دستگرمی

سایت ادابیت یکی از سایت‌هایی هست که می توانید در آن مساله‌هایی برای تمرین برنامه‌نویسی پیدا کنید. یکی از محاسن این سایت این هست که مساله‌ها سطح‌بندی و موضوع‌بندی شده اند.

برای اینکه برای حل سوالهای تمرین سری سوم، آمادگی بیشتری پیدا کنید، حل کردن سوالهای دسته کلاس‌ها در این سایت میتواند مفید باشد.

در این بخش از شما می‌خواهیم به دلخواه، 8 تا از سوالهای دسته کلاسها را حل کنید. اگر دوست داشتید میتوانید تعداد سوالهای بیشتری حل کنید.

پس از حل هر سوال می توانید از قسمت Solutions سایر راه حلها را هم ببینید و برای یادگیری بیشتر بررسی کنید.

پیشنهاد ما حل سوالهای زیر است:

۱. Make a circle with OOP
۲. Older than me
۳. Simple OOP Calculator
۴. FullName and Email
۵. Count Number of Instances
۶. BookShelf
۷. Employee Parsing
۸. Employee Class with Keywords

پاسخهای خود را به نحوی که نام فایل مطابق با نام سوال باشد، در مخزن گیتهاب خود قرار دهید. همچنین تصویر پروفایل خود در ادابیت را که نشان می دهد چه سوالهایی را حل کرده اید در همان مخزن قرار دهید.

کتابخانه

در این تمرین باید یک برنامه برای مدیریت یک کتابخانه بنویسید. در فضای این مساله سه کلاس قابل تعریف است:

- کتاب (Book) :
  - هر کتاب دارای یک نام و یک شناسه منحصر به فرد است.
- عضو کتابخانه (LibraryMember):
  - هر عضو دارای یک نام، یک شناسه منحصر به فرد و لیست کتابهایی است که به امانت گرفته است.
    - توجه: اعضای لیست کتابها از کلاس کتاب هستند.
- کتابخانه (Library):
  - هر کتابخانه دارای لیست اعضا و لیستی از کتابها به همراه تعداد موجودی آنها است.
    - در این سوال شما باید کتاب ها و اعضا را به صورت شی از کلاس خودشان ذخیره کنید.

مسئول کتابخانه با اجرای دستور های زیر کتابخانه را مدیریت میکند (در هر خط یک دستور وارد میشود. شما باید متودهایی برای انجام دادن دستورات زیر به کلاس مربوطه اضافه کنید.

۱. `addBook(self, id, name,count)` این تابع تعداد count جلد از کتاب با شناسه id را به مخزن کتابخانه اضافه می‌کند.
۲. `addMember(self, id, name)` این تابع عضو با شناسه id و نام name را به کتابخانه اضافه می‌کند. بدیهی است فهرست کتابهای امانت گرفته شده برای عضو جدید در ابتدای عضویت خالی است.
۳. `get(self, member_id, book_id)` با این تابع به کاربری با شناسه `id_member` کتابی با شناسه `id_book` امانت داده میشود.

- در صورتی که تعداد کتاب هایی که امانت گرفته از سقف مجاز (5 جلد) بیشتر باشد پیام زیر چاپ می‌شود:

MaxReached : [member_name] [member_id]

- در صورتی که حداقل یک جلد از کتاب مربوطه موجود نباشد پیغام زیر چاپ می‌شود:

NotAvailable : [book_name] [book_id]

- بدیهی است که تعداد جلدهای موجود از کتاب فوق در کتابخانه و فهرست کتابهای امانت گرفته شده عضو مربوطه باید آپدیت شوند.

۴. `return(self, member_id, book_id)` با این دستور باید کتاب با شناسه `book_id` که توسط کاربر با شناسه `member_id` به امانت گرفته شده است، به فهرست کتابهای قابل امانت کتابخانه اضافه شود و از فهرست کتابهای امانت گرفته شده عضو مربوطه حذف شود.

