极验滑动验证码破解

最终代码：

`easing.py`

`run.py`

评测结果

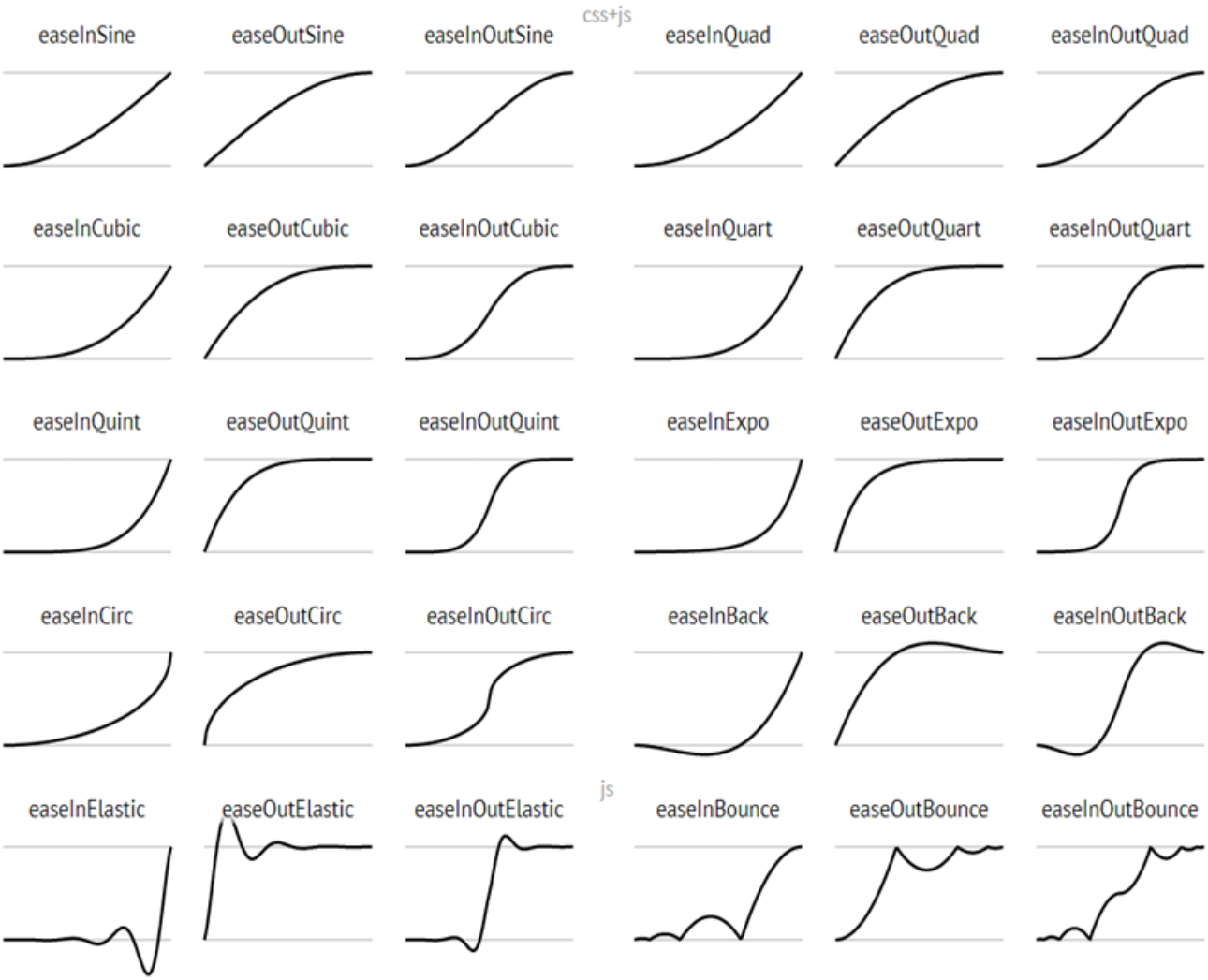
	6	8	10	12
ease_in_quad	41%/23%	72%/72%	62%/60%	32%/26%
ease_out_quad	72%/96%	88%/92%	77%/80%	50%/72%
ease_in_out_quad	92%/94%	95%/91%	87%/91%	93%/92%
ease_in_cubic	12%/16%	48%/41%	69%/67%	38%/54%
ease_out_cubic	98%/99%	94%/95%	83%/83%	62%/68%
ease_in_out_cubic	99%/98%	98%/66%	85%/87%	64%/50%
ease_in_quart【X】	14%/13%	X	43%/53%	52%/59%
ease_out_quart	99%/100%	94%/96%	83%/80%	55%/62%
ease_in_out_quart	92%/94%	94%/98%	85%/80%	76%/75%
ease_in_quint【X】	5%/5%	X	27%/28%	22%/23%
ease_out_quint	99%/95%	96%/98%	80%/64%	45%/51%
ease_in_out_quint	98%/88%	87%/93%	87%/81%	28%/41%
ease_in_sine	51%/55%	51%/50%	61%/51%	31%/35%
ease_out_sine	98%/94%	92%/93%	81%/88%	50%/69%
ease_in_out_sine	90%/93%	97%/97%	91%/87%	84%/88%
ease_in_expo【X】	0%/0%	X	21%/20%	47%/46%
ease_out_expo	100%/100%	100%/99%	83%/89%	61%/75%
ease_in_out_expo	98%/98%	97%/96%	93%/87%	92%/90%
ease_in_cric【X】	X	X	X	X
ease_out_cric	100%/99%	100%/94%	88%/89%	75%/79%
ease_in_out_cric	97%/100%	99%/99%	90%/93%	96%/97%
ease_in_elastic【X】	X	X	X	X
ease_out_elastic	98%/99%	95%/83%	56%/60%	9%/17%
ease_in_out_elastic	92%/92%	68%/57%	54%/68%	72%/78%
ease_in_back【X】	3%/2%	28%/24%	54%/68%	34%/33%
ease_out_back	96%/98%	94%/93%	73%/74%	59%/62%
ease_in_out_back	95%/94%	87%/89%	81%/81%	85%/82%
ease_in_bounce【X】	46%/48%	27%/25%	29%/20%	23%/28%

