Омский Научный центр Сибирского отделения Российской академии наук

Региональная общественная организация «Омский совет ректоров»

Омское региональное отделение Всероссийской общественной организации «Русское географическое общество»

Детская областная общественная организация «Научное общество учащихся «Поиск»

Бюджетное общеобразовательное учреждение Полтавского муниципального района «Полтавский лицей»

LI
Межрегиональная научно-практическая конференция школьников и учащейся молодежи

Тема: «Исследование возможности реализации игры Брейн-ринг с использованием библиотеки Pygame на Python 3»

Учебно-исследовательская работа
Научное направление: информатика 5-9 классы

Выполнил:
ученик 8 класса
БОУ «Полтавский лицей»
Кондратенко Данила Александрович

Научный руководитель:
БОУ «Полтавский лицей»
Панченко Светлана Викторовна

Омск - 2019

Оглавление

Введение

Целеполагание

Цель работы: Создать программу для проведения игры «Брейн-ринг».

Задачи:

1. Изучить язык *Python 3* и библиотеку *Pygame*.
2. Написать рабочую программу с обеспечением визуализации процесса игры.
3. Разработать техническое оснащение.

Определения

Python — объектно-ориентированный функциональный высокоуровневый язык программирования общего назначения с динамической типизацией, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций, в частности, обработка CSV[1] и JSON[2], статистические функции, сжатие файлов и многое другое. Всего в стандартной библиотеке насчитывается 240 модулей, из которых 46 модулей упакованы в 12 пакетов, 12 модулей связаны со средой исполнения Python 3, 4 модуля признаны устаревшими, а 17 предоставляют специфичные для платформ функции. Он более приспособлен к работе с различными контейнерами (списками, кортежами, множествами, словарями), ведь в цикле for можно перебирать не индексы элементов, а сами элементы.

Язык был создан Гвидо ван Россумом, а первая версия (Python 0.9) вышла 20 февраля 1991. Название языка произошло, вопреки расхожему мифу, от названия юмористической программы 70-х годов «Monty Python's

1 CSV — **C**omma **s**eparated **v**alues (Значения, разделённые запятыми)
2 JSON — **J**ava**s**cript **o**bject **n**otation (Запись объектов согласно Javascript)

flying circus[3]». Позже это название было связано с названием змеи, но ошибочно, ведь в справочной документации можно заметить понятия, связанные с Monty Python или получившие популярность благодаря Monty Python, например, наряду с традиционными названиями foo и bar используются spam и eggs, также «spam» используется в выводимых результатах.

Также язык содержит свою философию, именуемую «Дзен Пайтона» (The Zen of Python), выраженную в 19 предложениях.

Третья версия языка Python вышла 3 декабря 2008 года, получившая множество прогрессивных изменений по сравнению с Python 2, в частности, оператор print стал функцией, появились генераторы, убрано ограничение на размер целого числа в памяти, а кодировка строк была заменена на UTF-8[4].

Pygame (от **Py**thon и **game** - игра) — набор модулей для кроссплатформенной разработки игр. Базируется на мультимедийной библиотеке SDL[5]. Был создан Питом Шиннерсом, первая версия вышла 28 октября 2000 года. Последняя стабильная версия вышла 19 июля 2018 года (Pygame 1.9.4)

Краткое описание игры «Брейн-ринг»

Брейн-ринг — интеллектуальная игра, созданная Владимиром Ворошиловым по образу и подобию «Что? Где? Когда?», только в отличие от неё, одновременно соревнуются между собой две команды. Со временем «Брейн-ринг» стал одним из видов интеллектуального спорта, и по нему стали проводиться целые турниры.

В игре могут принимать участие 2 команды, снабжённых кнопками. Ведущий задаёт вопрос и запускает таймер. Когда команда нажимает кнопку,

3 Летающий цирк Монти Пайтона
4 UTF-8 — **U**nicode **t**ransformation **f**ormat, **8**-bit (Восьмибитный формат трансформации Юникода)
5 SDL — **S**imple **D**irectMedia **l**ayer (Простой слой DirectMedia)

происходит остановка таймера. В случае неправильного ответа ведущий запускает таймер снова, а если та или иная команда даёт правильный ответ, то ведущий обнуляет таймер и добавляет очки команде. Правильно ответившая команда получает одно очко, но если обе команды ответили неправильно, то в случае правильного ответа на следующем вопросе какая-либо команда может получить два очка. Если на два вопроса подряд не было дано правильного ответа, то команда, ответившая на следующий вопрос правильно, может получить три очка. Если же на три вопроса никто не ответил, то обе команды дисквалифицируются.