۵. `memberStat(self)` با دستور بالا، خلاصه ای از وضعیت اعضا (شامل لیست کتاب هایی که امانت گرفته اند) با فرمت زیر گزارش میشود:

```
------------------------------------------------------------
[Member1 Name] [Member1 ID]
    - [Book1 Name] [Book1 ID]
    - [Book2 Name] [Book2 ID]
    - ...

[Member1 Name] [Member1 ID]
    - [Book1 Name] [Book1 ID]
    - ...
------------------------------------------------------------
```

حساب بانکی

می خواهیم یک کلاس طراحی و پیاده سازی کنیم که عملکردهای مربوط به حسابهای بانکی را نشان دهد.

بخش اول: پیاده سازی کلاس

این کلاس باید دارای خصوصیات و عملکردهای زیر باشد:

شماره حساب (account_number)

هر حساب باید دارای یک **شماره حساب** یکتا باشد. (فرض کنید که شماره حساب منحصرا از طریق initializer قابل مقداردهی است)

نام و نام خانوادگی (first_name | last_name)

صاحب حساب دارای نام و نام خانوادگی است.

اختلاف زمانی (timezone)

برای هر حسابی باید میزان اختلاف زمانی نسبت به ساعت هماهنگ جهانی (Time Zone Offset) مشخص باشد. مثلا برای منطقه زمان ایران این مقدار 3:30+ است.

موجودی حساب (balance)

موجودی حساب باید حتما مقدار نامنفی باشد و نباید بتوان آن را به طور مستقیم تغییر دارد.

واریز و برداشت (deposit | withdrawal)

از طریق عملیات واریز یا برداشت (در صورت کافی بودن موجودی حساب) مقدار موجودی تغییر می کند.

- در صورتی که درخواست برداشت از حساب منجر به منفی شدن مقدار موجودی شود، رد می شود.

نرخ سود ماهانه (interest_rate)

مقدار نرخ سود ماهانه باید به صورت هماهنگ برای همه حسابها اعمال شود. لازم است کلاس دارای یک متود باشد که میزان سودی که به هر حساب تعلق می گیرد را با استفاده از نرخ سود ماهانه محاسبه کند و به همان مقدار به موجودی حساب اضافه کند.

کد تایید (confirmation_number)

هر درخواست تراکنش باید یک کد تایید شامل اطلاعات زیر را ایجاد کند:

- نوع تراکنش (transaction type)

```
'D' : deposit
'W' : withdrawal
'I' : interest deposit
'X' : declined
```

واضح است حالت چهارم زمانی اتفاق می افتد که تراکنش مجاز نبوده و میزان موجودی حساب تغییر نمی کند.

- شماره حساب
- زمان انجام تراکنش (طبق ساعت هماهنگ جهانی)
- شناسه تراکنش (یک عدد صحیح است که به ترتیب به تراکنش‌ها اختصاص پیدا می کند. برای سادگی فرض کنید این عدد از 100 برای تراکنش اول شروع می شود، برای تراکنش دوم مقدار آن 101 می شود و ... به همین صورت به ازای هر تراکنش موفق یا ناموفق از هر کدام از انواع تراکنش (واریز، برداشت، پرداخت سود، تراکنش غیرمجاز) یک واحد به آن اضافه می شود).

تجزیه گر کد تایید (confirmation_code_parser)

این کلاس باید دارای یک متود باشد که با داشتن کد تایید و منطقه زمانی ترجیحی به عنوان پارامتر اطلاعات زیر را برگرداند:

- شماره حساب، کد تراکنش (D, W, ...)، زمان و تاریخ (فرمت UTC)، زمان و تاریخ در منطقه زمانی پارامتر متود، شناسه تراکنش
- مقدار برگشتی این متود باید به شکلی باشد که بتوان با استفاده از dot notation به هریک از مقادیر خواسته شده دسترسی پیدا کرد. پیشنهاد: از Named Tuple استفاده کنید [مرجع].

- توجه کنید که این متود به هیچ اطلاعاتی از instance نیاز ندارد.