	6	8	10	12
ease_out_bounce	100%/99%	67%/76%	39%/42%	91%/92%
ease_in_out_bounce	73%/57%	67%/70%	16%/23%	19%/13%

总体上看，参数为6和8的正确率较10和12高，于是将6和8下的准确率作为评判标准。

标红的是在6和8下准确率超过了85%，甚至能达到95%~100%的轨迹（共16个）

附轨迹图片和轨迹源码网址：https://github.com/gdsmith/jquery.easing/blob/master/jquery.easing.js




```
# -*- coding: utf-8 -*-
# easing.py

import numpy as np
import math

c1 = 1.70158
c2 = c1 * 1.525
c3 = c1 + 1
c4 = ( 2 * math.pi ) / 3
```

```python
c5 = ( 2 * math.pi ) / 4.5

def ease_in_quad(x):
    return x * x

def ease_out_quad(x):
    return 1 - (1 - x) * (1 - x)

def ease_in_out_quad(x):
    if(x<0.5):
        return 2*x*x
    else:
        return 1-pow(-2 * x +2 , 2) / 2

#-----------------------------------------------

def ease_in_cubic(x):
    return x * x * x

def ease_out_cubic(x):
    return 1 - pow( 1-x, 3)

def ease_in_out_cubic(x):
    if x < 0.5:
        return 4 * x * x * x
    else:
        return 1 - pow( -2 * x + 2, 3 ) / 2

#-----------------------------------------------

def ease_in_quart(x):
    return x * x * x * x

def ease_out_quart(x):
    return 1 - pow(1 - x, 4)

def ease_in_out_quart(x):
    if x < 0.5:
        return 8 * x * x * x * x
    else:
        return 1 - pow( -2 * x + 2, 4 ) / 2

#-----------------------------------------------

def ease_in_quint(x):
    return  x * x * x * x * x

def ease_out_quint(x):
    return 1 - pow( 1 - x, 5 )

def ease_in_out_quint(x):
    if x < 0.5:
        return 16 * x * x * x * x * x
```

```
    else:
        return 1 - pow( -2 * x + 2, 5 ) / 2

#-----------------------------------------------

def ease_in_sine(x):
    return 1 - math.cos( x * math.pi/2 )

def ease_out_sine(x):
    return math.sin( x * math.pi/2 )

def ease_in_out_sine(x):
    return -( math.cos( math.pi * x ) - 1 ) / 2

#-----------------------------------------------

def ease_in_expo(x):
    if x==0:
        return 0
    else:
        return pow( 2, 10 * x - 10)

def ease_out_expo(x):
    if x == 1:
        return 1
    else:
        return 1 - pow(2, -10 * x)

def ease_in_out_expo(x):
    if x==0:
        return 0
    else:
        if x==1:
            return 1
        else:
            if x<0.5:
                return pow( 2, 20 * x - 10 ) / 2
            else:
                return ( 2 - pow( 2, -20 * x + 10 ) ) / 2

#-----------------------------------------------------------

def ease_in_cric(x):
    return 1 - math.sqrt( 1 - pow( x, 2 ) )

def ease_out_cric(x):
    return math.sqrt( 1 - pow( x - 1, 2 ) )

def ease_in_out_cric(x):
    if x < 0.5:
        return ( 1 - math.sqrt( 1 - pow( 2 * x, 2 ) ) ) / 2
    else:
        return ( math.sqrt( 1 - pow( -2 * x + 2, 2 ) ) + 1 ) / 2
```

```
#-----------------------------------------------------------------

def ease_in_bounce(x):
    return 1 - ease_out_bounce( 1-x )

def ease_out_bounce(x):
    n1 = 7.5625
    d1 = 2.75
    if x < 1 / d1 :
        return n1 * x * x
    elif x < 2 / d1:
        x -= 1.5 / d1
        return n1 * x*x + 0.75
    elif x < 2.5 / d1:
        x -= 2.25 / d1
        return n1 * x*x + 0.9375
    else:
        x -= 2.625 / d1
        return n1 * x*x + 0.984375

def ease_in_out_bounce(x):
    if x<0.5:
        return (1 - ease_out_bounce(1- 2 * x ) ) / 2
    else:
        return (1 + ease_out_bounce(2 * x -1 ) ) / 2
#-----------------------------------------------

def ease_in_elastic(x):
    if x==0:
        return 0
