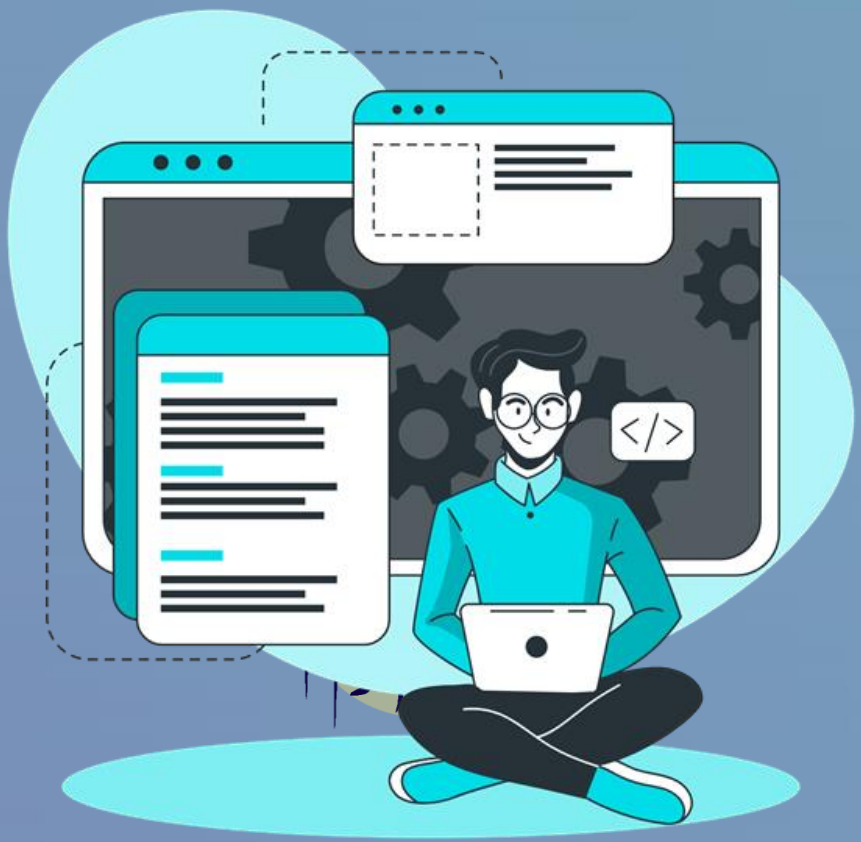
مکتب شریف

اولین بوتکمپ آموزشی - استخدامی ایران

PYTHON BOOTCAMP

PRACTIC 3



1. در دیجی‌کالا برای ذخیره‌سازی برخی از رشته‌های عددی از نوعی فشرده‌سازی استفاده می‌شود تا کسی نتواند از خروجی تولید شده به رشته‌ی عددی اصلی دست پیدا کند.

1.1. البته ما در اینجا از این نکته که این روش تصادم دارد؛ به این معنا که چند ورودی مختلف ممکن است خروجی یکسانی تولید کنند؛ چشم پوشی می‌کنیم ! لازم به ذکر است که رشته‌ی عددی فقط شامل ارقام ۰ تا ۹ است.

1.2. الگوریتم به این صورت است که تعداد تکرار همه‌ی ارقام رشته را حساب می‌کند، ارقام تکراری را حذف می‌کند و تعداد تکرار هر رقم (با شرط این که حداقل ۲ بار تکرار شده باشد) را در رشته‌ی ورودی می‌نویسد و در نهایت رشته‌ی عددی را به صورت صعودی مرتب می‌کند.

1.3. این کار روی خروجی به دست آمده مجدد تکرار می‌شود و آنقدر ادامه دارد تا خروجی نهایی با خروجی مرحله‌ی قبل تفاوتی نکند.

تذکر: تابع عملیات بالا باید پیاده سازی و ایجاد شود.

ورودی نمونه و خروجی نمونه :

تنها عبارت ورودی رشته عددی مورد نظر است که می‌خواهیم آن را فشرده کنیم.

