intermediate_python_samples

October 5, 2022

1 F-string formatting

[1]:
```
name = 'Bobert'
greeting = f"Hello {name}"

print(greeting)
```

```
Hello Bobert
```

2 Lambda functions

[3]:
```
def square(x):
    return x**2

lamSquare = lambda x : x**2

lamSquare(5)
```

[3]:
```
25
```

3 Applying lambdas to pandas

[8]:
```
import seaborn as sns
df = sns.load_dataset('iris')

petal_length_cutter = lambda x: x['petal_length'] > df['petal_length'].quantile(.
 ↪90)

to_remove = df.apply(petal_length_cutter, axis=1)

df[to_remove]
```

[8]:
```
     sepal_length  sepal_width  petal_length  petal_width    species
100           6.3          3.3           6.0          2.5  virginica
102           7.1          3.0           5.9          2.1  virginica
105           7.6          3.0           6.6          2.1  virginica
```

```
107           7.3          2.9           6.3          1.8  virginica
109           7.2          3.6           6.1          2.5  virginica
117           7.7          3.8           6.7          2.2  virginica
118           7.7          2.6           6.9          2.3  virginica
122           7.7          2.8           6.7          2.0  virginica
125           7.2          3.2           6.0          1.8  virginica
130           7.4          2.8           6.1          1.9  virginica
131           7.9          3.8           6.4          2.0  virginica
135           7.7          3.0           6.1          2.3  virginica
143           6.8          3.2           5.9          2.3  virginica
```

[18]:
```
# making an even more complicated example...

petal_length_cutter = lambda x: x['petal_length'] > df['petal_length'].quantile(.
↪90)
sepal_length_cutter = lambda x: x['sepal_length'] > df['sepal_length'].quantile(.
↪95)

twof = lambda x: (petal_length_cutter(x) and sepal_length_cutter(x))

to_remove_both = df.apply(twof, axis=1)

df[to_remove_both]
```

[18]:
```
     sepal_length  sepal_width  petal_length  petal_width    species
105           7.6          3.0           6.6          2.1  virginica
107           7.3          2.9           6.3          1.8  virginica
117           7.7          3.8           6.7          2.2  virginica
118           7.7          2.6           6.9          2.3  virginica
122           7.7          2.8           6.7          2.0  virginica
130           7.4          2.8           6.1          1.9  virginica
131           7.9          3.8           6.4          2.0  virginica
135           7.7          3.0           6.1          2.3  virginica
```

4 Zipping lists

[15]:
```
names = ['bob', 'bobert', 'bobby']
ages = [10, 20, 60]

for name, age in zip(names, ages):
    print(name, age)
```

```
bob 10
bobert 20
bobby 60
```

5 Map functions

```
[11]: import numpy as np
      square = lambda x: x**2

      values = np.arange(1, 10)

      *map(square, values),
```

```
[11]: (1, 4, 9, 16, 25, 36, 49, 64, 81)
```

6 Filter functions

```
[13]: def filter_odd_numbers(num):

          if num % 2 == 0:
              return True
          else:
              return False

      *filter(filter_odd_numbers, values),
```

```
[13]: (2, 4, 6, 8)
```

7 Iterators / Generators]

```
[20]: colors = ['red', 'blue', 'green']

      c = iter(colors)

      for i in range(3):
          print(next(c))
```

```
red
blue
green
```

```
[21]: def generator_example(x):
          for i in x:
              yield i

      X = np.arange(1,10)

      generator_example(X)
```

```
[21]: <generator object generator_example at 0x7f8f83bc7d60>
```

```
[22]: y = generator_example(X)

      for i in y:
          print(i)
```

```
1
2
3
4
5
6
7
8
9
```

```
[36]: def fib(n):
          a, b = 0, 1
          for _ in range(n):
              yield a
              a, b = b, a + b
      # These functions are not computed until needed, so we can ask for silly numbers
      fib_n = fib(1000000000)
```

```
[37]: for i in range(10):
          a = next(fib_n)
          print(a)
```

```
0
1
1
2
3
5
8
13
21
34
```

8 List comprehensions

```
[38]: # squish for loops

      # this takes too long
      values = []
      for i in range(10):
```

```python
    values.append(i)

# this is great
values = [i for i in range(10)]
print(values)
```

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