مداد رنگی

سارا به تازگی فروشگاه لوازم التحریری به نام مداد رنگی را راه‌اندازی کرده است. او می خواهد سیستمی برای بخش مالی فروشگاه داشته باشد که به او در نگهداری حساب ها کمک کند. برای این منظور با استفاده از برنامه نویسی شی گرایی یک کلاس به اسم MedadRangi ایجاد کرده و تمام مواردی که نیاز دارد را در آن قرار می دهد. از آنجا که روزهای اول افتتاح فروشگاه سارا سرگرم چیدمان است، دنبال برنامه نویسی میگردد که به آن کمک کند. شما هم می توانید از این فرصت استفاده کنید :)

متغیرهایی که در این کلاس باید تعریف شوند، دو دسته اند:

۱. متغیرهایی که برای هر کالا وجود دارند عبارتند از: اسم، قیمت، تعداد، کشور سازنده و نام کارخانه
۲. متغیرهایی که برای فروشگاه هستند و تغییر نمی کنند، که عبارتند از: نرخ تخفیف و طول و عرض جغرافیایی

نحوه کارکرد کلاس باید به این گونه باشد که بتواند هر کالا که ایجاد می شود به کالاهای قبلی اضافه شود.

متدهایی که برای این کلاس باید در نظر گرفته شوند:

- متد final_price که با توجه به **نرخ تخفیف** و تعداد کالا، قیمت نهایی را برای هر کالا به طور جداگانه محاسبه کند.
- متد welcome که با توجه به ساعت، صبح بخیر (از 6 صبح تا 12 ظهر)، بعد از ظهر بخیر (از 12 ظهر تا 18بعد از ظهر) و عصر بخیر چاپ کند.
- متد calculate_distance که با توجه به طول و عرض جغرافیایی فروشگاه، مسافت را از طول و عرض جغرافیایی مقصد محاسبه کند.
- متد load_csv که فایل csv کالا ها را خوانده و آن ها را ذخیره کند.

توجه داشته باشید که طول و عرض جغرافیایی فروشگاه (35.74317403843504, 51.50185488303431) و نرخ تخفیف افتتاحیه 10 درصد است.

ماتریس

در این سوال از شما می‌خواهیم کلاس Matrix را پیاده سازی کنید و برای این کار دو کلاس دیگر به نام های Complex و Integer خواهیم داشت که در ادامه توضیح خواهیم داد.

می توانید توضیحات هر یک از مواردی که باید در این کلاس ها تعریف شوند را از پایین ببینید.

▼ تابع init یا سازنده

کلاس Integer

این تابع ورودی value را می گیرد که باید از تایپ int باشد و اگر نبود در init اکسپشن TypeError دهد، اگر تایپش int بود، آن را به عنوان یک instance variable در خود ذخیره کند.

کلاس Complex

این تابع همانطور که سر کلاس دیدید، یک بخش حقیقی و یک بخش موهومی به عنوان ورودی می‌گیرد.

کلاس Matrix

این کلاس تعداد ردیف و تعداد ستون و همچنین یک لیست شامل اعضا را به عنوان ورودی می‌گیرد. اعضای این لیست از تایپ Integer یا Complex یا مخلوطی از این دو است. برای مثال اگر ورودی ما لیست [1,2,3,4,5,6] باشد و تعداد سطر عدد 2 و تعداد ستون عدد 3 باشد، ماتریس ما به شکل زیر خواهد شد.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

نکته: شما می توانید لیست اعضا را به همان شکل ذخیره کنید و به کمک تعداد ردیف و تعداد ستون به اعضای مختلف برای مثال عضو ردیف دوم و ستون سوم دسترسی پیدا کنید.

▼ تعریف چند تا static method و class method

static methods

تابع make_unit_matrix(n)

این تابع تعداد ردیف و ستون را در قالب یک عدد به عنوان ورودی می‌گیرد و یک ماتریس همانی با همان تعداد ردیف و ستون که اعضایش از کلاس Integer هستند، برمیگرداند.