Когда таймер не запущен, любое нажатие кнопки является фальстартом, а команда, его допустившая, снимается с вопроса.

Процесс реализации

Создание визуализации счёта и таймера

Счёт

```
"""
Срез кода программы Брейн-ринг для счёта
"""
# Импортируем необходимые модули
import sys, pygame
import pygame.mouse

state = True

# Создаём переменные для красной и зелёной команды
score_red, score_green = 0, 0      # Счёт красной и зелёной команды
score_plus = 1                     # Количество прибавляемых очков
is_red = True                      # Какой команде увеличить счёт

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()

# Задаём шрифты
score_font = pygame.font.SysFont(None, 400)          # Шрифт очков
small_font = pygame.font.SysFont('Courier New', 48)  # Шрифт количества
прибавляемых очков
```

```
# Задаём окна
lscreen = pygame.Surface((640, 480)) # левая половина окна (счёт)
screen = pygame.display.set_mode((1280, 480)), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) #
Разрешаем обработку событий мыши и клавиатуры
pygame.mouse.set_visible(False)

r1 = pygame.Rect(310, 230, 20, 20) # добавляем квадрат, показывающий, какой
команде присуждаются баллы

while state:
      lscreen.fill(black) # заполняем левую половину экрана чёрным
      # Вывод квадрата r1, окрашенного в цвет команды, которой нужно присудить
баллы
      if (is_red):
            pygame.draw.rect(lscreen, red, r1)
      else:
            pygame.draw.rect(lscreen, green, r1)
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Переключение команды (Numpad /)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_DIVIDE):
                  is_red = not is_red
            # Добавление очков команде (Numpad +)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and (is_red):
                  score_red += score_plus
                  score_plus = 1
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and not (is_red):
                  score_green += score_plus
                  score_plus = 1
            # Увеличение коэффициента прибавления (стрелки ← →)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_LEFT):
                  score_plus -= 1
                  if (score_plus <= 0):
                        score_plus = 1
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RIGHT):
                  score_plus += 1
            # Обнуление очков (Backspace)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_BACKSPACE):
                  score_red = 0
                  score_green = 0
            # Выход из программы
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                  pygame.display.quit()
                  pygame.font.quit()
                  state = False
                  break
      # Отрисовка
      ## Создаём объекты red_score и green_score, в которых хранится рисунок с
числом, соответствующим счёту команд
```

```
      red_score = score_font.render(("%d" % score_red), True, red)
      green_score = score_font.render(("%d" % score_green), True, green)
      ## Создаём объект plus_score, в котором хранится рисунок с числом,
соответствующим количеству прибавляемых очков
      plus_score = small_font.render(("%d" % score_plus), True, white)
      ## Перерисовываем эти объекты на левую половину окна, центруя их
      lscreen.blit(red_score, [(lscreen.get_width()//2 -
red_score.get_width()) // 2, (lscreen.get_height() -
red_score.get_height()) // 2])
      lscreen.blit(green_score, [((lscreen.get_width()//2 -
green_score.get_width()) // 2)+lscreen.get_width()//2, (lscreen.get_height()
- green_score.get_height()) // 2])
      lscreen.blit(plus_score, [(lscreen.get_width()-
plus_score.get_width())//2, 436])
      ## Перерисовываем левую половину окна
      screen.blit(lscreen, [0, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Таймер

```
"""
Срез кода программы Брейн-ринг для таймера
"""
# Импортируем необходимые модули
import sys, pygame
import pygame.mixer, pygame.mouse
from timer import Timer # код модуля timer в приложении 3

state = True

# Создаём таймер
timer = Timer(60)

current_str = '0.00'

# Задаём параметры
lock = False            # блокировка кнопки
cmd = -1                # номер команды: 1, 3 - красная; 2, 4 - зелёная, 0 -
таймер работает, -1 - таймер стоит