مثال

فرض کنید یک حساب با اطلاعات زیر وجود دارد:

- شماره حساب: 140568
- اختلاف زمانی صاحب حساب: 7-
- موجودی حساب: 100.00

فرض کنید که شناسه تراکنش قبلی در سیستم 123 بوده و تراکنش موفق واریز به حساب به میزان 50.00 واحد در زمان زیر طبق ساعت هماهنگ جهانی (UTC) :`

2019-03-15T14:59:00

یا زمان زیر طبق ساعت محلی صاحب حساب

2019-03-15T07:59:00

صورت گرفته است.

مقدار موجودی حساب باید 150.00 واحد شود و کد تایید هم به شکل زیر باشد:

```
D-140568-20190315145900-124
```

متود تجزیه‌گر کد تایید هم باید یک شئ با attribute های زیر را برگرداند:

- `result.account_number --> 140568`
- `result.transaction_code --> D`
- `result.transaction_id --> 124`
- `result.time --> 2019-03-15 07:59:00 (MST)`
- `result.time_utc --> 2019-03-15T14:59:00`

به علاوه، اگر مقدار فعلی نرخ سود مثلا 0.5 درصد است، و میزان موجودی حساب 1000.00 واحد است، با فراخوانی متود `pay_interest` یک تراکنش جدید با کد `I` ایجاد شود و مقدار جدید موجودی حساب به 1050.00 برسد و برای آن یک کد تایید ایجاد شود.

توجه

- شما به خوبی از ویژگی های تایپ `float` آگاه هستید. برای دقت بهتر مقادیر اعشاری از نوع `Decimal` استفاده کنید.
- راه های زیادی برای پیاده سازی این کلاس وجود دارد و در صورت مساله به بیان همه جزئیات لازم برای پیاده سازی کلاس نپرداخته ایم. درباره سایر متودها یا ویژگی های مورد نیاز برای پیاده سازی کلاس خودتان تصمیم بگیرید.
- سعی کنید از همه مفاهیمی که تا اینجای درس برنامه نویسی شئ گرا خوانده اید، استفاده کنید (البته تا جای ممکن)

راهنمایی

همان طور که گفته شد، بهتر است خودتان فکر کنید و ببینید بهترین شکل پیاده سازی این کلاس چگونه است. اما می توانید از این راهنمایی هم کمک بگیرید.

می توانید مانند مثالی که قبلا در کلاس حل شده است، یک کلاس TimeZone برای نگهداری نام منطقه زمانی و اختلاف زمانی (بر حسب ساعت و دقیقه) و یک کلاس اصلی به اسم `Account` تعریف کنید که دارای یک public interface به شکل زیر باشد:

متود `__init__`

که شماره حساب، نام و نام خانوادگی، منطقه زمانی (به عنوان یک پارامتر دلخواه) و میزان موجودی اولیه (با مقدار پیش فرض 0) را مقداردهی کند.

پراپرتی نام/نام خانوادگی

به صورت خواندنی/نوشتنی

پراپرتی نام کامل (`fullname`)

به صورت فقط خواندنی

پراپرتی موجودی حساب (`balance`)

به صورت فقط خواندنی

پراپرتی نرخ سود

یک پراپرتی سطح کلاس

متودهای واریز و برداشت و پرداخت سود

متود تجزیه کننده کد تایید

البته هنوز هم کلاس شما دارای متودها و ویژگی های دیگری هم هست که باید موقع پیاده سازی به آنها فکر کنید.

بخش دوم: تست

با استفاده از بسته `unittest` سعی کنید حالت های خاصی که ممکن است در عمل برای استفاده از این کلاس پیش بیاید تست کنید، باگ های کلاس خود را پیدا کنید و آنها را رفع کنید. برای این سوال استفاده ساده از `unittest` مورد انتظار است.

مهم است

سعی کنید همه کلاسها و متودهایی را که پیاده سازی می کنید با docstring ها و annotation های مناسب تکمیل کنید. پیاده سازی راه حل این مساله به نحوی که کامپیوتر آن را خوب بفهمد، کار زیاد سختی نیست؛ اما پیاده سازی و اضافه کردن توضیحات لازم به نحوی که ما هم خوب آن را بفهمیم، مهم است!

