Python小练习：进度条

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

用Python两种方式实现进度条功能，一种是自定义的进度条，一种是调用tqdm库来实现。

1. process_test.py

```python
# -*- coding: utf-8 -*-
# Author：凯鲁嘎吉 Coral Gajic
# https://www.cnblogs.com/kailugaji/
# Python小练习：进度条
import time
import math
from tqdm import tqdm

# 字符串左对齐右对齐
def format_for_process(params):
    new_params = []
    for k, v in params.items():
        k = k.rjust(18)  # 右对齐
        # rjust() 方法是向字符串的左侧填充指定字符，从而达到右对齐文本的目的。
        v = '{:<12}'.format(round(v, 6))[:12]  # 左对齐，并且保留6位小数
        new_params.append([k, v])
    return new_params

class Progress:
# 用于在控制台中显示进度条
    def __init__(self, total, name='Progress', ncol=3, max_length=30, indent=0, line_width=100, speed_update_freq=50):
        self.total = total # 进度总量
        self.name = name # 进度条的名称
        self.ncol = ncol  # 3 # 进度条的列数
        self.max_length = max_length  # 30 进度条的最大长度
        self.indent = indent  # 0 进度条的缩进量
        self.line_width = line_width  # 100 进度条的宽度
        self._speed_update_freq = speed_update_freq  # 50 进度条的速度更新频率
        self._step = 0 # 进度条的当前步数
        self._prev_line = '\033[F' # 上一次进度条的内容
        self._clear_line = ' ' * self.line_width  # 100, ''中间空100个字符 清空进度条的内容
        # '                                                                                                    '
        self._pbar_size = self.ncol * self.max_length  # 3*30 进度条的分组大小
        self._complete_pbar = '#' * self._pbar_size  # # * 90,90个# 完成进度条的内容
        # ############################################################################################
        self._incomplete_pbar = ' ' * self._pbar_size # 未完成进度条的内容
        # '                                                                                            '
        self.lines = [''] # 进度条的内容列表
        self.fraction = '{} / {}'.format(0, self.total) # 进度条的进度
        self.resume() # 重新开始进度条的计时

    def update(self, n=1):
        self._step += n
        if self._step % self._speed_update_freq == 0:
            self._time0 = time.time()
            self._step0 = self._step
    # 该方法的作用是更新步数_step并在每次更新后打印当前时间_time0和步数_step0。
    # 如果_step的值是_speed_update_freq的倍数，则打印一个换行符，并将_time0和_step0设置为当前时间和步数。
    # 如果_step的值不是_speed_update_freq的倍数，则不打印任何内容，并将_time0和_step0设置为当前时间和步数。
    # 如果_step的值是0，则打印一个换行符，并将_time0和_step0设置为当前时间和步数。

    def resume(self):
        self._skip_lines = 1
        print('\n', end='')
        self._time0 = time.time()
        self._step0 = self._step
    # self._skip_lines是一个实例变量，其值为1表示正在执行代码块，为0表示已经执行完毕。
```

```python
    # 该方法的作用是在每次更新后打印一个换行符，并将_time0和_step0设置为当前时间和步数。
    # 如果self._skip_lines的值是1，则在每行代码末尾打印一个换行符。

    def pause(self):
        self._clear()
        self._skip_lines = 1

    def set_description(self, params=[]):
        # Position
        self._clear()
        # Percent
        percent, fraction = self._format_percent(self._step, self.total)
        self.fraction = fraction
        # Speed
        speed = self._format_speed(self._step)
        # Params
        num_params = len(params)
        nrow = math.ceil(num_params / self.ncol)  # 返回大于等于参数x的最小整数,即对浮点数向上取整
        params_split = self._chunk(params, self.ncol)
        params_string, lines = self._format(params_split)
        self.lines = lines

        description = '{} | {}{}'.format(percent, speed, params_string)
        print(description)
        self._skip_lines = nrow + 1

    def set_description2(self, params=[]):
        # Params
        num_params = len(params)
        nrow = math.ceil(num_params / self.ncol)  # 返回大于等于参数x的最小整数,即对浮点数向上取整
        params_split = self._chunk(params, self.ncol)
        params_string, lines = self._format(params_split)
        print('\n', params_string)

    def _clear(self):
        position = self._prev_line * self._skip_lines