تابع get_ith_row(matrix, i)

این تابع یک ماتریس و شماره ردیف که از صفر شروع می شود را به عنوان ورودی می‌گیرد و و اعضای ردیف i ام رو در قالب یک لیست برمیگرداند.

تابع get_ith_col(matrix, i)

تعریف این تابع همانند تابع get_ith_row است ولی در پیاده سازی تفاوت دارد که متوجه آن خواهید شد. این تابع ستون j ام را در قالب لیست برمیگرداند.

تابع is_zero_matrix(matrix)

بررسی میکند که آیا ماتریس ورودی یک ماتریس صفر است یا نه و بولین برمیگرداند.

تابع is_unit_matrix(matrix)

همانی بودن ماتریس را بررسی میکند.

تابع is_top_triangular_matrix(matrix)

بالامثلثی بودن ماتریس را بررسی می‌کند.

تابع is_bottom_triangular_matrix(matrix)

پایین مثلثی بودن ماتریس را بررسی می‌کند.

class methods

تابع make_matrix_from_string(elements)

این تابع بدین گونه است که اگر ما رشته‌ی 9 8 7,6 5 4,3 2 1 را بدهیم ماتریس زیر ساخته میشود.

$$\begin{bmatrix} 1 & 2 & 4 \\ 3 & 5 & 7 \\ 6 & 8 & 9 \end{bmatrix}$$

ولی توجه کنید که اعضای ماتریس نباید از تایپ int تعریف شده در پایتون باشند و باید از یکی از تایپ های Integer یا Complex یا مخلوطی از این دو باشند، پس برای هر کدام از عناصر این رشته که با فاصله از هم جدا شده اند، ابتدا باید تشخیص دهید که از تایپ Integer است یا Complex و بعد آن را به class method تعریف شده در آن کلاس بدهید تا آن آبجکت را بسازد. نکته: در کلاس های Integer و Complex باید متد هایی تعریف کنید که رشته می گیرند و آبجکت را میسازند مثلا برای Complex رشته‌ی 2i+5 را میدهیم(عدد مختلط را دقیقا به همین شکل ورودی می‌دهیم) و یک آبجکت از این کلاس با مقدار حقیقی 5 و موهومی 2 تحویل میگیریم. ** نکته: ممکن است برای پیاده‌سازی این توابع بخواهید عملگر های دیگری را برای راحتی کار پیاده سازی کنید برای مثال عملگر تساوی که این به خودتان بستگی دارد.

▼ بخش operator overloading

عملگر +

کلاس Matrix

این عملگر برای کلاس ماتریس اینگونه است که ابتدا بررسی می‌کند که تایپ عملوند دیگر چه چیزی است، اگر ماتریس بود باید جمع ماتریسی(اگر ممکن بود وگرنه اکسپشن ValueError دهد) انجام دهد و اگر تایپ عملوند دیگر Integer یا Complex بود، باید تک تک اعضای ماتریس را با آن عدد جمع کند.

نکته: اگر تایپ عملوند دیگر هیچکدام از این‌ها نبود، اکسپشن TypeError بدهد. برای چک کردن تایپ ها می توانید از تابع isinstance در پایتون استفاده کنید.

کلاس Complex

همانند کلاس ماتریس، اینجا هم باید تایپ عملوند سمت راست را بررسی کنیم، اگر تایپ آن Integer یا Complex بود، باید عدد مختلط جدید را برگرداند. برای حالتی که ماتریس بود باید از یک

ترفند استفاده کنید، برای این حالت تنها به یک خط کد نیاز است. دوباره اگر تایپ عملوند دیگر هیچکدام از اینها نبود، اکسپشن TypeError بدهد.

کلاس Integer

در این کلاس برای حالتی که تایپ عملوند سمت راست ماتریس یا عدد مختلط است، از همان ترفندی که در کلاس Complex گفته شد استفاده کنید و برای حالتی که تایپ عملوند دیگر Integer بود، یک Integer جدید با مقدار مجموع این دو برگردانید.

عملگر *

کلاس Matrix

اگر تایپ عملگر دیگر ماتریس بود باید ضرب ماتریسی(اگر ممکن بود وگرنه اکسپشن ValueError دهد) انجام دهد و اگر تایپ عملوند دیگر Integer یا Complex بود، باید این عدد را در تک تک اعضای ماتریس ضرب کند.