طول این رشته حداقل یک و حداکثر 1000 رقم است.

تنها عبارت خروجی، عبارت فشرده شده نهایی است.

```
...
enter digit string : 442254545
22345
>>>
```

توضیح:

در این مثال رقم 4، 4 بار تکرار شده است و رقم‌های 2 و 5 هم به ترتیب 2 و 3 بار تکرار شده‌اند. رقم‌های تکراری حذف می‌شوند و فقط یکی از آن‌ها در رشته باقی می‌ماند، بنابراین رشته‌ی 425 باقی می‌ماند. سپس تعداد تکرار هر رقم در ادامه‌ی رشته نوشته می‌شود، بنابراین رشته‌ی 425423 ایجاد می‌شود و در نهایت ارقام به صورت صعودی مرتب می‌شوند که در این‌جا 223445 حاصل می‌شود.

مجدداً عملیات فشرده‌سازی روی این رشته‌ی حاصل شده اعمال می‌شود و نتیجه‌ی آن 222345 می‌شود.یک بار دیگر عملیات فشرده‌سازی اعمال می‌شود و نتیجه‌ی آن 23345 می‌شود و با اعمال مجدد این الگوریتم خروجی 22345 حاصل می‌شود که دیگر قابل فشرده‌سازی نیست.

2. یک برنامه بنویسید که سبد خرید یک فروشگاه آنلاین ساده را شبیه سازی کند که شامل افزودن محصول، حذف محصول، محاسبه‌ی قیمت کل سبد خرید یک مشتری و همینطور نمایش سبد خرید مشتری باشد .

2.1. از نام مشتری برای ایجاد سبد خرید مربوط به مشتری استفاده کنید . به طوری که اگر دیکشنری مربوط به مشتری وجود داشت آن را به‌روز کرده و در غیر این صورت آن را ایجاد کند.

2.2. لازم به ذکر است که در مواقع خواندن سبد خرید مشتری، در صورت عدم وجود این سبد خرید از exception handling مناسب استفاده کنید.

2.3. از انجایی که هیچ ایده‌ای برای ذخیره سازی دیتا ندارید . از حلقه بینهایت استفاده کنید که پس از ساخت دیتا برنامه متوقف نشود.

```
>>> add_item(Hossein , 'laptop' , 1300 , 2)
2 laptop added to Hossein Cart lists thats equal to : 2600$
>>>
```

2.4. برای توابع نام برده شده از فرمتی شبیه به بالا با تعداد ورودی‌های کمتر یا بیشتر استفاده کنید. که در مثال بالا، ورودی اول نام مشتری که برابر است با نام دیکشنری سبد خرید، ورودی دوم نام محصول، ورودی سوم قیمت و ورودی چهارم تعداد محصول بوده که می‌تواند مقدار پیش‌فرض داشته باشد.

3. تصور کنید یک فایل csv از نام و نمرات دانش آموزان یک کلاس داریم.

3.1. فایل csv را خوانده و دیتای آن را در سه دیکشنری جداگانه بریزید.

3.2. یک دیکشنری شامل نام و نمره تمامی دانش آموزان می‌باشد.

3.3. یک دیکشنری شامل نام و نمره تمامی دانش آموزانی که نمره‌ی آن‌ها بین 17 تا 20 می‌باشد.

3.4. یک دیکشنری شامل نام و نمره تمامی دانش آموزانی که نمره ی آن ها بین 0 تا 10 است.

3.5. در اخر نیز یک تابع search تعریف کرده تا با گرفتن نام دانش‌آموز نمره‌ی آن را با پیام مناسب برگرداند.

```
...
>>>
>>> search('Ava')
you pass with 19 Ava, congratulate
>>>
```

4. ابتدا تابعی برای برگرداندن تمامی کلمات یکتای یک جمله بنویسید ؛ سپس بدون استفاده از حلقه‌ها، این تابع را روی لیستی از جملات اعمال کرده و لیستی جدید تشکیل دهید.

```
>>>
>>> input_sentences = [
...     "This is a sample sentence.",
...     "Python programming is fun."
... ]
>>>
>>> unique_sorted_words = get_unique_sorted_words(input_sentences)
>>> print(unique_sorted_words)
['Python', 'This', 'a', 'fun.', 'is', 'programming', 'sample', 'sentence.']
>>>
```