    else:
        if x==1:
            return 1
        else:
            return -pow( 2, 10 * x - 10 ) * math.sin( ( x * 10 - 10.75 ) * c4 )

def ease_out_elastic(x):
    if x == 0:
        return 0
    elif x == 1:
        return 1
    else:
        return pow(2, -10 * x) * math.sin((x * 10 - 0.75) * c4) + 1

def ease_in_out_elastic(x):
    if x==0:
        return 0
    else:
        if x==1:
            return 1
        else:
            if x <0.5:
```

```
                return -( pow( 2, 20 * x - 10 ) * math.sin( ( 20 * x - 11.125 ) * c5 )) / 2
            else:
                return pow( 2, -20 * x + 10 ) * math.sin( ( 20 * x - 11.125 ) * c5 ) / 2 +
1

#---------------------------------------------------------------

def ease_in_back(x):
    return c3 * x * x * x - c1 * x * x

def ease_out_back(x):
    return 1 + c3 * pow( x - 1, 3 ) + c1 * pow( x - 1, 2 )

def ease_in_out_back(x):
    if x<0.5:
        return ( pow( 2 * x, 2 ) * ( ( c2 + 1 ) * 2 * x - c2 ) ) / 2
    else:
        return ( pow( 2 * x - 2, 2 ) *( ( c2 + 1 ) * ( x * 2 - 2 ) + c2 ) + 2 ) / 2

#---------------------------------------------------------------


def get_tracks(distance, seconds, ease_func):
    tracks = [0]
    offsets = [0]
    for t in np.arange(0.0, seconds, 0.1):
        ease = globals()[ease_func]
        offset = round(ease(t/seconds) * distance)
        tracks.append(offset - offsets[-1])
        offsets.append(offset)

    return offsets, tracks
```

```
# -*- coding: utf-8 -*-
# run.py

import random
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
import time
import os
from selenium import webdriver
from PIL import Image #获取图像RGB值
import easing

class CrackGreetest():
    def __init__(self):
        self.url = "https://auth.geetest.com/login/"
        self.browser = webdriver.Chrome(executable_path =
'D:\\2019Download\\ChromeDriver\\chromedriver.exe')
```

```
        self.wait = WebDriverWait(self.browser,20)
        self.email = 'youremail@email.com'

    def open(self):
        """
        打开网页输入用户名密码
        :return:
        """
        self.browser.get(self.url)
        email =
self.wait.until(expected_conditions.presence_of_element_located((By.CLASS_NAME,'ivu-
input')))
        # password =
self.wait.until(expected_conditions.presence_of_element_located((By.ID,'password')))
        email.send_keys(self.email)
        # password.send_keys(self.password)

    def get_geetest_button(self):
        '''
        获取初始验证码按钮
        :return: 按钮对象
        '''
        button =
self.wait.until(expected_conditions.element_to_be_clickable((By.CLASS_NAME,'geetest_radar_t
ip')))
        return button

    def get_unfull_image(self, path):
        """
        获取未完整验证码图片
        :param name:
        :return: 图片对象
        """
        element = self.browser.find_element_by_css_selector(
            "body > div.geetest_fullpage_click.geetest_float.geetest_wind.geetest_slide3 >
div.geetest_fullpage_click_wrap > div.geetest_fullpage_click_box > div > div.geetest_wrap >
div.geetest_widget > div > a > div.geetest_canvas_img.geetest_absolute > div >
canvas.geetest_canvas_slice.geetest_absolute")

        unfull_captcha = element.screenshot(path)
        return unfull_captcha

    def get_full_image(self,path):
        '''
        获取完整验证码图片
        :param name:
        :return:
        '''
        # 这里要执行JavaScript脚本才能拿到完整图片的截图
