

بخش اول تمرین

مطابق موارد مطرح شده در صورت سوال، ابتدا در پوشه utils سه تابع مدنظر ایجاد شدند.

I. فایل random_number حاوی یک تابع است که به کمک دستور np.random.randint(0, 101) یک عدد صحیح مابین 0 تا 100 تولید می‌کند.

نکته: به منظور تولید عدد صحیح رندوم در بازی (a,b) با random.randint می‌بایست b+1 در آرگمان دوم تابع درج شود.

نکته: به کمک همین دستور این امکان وجود داشت که با وارد کردن آرگمان دیگر به جای یک عدد رندوم، آرایه‌ای از آن‌ها تولید کرد اما از آنجا که در صورت سوال درج شده بود تابع دوم آرایه‌ای از اعداد را ایجاد کند، این کار انجام نشد.

II. فایل make_array آرایه‌ای از اعداد رندوم تولید می‌کند. این تابع به کمک فراخوانی تابع قبل و البته استفاده از یک حلقه ایجاد می‌شود ولی از آنجا که باید اعداد تولیدی به صورت numpy باشند از np.array قبل از list comprehensive استفاده می‌شود.

```
random_array = np.array([random_number.generate_random_number() for _ in range(length)])
```

نکته: تابع استفاده شده در این مرحله دارای یک آرگمان ورودی است (length). این ورودی تعیین‌کننده طول آرایه تولیدی است که کاربر می‌بایست هنگام فراخوانی تابع وارد کند.

III. فایل identify نیز دارای یک آرگمان ورودی است که به موجب آن آرایه‌ای را دریافت می‌کند سپس به کمک دستور np.max(array) مقدار ماکزیمم آرایه را تعیین می‌کند و در گام بعدی به کمک یک if و else به راحتی برنده یا بازنده شدن کاربر را مشخص می‌کند.

IV. در گام بعدی نوبت به تهیه فایل app.py رسید. این فایل دربردارنده توابع ساخته شده در گام‌های قبلی است. فقط می‌بایست آن‌ها را فراخوانی نمود.

نکته: به منظور فراخوانی توابع موجود در پوشه utils در فایل app.py باید آن‌ها را در ابتدا import نمود و سپس در کد آن‌ها را فراخوان کرد در غیر اینصورت با خطا مواجه می‌شویم.

در ادامه به منظور تهیه فایل exe تصمیم گرفتیم از فونت Comic Sans MS بهره ببریم که برای setup مد نظر خودم یک لیبل برای عنوان بازی، یک لیبل برای درخواست از کاربر جهت ورود عدد، یک فیلد ورودی جهت دریافت ورودی از کاربر، و دو دکمه جهت محاسبه برنامه و خروج از آن استفاده کردم.

اما بخش حائز اهمیت در app.py فراخوانی تابع‌های نوشته شده بود. که آن‌ها را مطابق ترتیب زیر استفاده کردم:

<code>length = int(entry.get())</code>	در گام اول کاربر طول آرایه را تعیین می کند.
<code>make_array.create_random_array(length)</code>	در گام دوم به کمک تابع دوم (که این تابع خود به کمک تابع اول نوشته شده است) مطابق دستور روبرو آرایه ای از اعداد رندوم تولید شد.
<code>result_message = identify.win_or_fail(random_result)</code>	نهایتاً در گام بعدی این آرایه وارد تابع تعیین شکست و پیروزی شد تا نتیجه بازی مشخص شود.

نکته: اما به منظور انجام عملیات تولید عدد و چک کردن پیروزی یا شکست، می بایست کدهای بخش بالا در یک تابع `check` نوشته شوند تا در قسمت `command` برای `tkinter.Button` مورد استفاده قرار گیرند.

نکته: از آنجا که کاربر ممکن است عدد غیر معتبر یا رشته وارد کند، در قسمت `try` و `except` کدهای مربوط به تابع `check` نوشته می شوند.

