Experimentando Python

“Ensinar é indicar o caminho,
mas na viagem cada um vai ver
o que quiser ver.”

(Plotino)

1.Introdução

Let´s understand


python

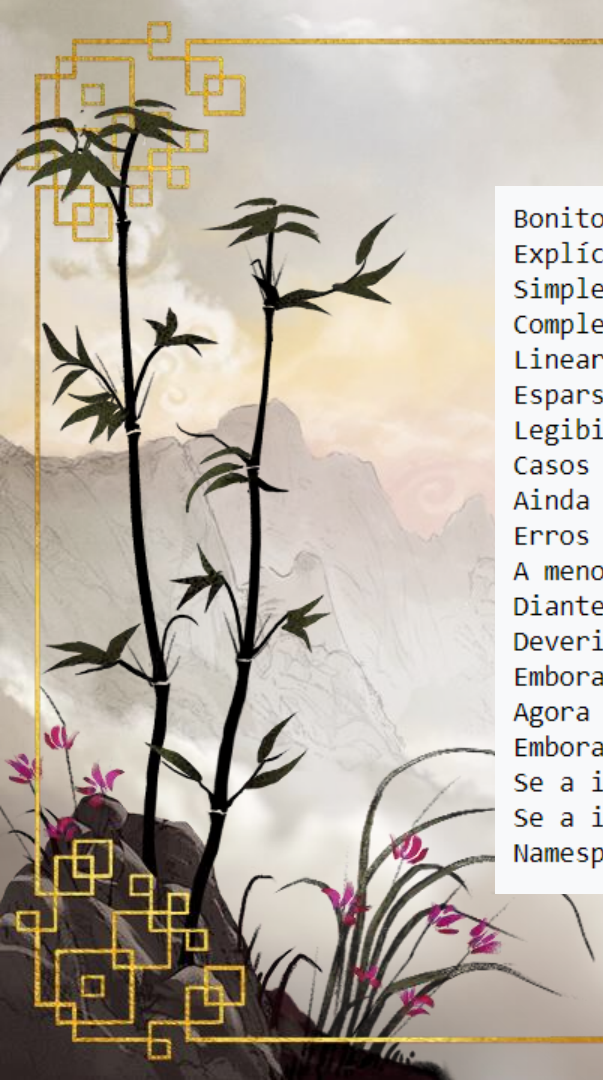
Python

- É uma linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.
- Foi lançada por Guido van Rossum em 1991.
- Muito popular nas áreas da tecnologia relacionadas à análise de dados, pesquisa, desenvolvimento de algoritmos e IA

```
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!
```



```
Bonito é melhor que feio
Explícito é melhor que implícito
Simples é melhor que complexo
Complexo é melhor que complicado
Linear é melhor do que aninhado
Esparso é melhor que denso
Legibilidade conta
Casos especiais não são especiais o bastante para quebrar as regras.
Ainda que praticidade vença a pureza
Erros nunca devem passar silenciosamente.
A menos que sejam explicitamente silenciados
Diante da ambiguidade, recuse a tentação de adivinhar
Deveria haver um — e preferencialmente apenas um — modo óbvio para fazer algo.
Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês
Agora é melhor que nunca
Embora nunca freqüentemente seja melhor que já
Se a implementação é difícil de explicar, é uma má ideia
Se a implementação é fácil de explicar, pode ser uma boa ideia
Namespaces são uma grande ideia — vamos ter mais dessas!
```

2.Python

Let´s code – Tipos de Dados

Operações matemáticas

```
2 + 3
```

```
5
```

```
3 - 1
```

```
2
```

```
4 * 3
```

```
12
```

```
9 / 2
```

```
4.5
```

```
9 % 2
```

```
1
```

```
2 ** 4
```

```
16
```

Operadores Lógicos

```
In [45]: 1 > 2
Out[45]: False

In [46]: 1 < 2
Out[46]: True

In [47]: 1 >= 1
Out[47]: True

In [48]: 1 <= 4
Out[48]: True

In [49]: 1 == 1
Out[49]: True

In [37]: 'oi' == 'tchau'
Out[37]: False
```

```
In [51]: (1 > 2) and (2 < 3)
Out[51]: False

In [52]: (1 > 2) or (2 < 3)
Out[52]: True

In [53]: (1 == 2) or (2 == 3) or (4 == 4)
Out[53]: True
```