پونگ

در رابط گرافیکی tkinter پایتون شما توانایی ساخت طرح های گرافیکی دلخواه را هم دارید. و می توانید با استفاده از توابع مختلف شکلهای متفاوتی را ایجاد کنید.

همانطور که قبلا گفتیم (در تمرین XO) هر چیزی در tkinter یک ویجت است. در تمرین قبلی با ویجت button آشنا شدیم. یک ویجت دیگر که در این تمرین می خواهیم با آن آشنا شویم، ویجت canvas است.

با استفاده از ویجت canvas میتوانیم روی پنجره برنامه یک صفحه ایجاد کنیم و در آن شکلهایی رسیم کنیم. مثلا فرض کنید میخواهیم روی پنجره اصلی root یک دایره کوچک رسم کنیم:

در مرحله اول یک ویجت canvas خوش رنگ روی root ایجاد می کنیم.

```
# importing tkinter modules
from tkinter import *

# building a window
root = Tk()

#adding window title
root.title("AP4002 Pong Game")

#initializing canvas area and making it ready
canvas = Canvas(root, width=500, height=320, bg='turquoise')
canvas.grid(row=0,column=0)

#creating the loop for the program
root.mainloop()
```

الان در واقع با فراخوانی کلاس Canvas یک instance از کلاس Canvas تعریف کرده ایم، attribute های عرض و طول و رنگ پس زمینه را برای آن مقداردهی کرده ایم و سپس جایگاه آن را در صفحه اصلی تعیین کرده ایم. در نهایت هم که با استفاده از mainloop یک حلقه بی نهایت ایجاد کرده ایم که مادامی که پنجره اصلی را نبسته ایم، پنجره اصلی روی صفحه نشان داده شود.

حالا وقت آن رسیده که دایره کوچکمان را رسم کنیم. کلاس Canvas دارای instance method های متعددی برای رسم انواع شکلها از جمله دایره است. می توانید با چک کردن namespace آن انواع این متودها را ببینید. برای رسم دایره از متود `create_oval` استفاده می کنیم:

```
#creating circle with oval
ball = canvas.create_oval(30,30, 50,50, fill="yellow")

"""
Note:
x1 and y1 coordinates for top left corner
and x2 and y2 coordinates for the bottom right corner
"""
```

به این ترتیب یک دایره زرد روی canvas ایجاد می شود. 4 آرگومان اول نشان دهنده مختصات محل قرار گرفتن دایره روی canvas هستند. توضیحات مربوطه به صورت Note در قطعه کد فوق نوشته شده است.

با ترکیب دو قطعه کد فوق به صورت زیر یک صفحه سبزآبی داریم که یک دایره زرد در گوشه آن درج شده است. (سعی کنید با تغییر دادن اعداد مختصات، با منطق آن آشنا شوید)

فقط در همین حد بگویم که مختصات بالاترین و سمت چپ ترین نقطه روی convas می شود (0,0).

```
# importing tkinter modules
from tkinter import *

# building a window
root = Tk()

#adding window title
root.title("AP4002 Pong Game")

#initializing canvas area and making it ready
canvas = Canvas(root, width=500, height=320, bg='turquoise')
canvas.grid(row=0,column=0)

#creating circle with oval
ball = canvas.create_oval(30,30, 50,50, fill="yellow")

```

```
#creating the loop for the program
root.mainloop()
```

به همین ترتیب می توانیم شکلهای دیگری هم به صفحه اضافه کنیم. مثلا یک مستطیل افقی در پایین صفحه رسم کنیم که نشان دهده راکت بازی باشد.

خودتان دستور زیر را با کد بالا ترکیب کنید و نتیجه را ببینید:

```
#creating rectangle
bat = canvas.create_rectangle(0, 310, 100,320, fill="blue")
```

اعداد مختصات گوشه بالا سمت چپ، و مختصات گوشه پایین سمت راست مستطیل هستند. (لطفا کمی تامل کنید تا منطق مختصات رسم شکلها را به خوبی درک کنید)

راستی همه اینها برای این هست که قصد داریم در این تمرین بازی معروف پونگ را به کمک tkinter و به روش OOP پیاده سازی کنیم.