        empty = '\n'.join([self._clear_line for _ in range(self._skip_lines)])
        print(position, end='')
        print(empty)
        print(position, end='')

    def _format_percent(self, n, total):
        if total:
            percent = n / float(total)

            complete_entries = int(percent * self._pbar_size)
            incomplete_entries = self._pbar_size - complete_entries

            pbar = self._complete_pbar[:complete_entries] + self._incomplete_pbar[:incomplete_entries]
            fraction = '{} / {}'.format(n, total)
            string = '{} [{}] {:3d}%'.format(fraction, pbar, int(percent * 100))
        else:
            fraction = '{}'.format(n)
            string = '{} iterations'.format(n)
        return string, fraction
        # string:
        # 9 / 10 [################################################################################         ]  90%
        # fraction: 9 / 10

    # 输出x.x Hz
    def _format_speed(self, n):
        num_steps = n - self._step0
        t = time.time() - self._time0
        speed = num_steps / (t + float("1e-8"))
        string = '{:.1E} Hz'.format(speed)
        if num_steps > 0:
            self._speed = string
        return string

    def _chunk(self, l, n):
        return [l[i:i + n] for i in range(0, len(l), n)]
```

```
129
130     def _format(self, chunks):
131         lines = [self._format_chunk(chunk) for chunk in chunks]
132         lines.insert(0, '')
133         padding = '\n' + ' ' * self.indent
134         string = padding.join(lines)
135         return string, lines
136
137     def _format_chunk(self, chunk):
138         line = ' | '.join([self._format_param(param) for param in chunk])
139         return line
140
141     def _format_param(self, param):
142         k, v = param
143         return '{} : {}'.format(k, v)[:self.max_length]
144
145     def stamp(self):
146         if self.lines != ['']:
147             params = ' | '.join(self.lines)
148             string = '[ {} ] {}{} | {}'.format(self.name, self.fraction, params, self._speed)
149             # name: Process
150             # fraction: 6 / 6
151             # 输出:
152             # [ Progress ] 6 / 6 |            weight : 149.8228  |            height : 214.32345 |            repeat : 45.8241   |            alpha : 43.32475  |            beta : 50.8254   |            gamma
153             self._clear()
154             print(string)
155             self._skip_lines = 1
156         else:
157             self._clear()
158             self._skip_lines = 0
159
160     def close(self):
161         self.pause()
162
163 total_num = 6
164 # 自己实现的进度条功能
165 progress = Progress(total_num)
166 for i in range(total_num):
167     progress.update()
168     params = {"weight": 127.0 - (i+1.5)*3.5112,
169               "height": 185.0 - (i+1.5)*4.5113,
170               "repeat": 10.0 - (i+1.5)*5.5114,
171               "alpha": 1.0 + (i+1.5)*6.5115,
172               "beta": 2.0 + (i+1.5)*7.5116,
173               "gamma": 5.0 + (i+1.5)*8.5117}
174     progress.set_description(format_for_process(params))
175 progress.stamp()
176 progress.close()
177 print('--------------------------------------------------------------------------')
178 # Python自带的进度条包
179 for i in tqdm(range(total_num)):
180      time.sleep(0.1)
181      params = {"weight": 127.0 - (i + 1.5) * 3.5112,
182                "height": 185.0 - (i + 1.5) * 4.5113,
183                "repeat": 10.0 - (i + 1.5) * 5.5114,
184                "alpha": 1.0 + (i + 1.5) * 6.5115,
185                "beta": 2.0 + (i + 1.5) * 7.5116,
186                "gamma": 5.0 + (i + 1.5) * 8.5117}
187      progress.set_description2(format_for_process(params))
188 progress.stamp()
189 progress.close()
```