Definição de Variáveis

```
>>> #comentários
>>> anoAtual = 2024
>>> anoNascimento = 1969
>>> nome = 'Lucilia'
>>> idade = anoAtual - anoNascimento
>>> print(nome)
    Lucilia
>>> print(idade)
    55
>>> print('Meu nome é %s e tenho %d anos' %(nome, idade))
    Meu nome é Lucilia e tenho 55 anos
```

3.Python

Outros Tipos de Dados – Let's code

Listas

```
In [20]: [1,2,3]
Out[20]: [1, 2, 3]

In [1]: ['Oi',1,[1,2]]
Out[1]: ['Oi', 1, [1, 2]]

In [12]: minha_lista = ['a','b','c']

In [13]: minha_lista.append('d')

In [14]: minha_lista
Out[14]: ['a', 'b', 'c', 'd']

In [15]: minha_lista[0]
Out[15]: 'a'

In [16]: minha_lista[1]
Out[16]: 'b'
```

```
In [17]: minha_lista[1:]
Out[17]: ['b', 'c', 'd']

In [18]: minha_lista[:1]
Out[18]: ['a']

In [28]: minha_lista[0] = 'NOVO'

In [29]: minha_lista
Out[29]: ['NOVO', 'b', 'c', 'd']

In [3]: lista2 = [1,2,3,[4,5,['nome']]]

In [4]: lista2[3]
Out[4]: [4, 5, ['nome']]

In [5]: lista2[3][2]
Out[5]: ['nome']

In [6]: lista2[3][2][0]
Out[6]: 'nome'
```

Dicionários

```
In [7]: nome = 'Lucília'

In [9]: d = {'nome':nome,'idade':51, 'hobby':'codar', 'filhos':['Bruna', 'André']}

In [10]: d
Out[10]: {'nome': 'Lucília',
          'idade': 51,
          'hobby': 'codar',
          'filhos': ['Bruna', 'André']}

In [11]: d['hobby']
Out[11]: 'codar'

In [12]: d['filhos'][0]
Out[12]: 'Bruna'
```

Tuplas

```
In [40]: t = (1,2,3)

In [13]: l = [1, 2, 3]

In [41]: t[0]
Out[41]: 1

In [14]: l.append(4)

In [15]: l
Out[15]: [1, 2, 3, 4]

In [16]: l[0] = 'novo'

In [17]: l
Out[17]: ['novo', 2, 3, 4]
```

```
In [18]: t.append(4)
---------------------
NameError
<ipython-input-18-a
----> 1 t.append(4)

NameError: name 't'
```

```
In [42]: t[0] = 'NEW'
---------------------
TypeError
<ipython-input-42-9
----> 1 t[0] = 'NEW'

TypeError: 'tuple'
```

4.Python

Estruturas de Controle – Let´s code

Estruturas Condicionais

```
In [20]: if (1 < 2):
             print('Sim!')
```

```
Sim!
```

```
In [21]: if (1 < 2):
             print('Primeiro')
         else:
             print('Último')
```

```
Primeiro
```

```
In [19]: if (1 > 2) and (5 <4):
             print('Primeiro')
         else:
             print('Último')
```

```
Último
```

```
In [22]: if (1 == 2):
             print('Primeiro')
         elif (3 == 3):
             print('Meio')
         else:
             print('Último')
```

```
Meio
```

Laços For

```
: seq = [1,2,3,4,5]
```

```
: for item in seq:
      print(item)
```

```
1
2
3
4
5
```

```
: for item in seq:
      print('!')
```

```
!
!
!
!
!
```

```
a = list(range(5, 10))
```

```
print(a)
```

```
[5, 6, 7, 8, 9]
```

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```
list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Laços While

```
i = 1
while i < 5:
    print('i = {}'.format(i))
    i = i+1
```

```
i = 1
i = 2
i = 3
i = 4
```

谢Perguntas?

professora@lucilia.com.br

Créditos

- Template: SlidesCarnival
- Ilustração de fundo: Alex Monge
- PythonBrasil
- Nerdologia