بخش دوم تمرین - extra

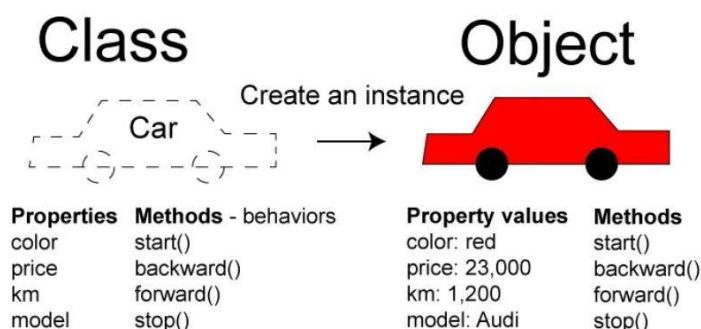
به منظور استفاده از برنامه نویسی شی گرا (OOP) مراحل زیر طی شد:

I. در گام نخست در فایل `game.py` توابع نوشته شده در مرحله قبل را دوباره کد کردم. همچنین به عنوان تمرین برای خود از `__init__` به عنوان یک magic method استفاده کردم. `__init__` به مقدارهی اولیه به هنگام استفاده از قابلیت شی گرایی کمک می کند. لازم به ذکر است که نام کلاس انتخاب شده `NumpyGame` است که متد آرایه ساز آن از کاربر یک ورودی می گیرد (`length`) و متد مربوط به تعیین برنده یا بازنده بودن، آرگمان ورودی آن یک آرایه است که از خروجی متد آرایه ساز حاصل می شود.

نکته: به طور کلی magic method ها به انعطاف پذیری برنامه نویسی در حالت شی گرایی کمک زیادی می کنند.

II. در این بخش از آنجا که ممکن است حضور `self` در برنامه نویسی شی گرا مورد سوال قرار گیرد، لازم دیدم توضیحاتی در این زمینه بیان کنم. وقتی ما از برنامه نویسی شی گرا استفاده می کنیم، پارامتر `self` برای دسترسی و تغییر `attribute` و `method` یک شیء درون کلاس مورد استفاده قرار می گیرد. وقتی یک `method` بر روی یک نمونه از یک کلاس فراخوانی می شود، نمونه خود به صورت ضمنی به عنوان اولین آرگومان منتقل می شود، که امکان دسترسی به داده ها و رفتار خاص نمونه را فراهم می کند.

نکته: در این بخش نیز تصویر زیر که برگرفته از سایت medium است به خوبی بیانگر تفاوت attribute و method در کلاس‌هاست.



همانطور که از شکل پیداست method ها بیانگر رفتار نمونه (behavior) یا همان function ها هستند. به عبارتی با فراخوانی و اجرای آن‌ها یک سری محاسبات انجام می‌شود که در کد با پرانتز همراه می‌شوند اما attribute ها بیانگر خواص نمونه (properties) هستند که بدون پرانتز هستند و با فراخوانی و اجرای آن‌ها محاسبات خاصی انجام نمی‌شود بلکه بیان‌کننده یک مشخصه از شی هستند. از آنجا که در تمرین تاکید بر docstring شده بود این مورد را برای تک تک توابع درون کلاس اعمال کردم که به موجب آن توضیح مختصر از هر کدام از توابع، آرگمان‌ها و خروجی آن‌ها ذکر شده است. **نکته:** docstring ها به عنوان اولین عبارت در تعریف توابع، کلاس‌ها و... مورد استفاده قرار می‌گیرند. از جمله مزایای آن‌ها می‌توان به درک بهتر کد یا خوانایی آن نام برد که هم برای خود شخص در آینده و یا کسانی که از کد استفاده می‌کنند، مفید است.

نکته: docstring ها را درون "..." می‌توان قرار داد. همچنین برای نمایش آن‌ها در خروجی از چنین دستوری می‌توان استفاده کرد:

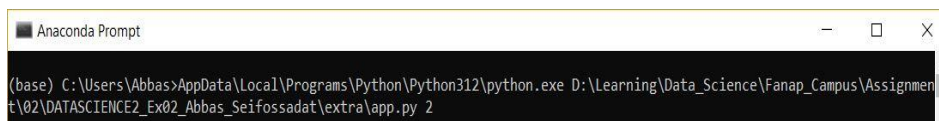
```
print(class.method_or_attribute.__doc__)
```

IV. فایل app.py در پوشه extra بیانگر استفاده از کلاس ساخته شده است. برای این منظور ابتدا یک شی به نام analyzer از کلاس گرفته می‌شود. سپس از این کلاس method مربوط به تولید آرایه‌ای از اعداد رندوم فراخوانی می‌شود که به موجب آن پس از اخذ length از کاربر، آرایه نامپایی با طول مدنظر ساخته می‌شود. در گام بعد از این کلاس، method مربوط به بررسی برنده یا بازنده شدن فراخوانی می‌شود و آرایه ساخته شده در خط قبل به آن اعمال می‌شود.

نکته: لازم به ذکر است چند خط کد مربوط به استفاده از method های کلاس جهت سهولت و برقراری کد تمیز درون یک تابع قرار داده می‌شوند و سپس به argparse اعمال می‌شوند.