نکته:از توابع استاتیکی که تعریف کردید استفاده کنید، همچنین اگر تایپ عملوند دیگر هیچکدام از این‌ها نبود، اکسپشن TypeError بدهد.

کلاس Complex

اگر تایپ عملوند راست Integer یا Complex بود، باید عدد مختلط جدید را برگرداند. برای حالتی که ماتریس بود تنها به یک خط کد نیاز است. دوباره اگر تایپ عملوند دیگر هیچکدام از اینها نبود، اکسپشن TypeError بدهد.

کلاس Integer

در این کلاس برای حالتی که تایپ عملوند سمت راست ماتریس یا عدد مختلط است، تنها به یک خط کد نیاز است و برای حالتی که تایپ عملوند دیگر Integer بود، یک Integer جدید با مقدار ضرب این دو برگردانید.

عملگر -

این عملگر را به خودتان می‌سپاریم فقط راهنمایی اینکه برای تفریق می‌توانید از جمع با قرینه‌ی عملوند دیگر استفاده کنید و این کار را می‌توانید به طرق مختلف انجام دهید. فقط یادتان باشد که نمی توانید از تایپ int تعریف شده در پایتون برای قرینه سازی استفاده کنید و اکثر عملیات ها بین همین سه نوعی که تعریف کردیم انجام می‌شود.

▼ تابع Multiply

این تابع دو ورودی میگیرد و حاصلضرب بین آن ها را برمیگرداند. این تابع را خارج از همه کلاس ها تعریف می‌کنیم. توجه کنید که طبق duck typing کافی است عملگر ضرب بین آن دو ورودی تعریف شده باشد و اینکه دو ورودی از یک نوع نباشند، اهمیتی ندارد.

- در این سوال ورودی و خروجی قرار نمی‌دهیم اما کد شما بررسی و تست می‌شود که ضرب و جمع و تفریق بین آبجکت های این کلاس ها به درستی انجام شود، همچنین class/static method ها تست می‌شوند.

موفق باشید.

هندسه

نکات مهم

- بیشتر تمرکز این تمرین بر OOP است.
- دقت کنید که کد شما باید به اندازه کافی comment و docstring داشته باشد.
- شما مجاز هستید که هر ماژول دیگری که نیاز دارید، پیاده‌سازی کنید. اصول کدنویسی تمیز را فراموش نکنید!
- در این تمرین می بایست ماژول `main.py` و 6 کلاس در ماژول `geometry.py` پیاده سازی کنید.

فایل اصلی (`main.py`)

این فایل، فایل شروع برنامه است. برنامه موجود در این فایل به کاربر اجاز می‌دهد که تعدادی شکل به یک لیست اضافه کند و با استفاده از آنها کارهایی انجام دهد. بعد از اجرای main.py باید یک منو به شکل زیر به کاربر نشان داده شود:

```
Learn Geometry.
  What do you want to do?
  (1) Add new shape
  (2) Show all shapes
  (3) Show shape with the largest perimeter
  (4) Show shape with the largest area
  (5) Show formulas
  (0) Exit program
```

توضیح کارکرد ها

با توجه به منو معلوم می‌شود که این برنامه دارای 5 کارکرد است.

1. اضافه کردن یک شکل جدید

با این دستور کاربر یک شکل جدید به لیست شکلها اضافه می‌کند. کاربر باید بتواند نوع شکلی که می‌خواهد اضافه بکند را انتخاب کند و مشخصات مورد نیاز برای ایجاد آن شکل را هم تعیین کند.

2. نشان دادن همه‌ی شکلها

با اجرای این دوست باید جدولی شامل همه شکل‌هایی که در لیست وجود دارد، نمایش داده شود. `ShapeList.get_shapes_table()` را ببینید.

3. نشان دادن شکلی که محیط آن از بقیه بیشتر است.

این دستور شکلی که محیط آن از بقیه بیشتر است را از لیست انتخاب می‌کند و نمایش می‌دهد.