        show_Full_img1 = "document.getElementsByClassName('geetest_canvas_fullbg')
[0].style.display='block'"
        self.browser.execute_script(show_Full_img1)
```

```
        show_Full_img2 = "document.getElementsByClassName('geetest_canvas_fullbg')
[0].style.opacity=1"
        self.browser.execute_script(show_Full_img2)
        # 等待完整图片加载
        time.sleep(2)
        element = self.browser.find_element_by_css_selector(
            "body > div.geetest_fullpage_click.geetest_float.geetest_wind.geetest_slide3 >
div.geetest_fullpage_click_wrap > div.geetest_fullpage_click_box > div > div.geetest_wrap >
div.geetest_widget > div > a > div.geetest_canvas_img.geetest_absolute > canvas")
        full_captcha = element.screenshot(path)
        return full_captcha

    def get_slider(self):
        """
        获取滑块
        :return: 滑块对象
        """
        slider =
self.wait.until(expected_conditions.element_to_be_clickable((By.CLASS_NAME,'geetest_slider_
button')))
        return slider

    def get_gap_left(self,image2):
        """
        获取滑块左端位置及滑块大小
        :param image2: 带缺口的图片
        :return: 滑块左端位置，滑块大小
        """
        st_black = 0
        st_yellow=0
        end_black=0
        end_yellow=0
        for i in range(0,image2.size[0]):
            #width
            ans=0
            for j in range(image2.size[1]):
                #height
                rgb = image2.load()[i, j]
                if  ( rgb[0]<120 and rgb[1] < 120 and rgb[2] <120):
                    ans=ans+1
                if(ans>10):
                    st_black = i
                    break
            if st_black>0:
                break

        for i in range(st_black,image2.size[0]):
            ans=0
            for j in range(image2.size[1]):
                #height
                rgb = image2.load()[i, j]
                if  ( rgb[0]>180 and rgb[1] > 180 and rgb[2] <200):
                    ans=ans+1
```

```
                if(ans>20):
                    st_yellow=i
                    break
            if st_yellow>0:
                break

        return st_yellow

        #暂时弃置
        for i in range(st_yellow+45,0,-1):
            #width
            ans=0
            for j in range(image2.size[1]):
                #height
                rgb = image2.load()[i, j]
                if  ( rgb[0]<120 and rgb[1] < 120 and rgb[2] <120):
                    ans=ans+1
                if(ans>15):
                    end_black = i
                    break
            if end_black>0:
                break

        for i in range(end_black,0,-1):
            ans=0
            for j in range(image2.size[1]):
                #height
                rgb = image2.load()[i, j]
                if  ( rgb[0]>165 and rgb[1] >165 and rgb[2] <200):
                    ans=ans+1
                if(ans>20):
                    end_yellow = i
                    break
            if end_yellow>0:
                break

        return st_yellow,end_yellow-st_yellow

    def get_gap_right(self,image1,image2):
        """
        获取缺口左端位置
        :param image1: 不带缺口的图片
        :param image2: 带缺口的图片
        :return: 像素是否相同
        """
        # 缺口在滑块右侧，设定遍历初始横坐标left为60
        left = 60
        # 像素对比阈值
        threshold = 50
        for i in range(image1.size[0]-1, left, -1):
            #width
            ans=0
            for j in range(image1.size[1]):
```

```