# Инициализируем звуки
pygame.mixer.init()

# Задаём звуковые сигналы
start_snd = pygame.mixer.Sound('timer_start.wav')          # запуск таймера
stop_snd = pygame.mixer.Sound('timer_stop.wav')            # остановка
таймера
end_snd = pygame.mixer.Sound('timer_end.wav')              # истечение
времени
falstart_snd = pygame.mixer.Sound('timer_falstart.wav')    # фальстарт

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()
```

```
# Задаём шрифт таймера
timer_font = pygame.font.SysFont('Courier New', 200)

# Задаём окна
rscreen = pygame.Surface((640, 480)) # правая половина окна (таймер)
screen = pygame.display.set_mode((1280, 480)), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) #
Разрешаем обработку событий мыши и клавиатуры
pygame.mouse.set_visible(False) # Делаем указатель невидимым

while state:
      current_str = '{:.2f}'.format(timer.current()) # Строка,
соответствующая значению таймера
      # Проигрывание сигнала остановки
      if (timer.current() == timer.end) and (timer._started) and (cmd != -1):
            end_snd.play()
            cmd = -1
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Запуск таймера (Enter)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RETURN):
                  cmd = 0                        # Сброс номера команды
                  pygame.mixer.stop()            # Остановка всех звуков
                  lock = False                   # Снятие блокировки
                  start_snd.play()
                  if (not timer.is_paused()):
                        timer.start()            # Запуск таймера
                  else:
                        timer.resume()           # Продолжение работы таймера

            # Остановка и обнуление таймера (Space)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_SPACE):
                  cmd = -1                       # Сброс номера команды
                  pygame.mixer.stop()            # Остановка всех звуков
                  lock = False                   # Снятие блокировки
                  timer.stop()                   # Остановка таймера
                  timer.null()                   # Обнуление таймера
            # Остановка красной командой (правая кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 3)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer.is_started()):
                        cmd = 1                  # Установка номера команды
                        timer.pause()            # Приостановка таймера
                        pygame.mixer.stop()      # Остановка всех звуков
                        stop_snd.play()          # Проигрывание сигнала
остановки
                        lock = True              # Установка блокировки
                  # Фальстарт
                  else:
                        cmd = 3                  # Установка номера команды
                        pygame.mixer.stop()      # Остановка всех звуков
                        falstart_snd.play()      # Проигрывание звука
фальстарта
                        break
            # Остановка зелёной командой (средняя кнопка мыши)
```

```
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 2)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer.is_started()):
                        cmd = 2                 # Установка номера команды
                        timer.pause()           # Приостановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        stop_snd.play()         # Проигрывание сигнала
остановки
                        lock = True             # Установка блокировки
                  # Фальстарт
                  else:
                        cmd = 4                 # Установка номера команды
                        pygame.mixer.stop()     # Остановка всех звуков
                        falstart_snd.play()     # Проигрывание звука
фальстарта
                        break
            # Выход из программы
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                  pygame.mixer.quit()
                  pygame.display.quit()
                  pygame.font.quit()
                  state = False
                  break
      # Окраска фона
      ## Команда красная, ответ во время работы таймера
      if (cmd == 1):
            rscreen.fill(red)
      ## Команда зелёная, ответ во время работы таймера
      elif (cmd == 2):
            rscreen.fill(green)
      ## Стандартный фон
      else:
            rscreen.fill(black)
      # Окраска текста
      # Создаём объект zeit, в котором будет храниться рисунок, точно
соответствующий значению на таймере
      ## Команда красная, фальстарт
      if (cmd == 3):
            zeit = timer_font.render(current_str, True, red)
      ## Команда зелёная, фальстарт
      elif (cmd == 4):
            zeit = timer_font.render(current_str, True, green)
      ## Стандартный цвет текста
      else:
            zeit = timer_font.render(current_str, True, white)
      # Отрисовка
      ## Перерисовываем на правую половину окна объект zeit, центруя его
      rscreen.blit(zeit, [(rscreen.get_width()-zeit.get_width())//2,
(rscreen.get_height()-zeit.get_height())//2])
      ## Перерисовываем правую половину окна
      screen.blit(rscreen, [(screen.get_width())//2, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Необходимое оборудование

Вывод изображения

Чтобы зрители и участники могли наблюдать за ходом игры, было принято решение выводить изображение на два монитора, так как проекторы не всегда отображают изображение в приемлемом качестве.