4. نشان دادن شکل با بیشترین مساحت

این دستور شکلی که مساحت آن از بقیه شکل‌ها بیشتر است انتخاب می‌کند و نشان می‌دهد.

5. نشان دادن فرمول

پس از اجرای این دستور یک باید کاربر یک نوع شکل را انتخاب کنید و در خروجی فرمول محاسبه محیط و مساحت آن شکل به او نمایش داده شود.

کلاس ها

این بخش، مهم‌ترین قسمت این تمرین است. باید همه‌ی کلاس‌ها را در ماژول `geometry.py` پیاده‌سازی کنید.

کلاس Shape

این یک کلاس انتزاعی (abstract) است. به این معنا که از این کلاس هیچ نمونه‌ای (instance) ساخته نخواهد شد. فقط از این کلاس به عنوان کلاس والد (parent) برای سایر کلاسهای عینی (concrete) استفاده خواهیم کرد. این کلاس در واقع یک boilerplate برای کلاس‌هایی است که از آن ارث می‌برند. یعنی شامل attribute ها و method هایی است که باید در کلاس‌های فرزند پیاده‌سازی شوند.

متودهای نمونه (Instance methods)

```
get_area(self)
```

مساحت شکل را برمی‌گرداند.

`get_perimeter(self)`

محیط شکل را برمی‌گرداند.

`__str__(self)`

اطلاعات مربوط به شکل را به صورت یک رشته برمی‌گرداند.

متودهای کلاس (class methods)

`check_if_args_not_below_zero(cls,*args)`

اگر مقدار هیچکدام از پارامترها (args) منفی نباشد، True برمی‌گرداند. در صورتی که مقدار یکی از پارامترها منفی باشد، موجب ValueError می‌شود. (مثلا برای اینکه دایره با شعاع منفی وجود ندارد.)

`get_area_formula(cls)`

فرمول محاسبه مساحت شکل را به صورت یک رشته برمی‌گرداند.

`get_perimeter_formula(cls)`

فرمول محاسبه‌ی محیط شکل را برمی‌گرداند.

توضیح دهید که چرا این متودها باید از نوع class methods باشند؟

کلاس دایره (Circle)

این کلاس، معرف شکل دایره است. کلاس والد آن کلاس Shape است.

- Attributes
  - `r`
    - نوع داده: float
    - توصیف: اندازه شعاع دایره
- Methods
  - `__init__(self, r)`
    - سازنده (constructor) کلاس دایره

سایر متودهایی که از کلاس والد به ارث برده شده‌اند را پیاده‌سازی (override) کنید.

کلاس مثلث (Triangle)

کلاس والد آن کلاس Shape است.

- Attributes
  - a
    - نوع داده: float
    - توصیف: اندازه ضلع اول مثلث
  - b
    - نوع داده: float
    - توصیف: اندازه ضلع دوم مثلث
  - c
    - نوع داده: float
    - توصیف: اندازه ضلع سوم مثلث
- Methods
  - `__init__(self, a, b, c)`
    - سازنده (constructor) کلاس مثلث

متودهای کلاس والد را override کنید.

راهنمایی: برای محاسبه‌ی مساحت مثلث از فرمول هرون استفاده کنید.

کلاس مثلث متساوی‌الاضلاع (Equilateral triangle)

کلاس والد آن کلاس Triangle است.

- Attributes
  - a
    - نوع داده: float
    - توصیف: اندازه اضلاع مثلث متساوی‌الاضلاع
- Methods

- `__init__(self, a)`
  - سازنده (constructor) کلاس مثلث متساوی‌الاضلاع

راهنمایی: سعی کنید از attribute های کلاس والد آن استفاده کنید. خودتان تصمیم بگیرید که آیا لازم است متودهای کلاس والد را override کنید یا خیر.

کلاس مستطیل (Rectangle)

کلاس والد آن کلاس Shape است.