V. در این بخش نکات و چالش‌های argparse بیان می‌شوند. به طور معمول argparse زمانی مورد استفاده قرار می‌گیرد که بخواهیم مستقیماً از command-line یا محیط terminal برنامه‌ها را ران کنیم یا به آن‌ها دسترسی داشته باشیم. مهمترین مزایای argparse عبارتند از:

- یک راه ساده و خوانا برای آنالیز آرگمان‌های command-line می‌دهد.
- بطور خودکار آرگمان‌های ورودی command-line را به انواع مشخص شده تبدیل می‌کند و نیاز به بررسی دستی تایپ و نوع ورودی را کاهش می‌دهد.
- **نکته:** مطابق این مزیت یاد شده، پس از استفاده از argparse دیگر نیازی به استفاده از try و except نداریم پس اگر کاربر، ورودی نامعتبر وارد کند اینجا به طور اتوماتیک توسط argparse این امر بررسی می‌شود.
- علاوه بر موارد فوق، argparse یک پیام راهنمای داخلی تولید می‌کند و استفاده از اسکریپت یا برنامه را ساده می‌کند. (help)
- **نکته:** برای اجرای برنامه در command-line یا terminal می‌بایست دستوری بدهیم که به موجب آن app.py توسط پایتون اجرا شود. برای این منظور کافی است ابتدا آدرس پایتون نصب شده در سیستم عامل و سپس آدرس فایل app.py را بدهیم. به طور مثال اگر بخواهیم برنامه app.py را با مقدار ورودی 2 (یعنی آرایه‌ای با دو عدد رندوم) اجرا کنیم کافی است چنین دستوری در محیط terminal اعمال کنیم:



```
Anaconda Prompt
(base) C:\Users\Abbas\AppData\Local\Programs\Python\Python312\python.exe D:\Learning\Data_Science\Fanap_Campus\Assignment02\DATASCIENCE2_Ex02_Abbas_Seifossadat\extra\app.py 2
```

البته که لازم به ذکر است چنین آدرسی برای سیستم من است که در چنین مسیرهایی پایتون و البته خود فایل app.py را دارم. همچنین اگر بخواهیم از help در argparse استفاده کنیم کافی است در انتها دستور app.py -h بنویسیم.

VI. در قسمت قبل به مزایا و نحوه اجرای برنامه با argparse صحبت شد. اما برای نحوه اعمال آن به کد مطابق زیر عمل شد:

- در گام اول باید ماژول argparse ایمپورت شود.
- سپس باید یک parser از کلاس ArgumentParser() از ماژول argparse ایجاد کنیم. آرگمان description یک ورودی برای این کلاس است که اگر برنامه را با help ران کردیم یک توصیف کلی از برنامه به ما می‌دهد.
- حال برای ترکیب app.py و argparse کافی است که ورودی app.py (طول آرایه تولیدی از اعداد رندوم) را درون add_argument() قرار دهیم و به parser اضافه کنیم. تایپ این ورودی به عنوان int تعریف می‌شود. لذا اگر کاربر عدد نامعتبر وارد کند به کمک این بخش از parser این امر قابل تشخیص است. همچنین می‌توان به ورودی help آن توضیحاتی اضافه کرد تا در صورت اجرای app.py -h به کاربر نمایش داده شود. کد زیر بیانگر این امر است:

```
parser.add_argument('length', type=int, help='Length of the random array')
```

- حال آرگمان parse شده باید داخل متغیر args منتقل شوند. این امر مطابق دستور زیر انجام می‌شود:

```
args = parser.parse_args()
```

- حال به راحتی از طریق `args.length` به ورودی کاربر دسترسی داریم و آن را می‌توانیم به عنوان ورودی به متد بررسی برنده یا بازنده شدن در کد اعمال کنیم.

نکته: معمولاً مرسوم است `argparse` را درون یک `if` مطابق کد زیر قرار دهند:

```
if __name__ == "__main__":
```

دلیل این است که اطمینان حاصل کنیم که کد داخل این بلاک در زمان وارد شدن فایل به عنوان یک ماژول در یک اسکریپت دیگر اجرا نشود. زمانی که یک فایل پایتون به صورت مستقیم (به عنوان برنامه اصلی) اجرا می‌شود، متغیر ویژه `__name__` به `__main__` تنظیم می‌شود. اگر فایل به عنوان یک ماژول در یک اسکریپت دیگر وارد شود، متغیر `__name__` به نام ماژول تنظیم می‌شود.