2. 结果

```
D:\ProgramData\Anaconda3\python.exe "D:/Python code/2023.3 exercise/time/process_test.py"


1 / 6 [###############                                                          ]  16% | 1.0E+08 Hz
          weight : 121.7332  |            height : 178.23305 |            repeat : 1.7329
```

```
            alpha : 10.76725  |              beta : 13.2674   |             gamma : 17.76755


2 / 6 [##############################                                                  ]  33% | 2.0E+08 Hz
            weight : 118.222  |            height : 173.72175 |            repeat : -3.7785
             alpha : 17.27875 |              beta : 20.779    |             gamma : 26.27925


3 / 6 [###############################################                                 ]  50% | 3.0E+08 Hz
            weight : 114.7108 |            height : 169.21045 |            repeat : -9.2899
             alpha : 23.79025 |              beta : 28.2906   |             gamma : 34.79095


4 / 6 [############################################################                    ]  66% | 4.0E+08 Hz
            weight : 111.1996 |            height : 164.69915 |            repeat : -14.8013
             alpha : 30.30175 |              beta : 35.8022   |             gamma : 43.30265


5 / 6 [#####################################################################           ]  83% | 5.0E+03 Hz
            weight : 107.6884 |            height : 160.18785 |            repeat : -20.3127
             alpha : 36.81325 |              beta : 43.3138   |             gamma : 51.81435


6 / 6 [##############################################################################################] 100% | 6.0E+03 Hz
            weight : 104.1772 |            height : 155.67655 |            repeat : -25.8241
             alpha : 43.32475 |              beta : 50.8254   |             gamma : 60.32605


[ Progress ] 6 / 6 |            weight : 104.1772  |            height : 155.67655 |            repeat : -25.8241  |            alpha : 43.32475  |              beta : 50.8254   |             gamma : 60.32605  | 6.0E

------------------------------------------------------------------------
  0%|          | 0/6 [00:00<?, ?it/s]
 17%|██          | 1/6 [00:00<00:00,  9.95it/s]
            weight : 121.7332 |            height : 178.23305 |            repeat : 1.7329
             alpha : 10.76725 |              beta : 13.2674   |             gamma : 17.76755

 33%|████▏       | 2/6 [00:00<00:00,  9.63it/s]
            weight : 118.222  |            height : 173.72175 |            repeat : -3.7785
             alpha : 17.27875 |              beta : 20.779    |             gamma : 26.27925


            weight : 114.7108 |            height : 169.21045 |            repeat : -9.2899
             alpha : 23.79025 |              beta : 28.2906   |             gamma : 34.79095
 50%|██████      | 3/6 [00:00<00:00,  9.46it/s]
 67%|████████▎   | 4/6 [00:00<00:00,  9.33it/s]
            weight : 111.1996 |            height : 164.69915 |            repeat : -14.8013
             alpha : 30.30175 |              beta : 35.8022   |             gamma : 43.30265

 83%|██████████▏ | 5/6 [00:00<00:00,  9.28it/s]
            weight : 107.6884 |            height : 160.18785 |            repeat : -20.3127
             alpha : 36.81325 |              beta : 43.3138   |             gamma : 51.81435

100%|████████████| 6/6 [00:00<00:00,  9.34it/s]

            weight : 104.1772 |            height : 155.67655 |            repeat : -25.8241
             alpha : 43.32475 |              beta : 50.8254   |             gamma : 60.32605

[ Progress ] 6 / 6 |            weight : 104.1772  |            height : 155.67655 |            repeat : -25.8241  |            alpha : 43.32475  |              beta : 50.8254   |             gamma : 60.32605  | 6.0E

Process finished with exit code 0
```

完成。