- Attributes
  - `a`
    - نوع داده: float
    - توصیف: اندازه طول مستطیل
  - `b`
    - نوع داده: float
    - توصیف: اندازه عرض مستطیل
- Methods
  - `__init__(self, a, b)`
    - سازنده (constructor) کلاس مستطیل

راهنمایی: سعی کنید از attribute های کلاس والد آن استفاده کنید. سایر متودهای به ارث رسیده از کلاس والد را override کنید.

کلاس مربع (Square)

کلاس والد آن کلاس Rectangle است.

- Attributes
  - `a`
    - نوع داده: float
    - توصیف: اندازه ضلع مربع
- Methods

    - `__init__(self, a)`
  - سازنده (constructor) کلاس مربع

راهنمایی: سعی کنید از attribute های کلاس والد آن استفاده کنید. خودتان تصمیم بگیرید که آیا نیاز هست متودهای به ارث رسیده را override کنید یا خیر.

کلاس پنج‌ضلعی منتظم (Regular pantagon)

کلاس والد آن کلاس Shape است.

- Attributes
  - `a`
    - نوع داده: float
    - توصیف: اندازه ضلع پنج‌ضلعی
- Methods
  - `__init__(self, a)`
    - سازنده (constructor) کلاس پنج‌ضلعی منتظم

راهنمایی: سعی کنید از attribute های کلاس والد آن استفاده کنید. سایر متودهای کلاس والد را override کنید.

کلاس لیست اشیاء (ShapeList)

این کلاس به منظور نگهداری از شکل‌های هندسی (اشیائی که از کلاس Shape ارث می‌برند) ایجاد می‌شود.

- Attributes
  - `shapes`
    - نوع داده: لیست
    - توصیف: لیست object های نوع Shape
- Methods
  - `__init__(self, shape)`
    - سازنده (constructor) شئ ShapeList

- `add_shape(self, shape)`
  - این تابع shape را به لیست شکل‌ها اضافه می‌کند. این متود باید چک کند که این shape از کلاس Shape ارث می‌برد یا خیر (به صورت مستقیم یا غیر مستقیم). در غیر این صورت باید TypeError بدهد. **(راهنمایی: تابع isinstance را چک کنید.)**
- `get_shapes_table(self)`
  - مقدار برگشتی: string
  - این متود لیست شکل‌ها را به صورت یک جدول در قالب رشته برمی‌گرداند. رشته برگشتی باید به شکل جداول فرمت مارک دان باشد (نمونه در انتهای توضیحات).
  - مقدار برگشتی : string
- `get_largest_shape_by_perimeter(self)`
  - شکلی را که محیط آن از همه بزرگتر است، برمی‌گرداند. **راهنمایی: نگاهی به متودهای مقایسه بندازید.**
  - مقدار برگشتی: Shape
- `get_largest_shape_by_area(self)`
  - شکلی را که مساحت آن از همه بزرگتر است، برمی‌گرداند. **راهنمایی: نگاهی به متودهای مقایسه بندازید.**
  - مقدار برگشتی: Shape

موارد امتیازی

همان‌طور که گفتیم تمرکز این تمرین بر جنبه‌ی برنامه‌نویسی شئ‌گرا است. اما اگر دوست داشته باشید، می‌توانید پس از انجام دادن بخش اصلی تکلیف، این بخش را هم انجام دهید.

۱. به کارکردهای برنامه یک کارکرد جدید به نام کوئیز اضافه کنید. برنامه باید یک شکل تصادفی ایجاد کند، به کاربر نوع شکل و ویژگی‌های آن را بگوید و از کاربر بخواهید، محیط و مساحت آن را حساب کند. برنامه باید پاسخ کاربر را چک کند.
۲. یک کارکرد جدید به برنامه اضافه کنید که بتواند شکل‌ها را با استفاده از ماژول Turtle رسم کند.
۳. می‌توانید شکل‌های دیگری را به geometry اضافه کنید.
۴. کل سوال را به صورت گرافیکی پیاده سازی کنید.

نمونه جدول مارک دان

برای راهنمایی بیشتر اینجا کلیک کنید.

idx	Class	__str__	Perimeter	Formula
0	Circle	Circle, r = 3	18.85	π×r^2
1	Square	Square, a = 2	8.00	a^2

آجرشکن!