5. کش کردن(Cache) یکی از راههای مرسوم برای ذخیره انرژی و سرعت بخشیدن به کد است؛ الگوریتم آن بدین صورت است دیتاهای قبلی و نتیجه را ذخیره می‌کند و هنگام برخورد با دیتای تکراری به جای محاسبه دوباره آن، با رجوع به محل ذخیره‌سازی (فایل و یا...) نتیجه را برای ما برمی‌گرداند.

پس از مطالعه و تحقیق بیشتر در مورد این موضوع ادامه تمرینات زیر را ادامه بدهید و بنویسید.

ابتدا توابع فیبوناچی و فاکتوریل را با الگوریتم بازگشتی پیاده‌سازی کرده و دکوراتور timer_process را بطوریکه زمان اجرای تابع را لاگ بیندازد را برای آن قرار داده و آن را برای اعداد بزرگ ثبت نمایید . توجه کنید که ادرس فایل لاگ و همینطور عدد مورد نظر را به عنوان ورودی برای این تابع در نظر بگیرید .

حال دکوراتوری به نام cache بنویسید که همانطور که در بالا گفته شد نتایج را برای توابع ما کش کند؛ سپس توابع را مجددا با همان اعداد تست شده در مرحله پیشین اجرا کرده و لاگ زمانی ثبت شده را مقایسه نمایید.

راهنمایی :درمورد توالی دکوراتورها(Decorators) و ترتیب اجرای آنها در پایتون تحقیق نمایید.

نکات :

- مهلت ارسال تمرین تا **پایان ساعت ۲۴ روز چهارشنبه** می‌باشد.
- نام فایل ارسالی خود را به این صورت قرار دهید. به عنوان مثال:
  - Firstname_Lastname_HWNumber_maktabNumber
  - Mohammad_Ali_Kargar_hw1_maktab100
  - دقت فرمایید که منظور از Number - بعد از HW شماره تمرین می‌باشد.
  - دقت فرمایید که منظور از Number - بعد از Maktab شماره بوتکمپ می‌باشد.
- در مورد تمرین‌های پایتون هر تمرین را در یک فایل جداگانه پایتون با پسوند py بنویسید و از ارسال فایل Jupyter یا مشابه آن خودداری کنید.
- در صورت یکه تمرین شامل چند فایل و فولدر می‌باشد حتماً آنها را در قالب یک فایل فشرده شده تجمیع کنید(ZIP/ RAR).
- در صورت لزوم یک فایل Word به عنوان توضیح در کنار کدهای خود قرار دهید.
- در صورت یکه سوالی دارید در کارتابل گروهی خود از مربیان بپرسید.
- **توصیه دوستانه.**از مواجهه با هیچ سوالی نترسید.به هر میزانی که در حل سوالات پیشروی کرده باشید نمره بخش مورد نظر را دریافت می‌کنید.بنابراین بیش از آن‌که رسیدن به خروجی نهایی مهم باشد، تلاش شما ارزشمندتر است.
- قطعا هدف از تمارین صرفا رسیدن به جواب نهایی نیست و تمیز بودن کد و خلاقیتی که در انجام آن به خرج می دهید از اهمیت و امتیاز بالایی برخوردار است. ارائه راه حل کلی و عمومی برای یک مسئله که حالت‌های مختلف آن را در نظر بگیرد و فراتر از خواسته‌ی مسئله است (خواسته‌ی مسئله گسترش داده شود یا حالت‌های خاص مسئله را پوشش دهد، قطعا مشمول امتیاز بیشتری خواهد شد.)
- سوالات امتیازی شامل مواردی است که نیازمند سرچ بیشتر شما عزیزان می‌باشد. بنابراین حل این سوالات نمره امتیازی دارد.

موفق باشید