اگر همه چیز خوب پیش رفته باشد، الان باید در یک صفحه سبزآبی، دو شکل ببینید، یک مستطیل آبی به عنوان راکت در پایین صفحه، سمت چپ و یک دایره زرد، تقریبا بالای صفحه سمت چپ.

چه طور کاری کنیم که این شکلها روی صفحه حرکت کنند؟

ویجتهایی که روی صفحه داریم ممکن هست لازم باشد دو جور حرکت کنند:

۱. کاربر یک کلید را فشار دهد یا عملی با ماوس انجام دهد که منجر به حرکت ویجت بشود.
۲. شکل خودش به صورت دلخواه حرکت کند.

برای راکت به حرکت نوع 2 نیاز داریم و برای توپ به حرکت نوع 1. اول حرکت نوع 2 را ببینیم.

حرکت دادن راکت (bat) روی صفحه با استفاده از کلید های --> و <--

(منظورم از این دو تا کلید روی کیبرد واضح هست دیگه؟)

```
# move bat
def move_bat(event):
    if event.keysym == 'Right':
        canvas.move(bat, 10,0)
```

```
    elif event.keysym == 'Left':
        canvas.move(bat, -10,0)

canvas.bind_all('<Right>', move_bat)
canvas.bind_all('<Left>', move_bat)
```

در قطعه کد بالا چند متود استفاده شده که یکی یکی توضیح می دهیم:

۱. متود `bind_all` : این متود با دو آرگومان فراخوانی شده است. آرگومان اول نشان دهنده کلیدی است که قرار است فشار داده شود. به صورت پیش فرض این متود می داند رشته `<Right>` معادل کلید -> یا right arrow است. بنابراین اگر اگر وقتی این canvas فعال است، کلید right arrow روی کیبرد فشار داده شود، تابع `move_bat` را اجرا می کند. فشرده شدن یک کلید یک event است. با فشرده شدن یک کلید، متود `bind_all` کاری میکند که `move_bat(event)` اجرا شود. در این مثال، event با رشته `<Right>` برابر است.

۲. متود `move` : این متود با سه آرگومان فراخوانی می شود. آرگومان اول، شیئی است که روی canvas قبلا ایجاد شده و حالا قصد حرکت دادن آن را داریم. آرگومان دوم میزان جابه‌جایی این شئ در محور افقی (به سمت راست با علامت مثبت و به سمت چپ با علامت منفی) را نشان می دهد. آرگومان سوم هم احتمالا حدس می زنید؛ میزان جابه‌جایی این شئ در محور عمودی (به سمت پایین با علامت مثبت و به سمت بالا با علامت منفی) نشان می دهد.

۳. تابع `move_bat` : این تابع را خودمان نوشته ایم. قرار است اگر event (یعنی یک رویداد فشرده شدن کلید) برابر با فشردن کلید right arrow باشد، شئ bat را به سمت راست و اگر event برابر با فشردن کلید سمت چپ باشد، bat را به سمت چپ حرکت دهد.

بسیار خوب. حالا لطفا قطعه کدهای فوق را در جای مناسب استفاده کنید. کد نهایی را اجرا کنید و ببینید آیا می توانید با موفقیت راکت را به سمت راست و چپ حرکت دهید یا خیر.

- امتیازی 1: یک اشکالی در این پیاده سازی وجود دارد، آن هم اینکه اگر بیش از حد کلید سمت راست را فشار دهیم راکت از صفحه خارج می شود. فعلا ایرادی ندارد. اما اگر بتوانید در پاسخ نهایی خود این مشکل را رفع کنید، نمره امتیازی خواهد داشت.
- امتیازی 2: برای اینکه canvas فشرده شدن کلیدها را بشنود باید پنجره بازی، پنجره فعال ویندوز شما باشد. که به صورت پیش فرض نیست. اگر یک بار روی این پنجره کلیک کنید، تبدیل به پنجره فعال میشود و فشرده شدن کلیدها را می شنود و کار متناسب با آن را انجام می دهد.

