Aula 5

Linguagem de programação

Prof. Wellington Rodrigo Monteiro

1

Conversa Inicial

2

Trabalhando com mais de um valor ao mesmo tempo

- **Iteradores**
  - **Geradores**
- **Combinatórios**
  - **Combinações**
  - **Permutações**
- **Sobrecarga de operadores**

3

Iteradores

4

Iteração

- **Repetição em objetos que podem ser repetidos**
  - **Listas**
  - **Dicionários**
  - **Matrizes**
  - ***Data frames***
  - ***Strings***
- **Loops**
  - **Analogia: assistir a um vídeo de cada vez**

5

Iteração

0	1	1	2	3	5	8	13	21	34	55

Aula 1	Aula 2	Aula 3	Aula 4	Aula 5	Aula 6

P	Y	T	H	O	N

6

- Iteradores finitos
- Iteradores infinitos
  - Count(0, 2): 0, 2, 4, 6, 8, 10, ...
  - Cycle([0, 1, 2]): 0, 1, 2, 0, 1, 2, 0, 1, ...
  - Repeat([0, 1, 2]): [0, 1, 2], [0, 1, 2], ...

7

Combinatórios

8

Combinações

- Combinações
  - Selecionar um conjunto de elementos a partir de um conjunto maior quando a ordem não importa
  - Escolher/selecionar elementos

9

Permutações

10

Permutações

- Permutações
  - Selecionar um conjunto de elementos a partir de um conjunto maior quando a ordem importa
  - Escolher a ordem/agendamento/arranjo dos elementos

11

Geradores

12

- **Funções que não retornam valores, mas um *iterator***
- ***Lazy loading***

13

Geradores

```
1  def gerar_recibos(limite):
2      contador = 1
3      recibos = []
4
5      while contador < limite:
6          recibos.append(f'Recibo {contador}')
7          contador += 1
8
9      # após criar os recibos retornamo-los
10     return recibos
11
12 # a linha abaixo dará um estouro de memória em vários computadores
13 lista_recibos = gerar_recibos(10000000000)
14 count = 0
15 for recibo in lista_recibos:
16     display(recibo)
17
18     # incrementando o contador
19     count += 1
20     # parando de mostrar os recibos se já mostramos 10
21     if count >= 10:
22         break
```

14

```
1  def gerar_recibos_generator(limite):
2      contador = 1
3
4      while contador < limite:
5          yield f'Recibo {contador}'
6          contador += 1
7
8  lista_recibos = gerar_recibos_generator(100000000000)
9
10 count = 0
11 for recibo in lista_recibos:
12     display(recibo)
13
14     # incrementando o contador
15     count += 1
16     # parando de mostrar os recibos se já mostramos 10
17     if count >= 10:
18         break
```

15

Sobrecarga de operadores

16

Operadores

- **+, -, *, @, /, //, %, **, <<, >>, &, ^, |**
- **+=, -=, *=, @=, /=, //=, %=, **=**
- **<<=, >>=, &=, ^=, |=**
- **divmod(), pow()**

17

Sobrecarga

- **1 + 5 = 6**
- **1 / 4 = 0.25**
- **"Maçã" + "Batata" = ?**

18

- **1 + 5 = 6**
- **1 / 4 = 0.25**
- **"Maçã" + "Batata" = "MaçãBatata"**
- **E para nossas classes e objetos?**

19

Soma de cachorros?

```
1  class Cachorro():
2      def __init__(self, nome=None, idade=0):
3          self.__nome = nome
4          self.__idade = idade
5
6      def ler_idade(self):
7          return self.__idade
8
9
10 pippoca = Cachorro('Pippoca', 9)
11 kora = Cachorro('Kora', 1)
12
13 pippoca + kora
```

20

Sobrecarga do operador "+"

```
1  class Cachorro():
2      def __init__(self, nome=None, idade=0):
3          self.__nome = nome
4          self.__idade = idade
5
6      def ler_idade(self):
7          return self.__idade
8
9      # implementando a sobrecarga de operadores
10     def __add__(self, outro_cachorro):
11         return self.__idade + outro_cachorro.__idade
12
13 pippoca = Cachorro('Pippoca', 9)
14 kora = Cachorro('Kora', 1)
15
16 pippoca + kora
```

21