Кнопки

При реализации игры «Брейн-ринг» возник вопрос о кнопках, необходимых для каждой команды. Было принято решение соединять кнопки с компьютером дистанционно. В качестве беспроводных кнопок использовалась технология и принцип действия беспроводных компьютерных мышей. Так как предусмотрена возможность игры двух команд, то необходимо две кнопки.

Заключение

Изучая язык Python 3 и библиотеку Pygame, я пришёл к выводу, что поставленные задачи вполне разрешимы.

Таким образом, мы можем считать, что наша цель достигнута. На опыте я выяснил, что библиотека Pygame, с помощью которой осуществлялась реализация игры «Брейн-ринг», не только удовлетворяет наши требования, но и пригодна к использованию для реализации более сложных проектов. С помощью неё можно создавать не только игры. Она может использоваться и для визуализации технических процессов и построения математических графиков.

Полный код игры «Брейн-ринг» с комментариями

```
# Импортируем необходимые модули
import sys, pygame
import pygame.mixer, pygame.mouse
from timer import Timer       # код модуля Timer в приложении 3

state = True

# Создаём таймер
timer = Timer(60)
current_str = '0.00'

lock = False            # блокировка кнопки
cmd = -1                # номер команды: 1, 3 - красная; 2, 4 - зелёная; 0
таймер работает, -1 - таймер стоит

score_red, score_green = 0, 0
score_plus = 1
is_red = True

# Инициализируем звуки
pygame.mixer.init()

# Задаём звуковые сигналы
start_snd = pygame.mixer.Sound('timer_start.wav')          # запуск таймера
stop_snd = pygame.mixer.Sound('timer_stop.wav')            # остановка
таймера
end_snd = pygame.mixer.Sound('timer_end.wav')              # истечение
времени
falstart_snd = pygame.mixer.Sound('timer_falstart.wav')    # фальстарт

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()

# Задаём шрифты
score_font = pygame.font.SysFont(None, 400)                # Шрифт очков
timer_font = pygame.font.SysFont('Courier New', 200)       # Шрифт таймера
small_font = pygame.font.SysFont('Courier New', 48)        # Шрифт количества
прибавляемых очков

# Задаём окна
lscreen = pygame.Surface((640, 480))     # левая половина окна (счёт)
rscreen = pygame.Surface((640, 480))     # правая половина окна (таймер)
screen = pygame.display.set_mode((1280, 480), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) #
Разрешаем обработку событий мыши и клавиатуры
pygame.mouse.set_visible(False) # Делаем указатель невидимым
```

```
r1 = pygame.Rect(310, 230, 20, 20) # Добавляем квадрат, показывающий, какой
команде присуждаются баллы

while state:
      current_str = '{:.2f}'.format(timer.current()) # Строка, соответствующая
значению таймера
      # Проигрывание сигнала окончания
      if (timer.current() == timer.end) and (timer._started) and (cmd != -1):
            end_snd.play()
            cmd = -1
      lscreen.fill(black) # Заполняем левую половину экрана чёрным
      # Вывод квадрата r1, окрашенного в цвет команды, которой нужно присудить
баллы
      if (is_red):
            pygame.draw.rect(lscreen, red, r1)
      else:
            pygame.draw.rect(lscreen, green, r1)
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Запуск таймера (Enter)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RETURN):
                  cmd = 0                       # Сброс номера команды
                  pygame.mixer.stop()           # Остановка всех звуков
                  lock = False                  # Снятие блокировки
                  start_snd.play()
                  if (not timer.is_paused()):
                        timer.start()           # Запуск таймера
                  else:
                        timer.resume()          # Продолжение работы таймера
            # Остановка и обнуление таймера (Space)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_SPACE):
                  cmd = -1                # Сброс номера команды
                  pygame.mixer.stop()     # Остановка всех звуков
                  lock = False            # Снятие блокировки
                  timer.stop()            # Остановка таймера
                  timer.null()            # Обнуление таймера
            # Остановка таймера красной командой (правая кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 3)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer.is_started()):
                        cmd = 1                 # Установка номера команды
                        timer.pause()           # Приостановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        stop_snd.play()         # Проигрывание сигнала
остановки
                        lock = True             # Установка блокировки
                  else:
                        cmd = 3                 # Установка номера команды
                        pygame.mixer.stop()     # Остановка всех звуков