و اما بازی گرافیکی این سری از تمرینها:

این بازی یک مقداری شبیه به بازی قبلی است؛ یک راکت در پایین صفحه دارید که به سمت چپ یا راست حرکت میکند. یک توپ دارید که به راکت برخورد میکند و برمیگردد. این توپ در صورتی که در پایین صفحه به جایی خارج از راکت برخورد کند، بازی تمام می شود.

در بالای صفحه بازی سه لایه آجر وجود دارد. هر کدام از آجرها در این سه لایه با تعداد ضربات متفاوتی شکسته می شوند: بعضی از آجرها تنها با یک ضربه شکسته می شوند (ظرفیت ضربه این آجرها برابر با 1 است)، بعضی از آجرها با دو ضربه شکسته می شوند و بعضی از آجرها با سه ضربه شکسته می شوند (ظرفیت ضربه این آجرها هم به ترتیب 2 و 3 است).

بازی به این صورت انجام می شود که توپ به راکت برخورد میکند و بعد از آن به سمت آجرها حرکت میکند، در صورت برخورد به هر آجر یک واحد از ظرفیت ضربه آن آجر کم می شود. در صورتی که ظرفیت ضربه یک آجر به صفر برسد، آن آجر از صفحه بازی محو می شود. زمانی که در صفحه بازی هیچ آجری باقی نماند و توپ هم از قسمت پایین صفحه خارج نشده باشد، بازی با پیروزی بازی کننده به پایان می رسد.

(فکر میکنم نسخه های مختلف این بازی را زیاد انجام داده باشید)

برای حل این سوال باید 5 تا کلاس پیاده سازی کنید:

۱. GameObject
۲. Ball
۳. Bat
۴. Brick
۵. Game

هر کدام از این کلاسها دارای متودهایی هستند که رفتار این کلاسها را پیاده سازی می کنند.

کد بازی

```python
import tkinter as tk

class GameObject(object):
    def __init__(self, canvas, item):
        self.canvas = canvas
        self.item = item

    def get_position(self):
        return self.canvas.coords(self.item)

    def move(self, x, y):
        self.canvas.move(self.item, x, y)

    def delete(self):
        self.canvas.delete(self.item)


class Ball(GameObject):
    def __init__(self, canvas, x, y):
        pass

    def update(self):
        pass

    def collide(self, game_objects):
        pass


class Bat(GameObject):
    def __init__(self, canvas, x, y):
        pass

    def set_ball(self, ball):
        pass

    def move(self, offset):
        pass


class Brick(GameObject):

    def __init__(self, canvas, x, y, hits):
```

```
43          pass
44
45      def hit(self):
46          pass
47
48
49  class Game:
50      pass
51
52
53
54  if __name__ == '__main__':
55      root = tk.Tk()
56      root.title('AP4002 Brick Breaker!')
57      # rest of the code
```

کلاس GameObject

این کلاس، والد همه کلاسهای دیگر (به جز کلاس Game) است. یعنی سایر کلاسها از این کلاس ارث می برند. این کلاس دارای متودهایی که رفتارهای عمومی موجودیت های مختلف در بازی را کنترل می کنند. پیاده سازی تمام متودهای این کلاس در قطعه کد فوق برای شما قرار داده شده است.

کلاس Ball

کلاس Ball از کلاس GameObject ارث می برد. (برای اینکه هر Ball یک GameObject است)

ویژگی ها (attributes)

۱. شعاع توپ (radius)

۲. جهت حرکت توپ بعد از برخورد به یک شئ (direction) توجه: میتوانید جهت یک توپ را صرفا به صورت [self.direction = [-1, 1 تعریف کنید.

۳. سرعت حرکت توپ (speed)

۴. آیتم گرافیکی نماینده شئ در صفحه گرافیکی بازی (item)

۵. صفحه گرافیکی بازی که این توپ در آن نمایش داده می شود (canvas)

متودها:

۱. متود `__init__`

دارای سه پارامتر canvas و x و y است. اولی ویجت ترسیم اشکال اشیاء بازی است. دومی و سومی هم به موقعیت اولیه قرار گرفتن توپ در صفحه اشاره می کنند.