                #height
                rgb1 = image1.load()[i, j]
                rgb2 = image2.load()[i, j]

                res1 = abs(rgb2[0] - rgb1[0])
                res2 = abs(rgb2[1] - rgb1[1])
                res3 = abs(rgb2[2] - rgb1[2])
                if  (res1 > threshold or res2 > threshold or res3 > threshold):
                    ans=ans+1
                if(ans>15):
                    return i-41  # 返回缺口偏移距离，这里需测试几次


    def move_to_grap(self,slider,distance,paring):
        '''
        拖动滑块到缺口处
        :param slider: 滑块
        :param track: 轨迹
        :return:
        '''
        ans=random.randint(3,4)
        offsets, track = easing.get_tracks(distance, ans*2, paring)
        # print(offsets)
        # 调用ActionChains的click_and_hold()方法按住拖动底部滑块，遍历运动轨迹获取每小段位置距离，调用
move_by_offset()方法移动此位移，最后调用release()方法松开鼠标
        ActionChains(self.browser).click_and_hold(slider).perform()
        for x in track:
            ActionChains(self.browser).move_by_offset(xoffset=x,yoffset=0).perform()
            time.sleep(0.01)
        time.sleep(0.3)
        ActionChains(self.browser).release().perform()

    def crack(self,paring):
        try:
            # 输入用户名
            self.open()

            # 模拟点击按钮
            button = self.get_geetest_button()
            button.click()
            time.sleep(2)

            tn=time.localtime(time.time())

            # 获取带缺口验证码图片
            unfull_path =
'D:\Documents\Desktop\TempHandle/ans/'+str(tn.tm_mday)+'.'+str(tn.tm_min)+'.'+str(tn.tm_sec
)+'pic1.png'
            self.get_unfull_image(unfull_path)
            image1 = Image.open(unfull_path)

            # 获取完整的验证码图片
```

```
            full_path =
'D:\Documents\Desktop\TempHandle/ans/'+str(tn.tm_mday)+'.'+str(tn.tm_min)+'.'+str(tn.tm_sec
)+'pic2.png'
            self.get_full_image(full_path)
            image2 = Image.open(full_path)
            #
            # 对比两张图片像素点，获取缺口位置，得到偏移距离
            # 获取缺口位置
            st=self.get_gap_left(image1)
            ed=self.get_gap_right(image1,image2)
            distance = ed-st
            # print("缺口距离",distance)

            # # 获取移动轨迹
            # track = self.get_track(distance)
            # print("滑动轨迹",track)

            # 模拟人的行为，拖动滑块，完成验证
            slider = self.get_slider()
            # slider.click()

            # 拖动滑块
            self.move_to_grap(slider, distance, paring)
            time.sleep(1)

 text=self.browser.find_element_by_class_name('geetest_result_title').get_attribute('textCo
ntent')

            return
            # if(text=='哇哦~怪物吃了拼图请 3 秒后重试'):
            #     res[1]+=1
            #     return
            # elif(text=='拖动滑块将悬浮图像正确拼合'):
            #     res[2]+=1
            #     return
            # else:
            #     #text=='sec 秒的速度超过 score% 的用户'
            #     res[0]+=1
            #     return
        except:
            return

if __name__ == "__main__":
    crack = CrackGreetest()

    parameter=['ease_in_out_quad','ease_out_cubic','ease_in_out_cubic','ease_out_quart',
               'ease_in_out_quart','ease_out_quint','ease_in_out_quint','ease_out_sine',
               'ease_in_out_sine','ease_out_expo','ease_in_out_expo','ease_out_cric',
               'ease_in_out_cric','ease_out_elastic','ease_out_back','ease_in_out_back']

    while(True):
        ans=random.randint(0,15)
        crack.crack( parameter[ans])
```