                        falstart_snd.play()     # Проигрывание сигнала
фальстарта
                        break
            # Остановка таймера зелёной командой (средняя кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 2)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer.is_started()):
                        cmd = 2                 # Установка номера команды
```

```
                        timer.pause()           # Приостановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        stop_snd.play()         # Проигрывание сигнала
остановки
                        lock = True             # Установка блокировки
                  # Фальстарт
                  else:
                        cmd = 4                 # Установка номера команды
                        pygame.mixer.stop()     # Остановка всех звуков
                        falstart_snd.play()     # Проигрывание сигнала
фальстарта
                        break
            # Переключение команды (Numpad /)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_DIVIDE):
                  is_red = not is_red
            # Добавление очков команде (Numpad +)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and (is_red):
                  score_red += score_plus
                  score_plus = 1
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and not (is_red):
                  score_green += score_plus
                  score_plus = 1
            # Изменение количества прибавляемых очков (стрелки ← →)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_LEFT):
                  score_plus -= 1
                  if (score_plus <= 0):
                        score_plus = 1
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RIGHT):
                  score_plus += 1
            # Обнуление очков (Backspace)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_BACKSPACE):
                  score_red = 0
                  score_green = 0
            # Выход из программы (Escape)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                  pygame.mixer.quit()
                  pygame.display.quit()
                  pygame.font.quit()
                  state = False
                  break
      # Окраска фона
      ## Команда красная, ответ во время работы таймера
      if (cmd == 1):
            rscreen.fill(red)
      ## Команда зелёная, ответ во время работы таймера
      elif (cmd == 2):
            rscreen.fill(green)
      ## Стандартный фон
      else:
            rscreen.fill(black)
      # Окраска текста
      # Создаём объект zeit, в котором будет храниться рисунок, точно
соответствующий значению таймера
      ## Команда красная, фальстарт
```

```
      if (cmd == 3):
            zeit = timer_font.render(current_str, True, red)
      ## Команда зелёная, фальстарт
      elif (cmd == 4):
            zeit = timer_font.render(current_str, True, green)
      ## Стандартный цвет текста
      else:
            zeit = timer_font.render(current_str, True, white)
      # Отрисовка
      ## Создаём объекты red_score и green_score, в которых хранится рисунок с
числом, соответствующим счёту команд
      red_score = score_font.render(("%d" % score_red), True, red)
      green_score = score_font.render(("%d" % score_green), True, green)
      ## Создаём объект plus_score, в котором хранится рисунок с числом,
соответствующим количеству прибавляемых очков
      plus_score = small_font.render(("%d" % score_plus), True, white)
      ## Перерисовываем эти объекты на левую половину окна, центруя их
      lscreen.blit(red_score, [(lscreen.get_width()//2 -
red_score.get_width()) // 2, (lscreen.get_height() -
red_score.get_height()) // 2])
      lscreen.blit(green_score, [((lscreen.get_width()//2 -
green_score.get_width()) // 2)+lscreen.get_width()//2, (lscreen.get_height()
- green_score.get_height()) // 2])
      lscreen.blit(plus_score, [(lscreen.get_width()-
plus_score.get_width())//2, 436])
      ## Перерисовываем на правую половину окна объект zeit, центруя его
      rscreen.blit(zeit, [(rscreen.get_width()-zeit.get_width())//2,
(rscreen.get_height()-zeit.get_height())//2])
      ## Перерисовываем левую половину окна
      screen.blit(lscreen, [0, 0])
      ## Перерисовываем правую половину окна
      screen.blit(rscreen, [(screen.get_width())//2, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Приложение 2

Клавиши управления Брейн-системой

Клавиша	Назначение
Enter	Запуск таймера
Space	Обнуление таймера
Numpad /	Переключение команды, которой следует начислить очки
Numpad +	Начисление очков выбранной команде
Backspace	Обнуление счёта
Esc	Выход
Left ←	Уменьшение количества прибавляемых очков
Right →	Увеличение количества прибавляемых очков
Правая кнопка мыши	Остановка таймера красной командой
Средняя кнопка мыши	Остановка таймера зелёной командой

Код модуля timer, используемого в программе