در این متود attribute های این کلاس مقداردهی می شوند.

بعضی از attribute ها از طریق متود **init** کلاس والد مقداردهی می شوند (مثل item و canvas) و مقدار اولیه آنها از طریق پارامترهای متود **init** محاسبه می شود (باید item با استفاده از متود create_oval نمونه canvas با توجه به شعاع توپ و موقعیت قرار گرفتن آن در صفحه ساخته شود). بعضی از attribute ها در کلاس والد وجود نداشتند، ممکن است با مقدار ثابت مقداردهی شوند یا آنها هم از طریق پارامترهای **init** قابل تنظیم باشند (مثل radius، direction, speed).

۲. متود update

با استفاده از متود get_position که از کلاس والد به ارث برده است، مختصات کنونی توپ به دست می آید. با توجه به direction و speed (و احتمالا طول و عرض صفحه بازی) موقعیت بعدی توپ محاسبه می شود و با استفاده از متود move که از کلاس والد به ارث برده است، توپ به مختصات جدید منتقل می شود.

۳. متود collide

پارامتر آن لیستی از اشیاء درون بازی است که توپ با آنها برخورد کرده است (این متود بعدا در کلاس Game فراخوانی می شود و لیست اشیائی که توپ با آنها برخورد کرده برای این تابع ارسال می شود)

در این تابع با توجه به تعداد و نوع اشیائی که توپ با آنها برخورد کرده است، جهت حرکت بعدی توپ (direction) تنظیم می شود. همچنین باید اگر نوع شیئی که توپ با آن برخورد داشته از نوع آجر است، اقدامات لازم صورت گیرد (تابع hit برای آجرهایی که توپ با آنها برخورد داشته فراخوانی شود. متود hit در کلاس Brick پیاده سازی خواهد شد.)

کلاس Bat

ویژگی ها

۱. اندازه افقی راکت (width).

۲. ارتفاع راکت (height)

۳. توپ مرتبط با راکت (ball)

۴. آیتم گرافیکی نماینده شئ در صفحه بازی (item)

۵. ویجت گرافیکی بازی که این راکت در آن نمایش داده می شود (canvas)

متودها

۱. متود `__init__` .

مقادیر اولیه attribute ها را تنظیم می کند.

۲. متود set_ball مقدار ویژگی ball را تنظیم میکند.

۳. متود move:

متود move را override میکند تا حرکت راکت را کنترل کند.

کلاس Brick

ویژگی ها

۱. عرض آجر (width)

۲. ارتفاع آجر (height)

۳. ظرفیت ضربه آجر (hits)

۴. رنگ آجر (color)

۵. شئ گرافیکی آجر در صفحه بازی (item)

۶. ویجت رسم آجرها در صفحه گرافیکی بازی (canvas)

متودها

۱. متود `__init__`

۲. متود hit: از تعداد ظرفیت ضربه آجر یکی کم می کند. در صورتی که تعداد ظرفیت ضربه آجر صفر شده باشد آن را از صفحه حذف می کند (با متود delete که در کلاس والد پیاده سازی شده است). هر کار دیگری که لازم است روی آجر پس از ضربه خوردن انجام شود (اگر کار دیگری هست) در این تابع پیاده سازی می شود.

کلاس Game

این کلاس، کلاس اصلی بازی است که کل بازی را کنترل می کند. طراحی آن، تعریف ویژگیها و متودهای مناسب در آن به عهده شماست.

در این کلاس باید موارد زیر مورد توجه قرار گیرد:

۱. شکل کلی صفحه بازی، رنگ پس زمینه و سایر خصوصیات ظاهری پنجره بازی در این کلاس تنظیم می شود.
۲. اضافه کردن آجرها، اضافه کردن توپ و راکت و نوشته های مورد نیاز در صفحه بازی در این کلاس صورت می گیرد. (برای هر کدام از این وظایف می توانید یک متود تعریف کنید)
۳. شروع بازی و اجزای حلقه بازی نیز در این کلاس صورت می گیرد.

ببینیم چه می آفرینید :)