البته یک راهی هم هست که پنجره بازی به صورت پیش فرض، به صورت فعال و روی سایر پنجره های ویندوز باز شود. اگر راهش را پیدا کردید، نمره امتیازی خواهد داشت.

حرکت کردن توپ (ball) روی صفحه

حرکت توپ باید به نحوی باشد که اگر به دیوارهای بالا و راست و چپ و همچنین راکت برخورد کند، در جهت مناسب برگردد و اگر به پایین سقوط کند، بازی تمام شود.

یک روش ساده برای شبیه سازی حرکت توپ در صفحه بازی را در قطعه کد زیر می بینید. در مورد بعضی از متودهایی که توضیح نداده ایم، جست و جو کنید و طرز کار آنها را یاد بگیرید.

(پیشنهاد: به قطعه کد زیر نگاه نکنید و خودتان بنویسید)

▼ نمایش قطعه کد

```
# move ball randomly
deltax = 1
deltay = 1
game_over = False
def move_ball():
    global deltax , deltay, game_over
    canvas.move(ball, deltax, deltay)

    ball_pos=canvas.coords(ball)
    bat_pos=canvas.coords(bat)

    if ball_pos[2] >= bat_pos[0] and ball_pos[0] <= bat_pos[2]:
        if ball_pos[3] >= bat_pos[1] and ball_pos[3] <= bat_pos[3]:
            deltay = -1
    if ball_pos[0] <= 0:
        deltax = 1
    if ball_pos[2] >= 500:
        deltax = -1
    if ball_pos[1] <= 0:
        deltay = 1
    if ball_pos[3] >= 320:
        messagebox.showinfo(title = "GAVE OVER", message = "you Lost")
        gave_over = True
        root.destroy()

```

```
    if not game_over:
        canvas.after(5, move_ball)
```

صورت مساله

تا اینجا بازی پونگ را در یک حالت ساده نوشته ایم. حالا شما باید بتوانید با رویکرد برنامه نویسی شئ گرا راه حل ارائه شده را بازنویسی کنید. سعی کنید موجودیت هایی را که در فضای مسئله ایفای نقش می کنند معین کنید (مثلا، صفحه بازی، توپ، راکت یا ...) و خصوصیات و رفتارهای هرکدام را فهرست کنید و سعی کنید بازی را یک بار دیگر به روش شئ گرا پیاده سازی کنید.

بخش‌های امتیازی:

۱. سعی کنید بازی را به صورت دو نفره طراحی کنید. یعنی دو راکت در بازی وجود داشته باشد؛ به نحوی که تعداد instance هایی که از کلاس راکت ساخته اید 2 تا باشد.
۲. سعی کنید بازی را به صورت یک نفره و با تعداد توپ های مختلف طراحی کنید؛ یعنی تعداد instance های کلاس توپ بیش از یک باشد.
۳. برای بازی یک منو تعریف کنید که امکان شروع و توقف بازی را در هر لحظه از بازی فراهم کند.
۴. امتیاز هر راکت (یا بازیکن) را حساب کنید و با درخواست کاربر آن را نمایش دهید.
۵. و هر آپشن دیگری که دوست می دارید.

- این موارد امتیازی صرفا در صورتی که به صورت شی گرا پیاده سازی شوند مد نظر قرار خواهند گرفت.

نحوه امتیازدهی

۱. پیاده سازی پونگ بدون شئ گرایی با توضیحات ارائه شده: 10 نمره
۲. پیاده سازی به صورت شئ گرا: 100 نمره
۳. پیاده سازی هر یک از موارد امتیازی: 10 نمره

- از حالتهای 1 و 2 یکی را انتخاب کنید.

حتما در ابتدای کد آپشن های بازی را در یک کامنت چندخطی لیست کنید. به میزان کافی از docstring و کامنت و ... استفاده کنید.