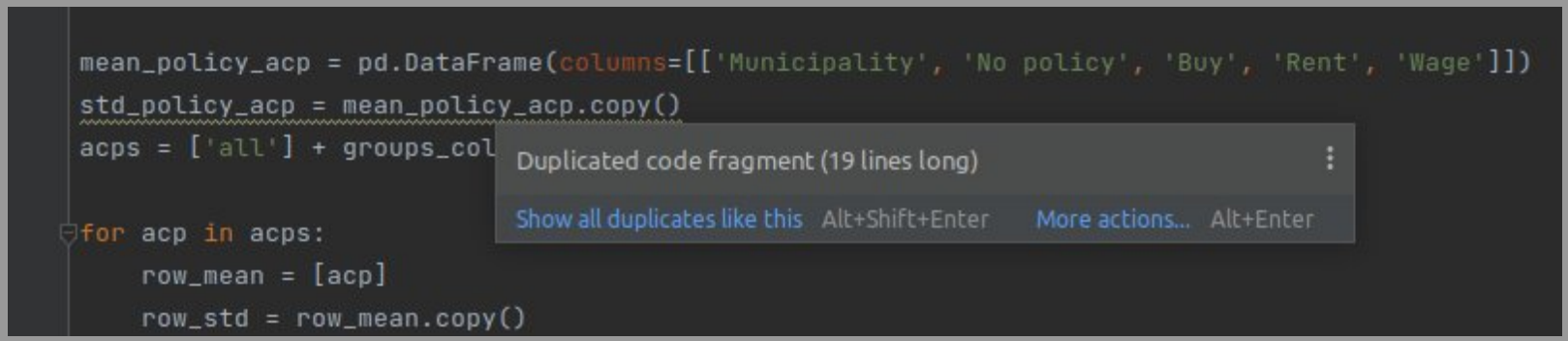
Menu do dia

- Functions
  - Sequence, main
  - Parâmetros
  - First `import`, `return`

Continuando última aula...

1. `import antigravity`
2. Outra rotina: captura tela, extração de informação, salvamento e plot
3. Reddit: what_have_you_recently_automated_at_work_using
4. py.checkio.org?
5. instalações?
6. 'hello.py', alguém?
7. aula **0**?

Duplicate – example



Exercicio Py.Checkio.org

```python
def mult_two(a, b):
    # your code here
    return a * b


if __name__ == '__main__':
    print("Example:")
    print(mult_two(3, 2))

    # These "asserts" are used for self-checking and not for an auto-testing
    assert mult_two(3, 2) == 6
    assert mult_two(1, 0) == 0
    print("Coding complete? Click 'Check' to earn cool rewards!")

```

ipea Institute for Applied Economic Research

Continuando II.. Reserved keywords python

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

Figure 1:
https://docs.python.org/3.3/reference/lexical_analysis.html#keywords

Continuando III... Built-in functions

Built-in Functions

A	**E**	**L**	**R**
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	**F**	**M**	
ascii()	filter()	map()	**S**
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	**G**	**N**	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	**H**	object()	
callable()	hasattr()	oct()	**T**
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		**P**	**V**
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	**Z**
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

Figure 2: source: https://docs.python.org/3/library/functions.html

function1.py

```
def print_sentences():
        print('A vida é bela')
        print('Adoro Python')

print_sentences()
```

function2.py

```
def soma():
        x = int(input('Entre um número: '))
        y = int(input('Entre outro número: '))
        print('A soma é:', x + y)

if __name__ == '__main__':
        soma()
```

function3.py

```
def soma(a, b):
        print('A soma é:', a + b)

if __name__ == '__main__':
        x = 10
        y = 9
        soma(x, y)
```

$ python **function3.py**

function5.py

```
import math

def area_circle(r):
        area = math.pi * r
        return area

if __name__ == '__main__':
        raio = 2
        result = area_circle(raio)
        print('A área é: {:.2f}'.format(result))
```

Observações

- Parâmetros
- Everything in python is an OBJECT
  - Tem atributos e funções típicas de cada tipo (built-in ou criado)
  - podem ser enviados como parâmetros, por exemplo
- Composition:
  - import math
  - 2 * math.pi * math.pow(2, 3)
- Variables and parameters are local. Show. Teste.

ipea Institute for Applied Economic Research

Gotchas – erros comuns e roteiro

1. Função começa com a palavra `def`
2. Tem nome e termina com **dois pontos**
3. Abaixo da primeira linha, sempre há 4 espaços/tab
4. `https://www.youtube.com/watch?v=SsoOG6ZeyUI`
5. Pode print(output) ou `return` objeto

PyCheckIO

https://py.checkio.org/en/mission/is-even/
https://py.checkio.org/en/mission/all-upper/

namespaces, local variables, exercises

- Exercício1: Faça um programa que calcule a área do triângulo. Parâmetros: base e altura. Fórmula: base * altura / 2
- Exercício2: importe o programa de área e print o resultado.

namespaces, local variables, exercises

- Exercício1: Faça um programa que calcule a área do triângulo. Parâmetros: base e altura. Fórmula: base * altura / 2
- Exercício2: importe o programa de área e print o resultado.
- `function4.py`
- `function6.py`
- `function7_function.py`
- Exercicio Think Python 3.1 p. 26! `exercise3_1.py`

namespaces, local variables, exercises

- Exercício1: Faça um programa que calcule a área do triângulo. Parâmetros: base e altura. Fórmula: base * altura / 2
- Exercício2: importe o programa de área e print o resultado.
- `function4.py`
- `function6.py`
- `function7_function.py`
- Exercicio Think Python 3.1 p. 26! `exercise3_1.py`
- Follow TP 3.2
- Implemente `floor_modulus.py` em uma função
  - Transforma número de minutos em horas e minutos
- Leia Chapter 3 (Think Python)