```
"""
Идея создания класса таймера состоит в том, что вместо изменения времени в
параллельном потоке отмечать время последнего запуска, последней приостанов-
ки и последней остановки. Чтобы ход времени сохранялся, каждый раз необходимо
сохранять значение таймера в момент остановки и прибавлять его после запуска.
Также реализована логическая остановка таймера. Когда значение таймера
больше некоторого заданного числа, обновление значения таймера прерывается.
"""
from time import time

class Timer(object):

    def __init__(self, end=-1):
        self._start = 0
        self._stop = 0
        self._pause = 0
        self._pplus = 0
        self._end = end
        self._started = False
        self._paused = False

    def start(self):
        self._start = time()
        self._started = True
        self._paused = False

    def stop(self):
        self._stop = time()
        self._started = False
        self._paused = False

    def pause(self):
        self._pause = time()-self._pplus
        self._paused = True

    def resume(self):
        self._pplus = time()-self._pause
        self._paused = False

    def null(self):
        self._start = 0
        self._stop = 0
        self._pause = 0
        self._pplus = 0

    def get_end(self):
        return self._end

    def set_end(self, value):
        if (isinstance(value, float)):
            self._end = value
        elif (isinstance(value, int)):
            self._end = float(value)

    end = property(get_end, set_end)
```

```
    def is_started(self):
        if (self._started) and (not self._paused):
            if ((time()-self._start-self._pplus <= self._end)
                or (self._end == -1)):
                return self._started
            else:
                return False
        elif (self._started) and (self._paused):
            if (self._pause-self._start <= self._end) or (self._end == -1):
                return self._started
            else:
                return False
        else:
            return self._started

    def is_paused(self):
        return self._paused

    def current(self, mode='float'):
        if (mode == 'float'):
            if (self._started):
                if (not self._paused):
                    if ((time()-self._start-self._pplus <= self._end)
                        or (self._end == -1)):
                        return time()-self._start-self._pplus
                    else:
                        return self._end
                else:
                    if ((self._pause-self._start <= self._end)
                        or (self._end == -1)):
                        return self._pause-self._start
                    else:
                        return self._end
            elif (not self._started):
                if ((self._stop-self._start <= self._end)
                    or (self._end == -1)):
                    return self._stop-self._start
                else:
                    return self._end
        elif (mode == 'int'):
            if (self._started):
                if (not self._paused):
                    if ((time()-self._start-self._pplus <= self._end)
                        or (self._end == -1)):
                        return int(time()-self._start-self._pplus)
                    else:
                        return int(self._end)
                else:
                    if ((self._pause-self._start <= self._end)
                        or (self._end == -1)):
                        return int(self._pause-self._start)
                    else:
                        return int(self._end)
            elif (not self._started):
                if ((self._stop-self._start <= self._end)
                    or (self._end == -1)):
                    return int(self._stop-self._start)
                else:
                    return int(self._end)

    def back_current(self, mode='int'):
```

```
    if (mode == 'float'):
        if (self._started):
            if (not self._paused):
                if ((time()-self._start-self._pplus <= self._end)
                    or (self._end == -1)):
                    return self._end-(time()-self._start)+self._pplus
                else:
                    return 0.0
            else:
                if ((self._pause-self._start <= self._end)
                    or (self._end == -1)):
                    return self._end-(self._pause-self._start)
                else:
                    return 0.0
        elif (not self._started):
            if ((self._stop-self._start <= self._end)
                or (self._end == -1)):
                return self._end-(self._stop-self._start)
            else:
                return 0.0
    elif (mode == 'int'):
        if (self._started):
            if (not self._paused):
                if ((time()-self._start-self._pplus <= self._end)
                    or (self._end == -1)):
                    return int((
                    self._end-(time()-self._start-1)+self._pplus))
                else:
                    return 0
            else:
                if ((self._pause-self._start <= self._end)
                    or (self._end == -1)):
                    return int(self._end-(self._pause-self._start-1))
                else:
                    return 0
        elif (not self._started):
            if ((self._stop-self._start <= self._end)
                or (self._end == -1)):
                return int(self._end-(self._stop-self._start-1))
            else:
                return 0

def __str__(self):
    if (self.is_started()):
        if (not self.is_paused()):
            state = "started"
        else:
            state = "paused"
    else:
        state = "stopped"
    return "<Timer current={0} state={1}>".format(self.current(), state)
```