

به نام خداوند علم و دانش



دانشکده مهندسی کامپیوتر

برنامه نویسی پیشرفته (پایتون)

تمرین سوم (oop)

دکتر مرضیه داودآبادی

زمستان 1403

طراحان تمرین: آقایان پارسا زرگر و محمد مردی و رامین کلانتری

- در صورت وجود هرگونه ابهام به طراح پیام دهید.
  - باتوجه به وجود تاخیر 10 روزه، امکان تاخیر تحت هیچ شرایطی امکان پذیر نیست.
  - انجام تمرین ها تک نفره می باشد.
  - زبان برنامه نویسی پایتون است.
  - موارد ارسال شده به صورت آنلاین تحویل گرفته خواهند شد.
  - ددلاین تمرین: 18 فروردین ساعت 23:59
  - برای دیدن تست کیس های نمونه متنوع به کوئرای درس بپیوندید.
  - [/https://quera.org/course/add to course/course/20444](https://quera.org/course/add to course/course/20444)
  - Password: ap4032
  - لینک تلگرام طراحان:
  - <https://telegram.me/parsazargar> | آقای پارسا زرگر
  - <https://telegram.me/MohamadMardi> | آقای محمد مردی
  - <https://telegram.me/jjirt1> | آقای رامین کلانتری
- 
-

## 01. محاسبه ویژگی‌های هندسی (مردی)

برنامه‌ای شی‌گرا بنویسید که بتواند اطلاعات مربوط به اشکال هندسی دایره، مستطیل و مثلث را مدیریت کند. این برنامه باید از اصول وراثت (Inheritance) و چندریختی (Polymorphism) استفاده کند تا محاسبات مربوط به مساحت و محیط را انجام دهد.

مواردی که باید رعایت شوند:

1. تعریف کلاس پایه:

- کلاسی به نام Shape تعریف کنید که شامل دو متد `calculate_area()` و `calculate_perimeter()` باشد.

- این متدها در کلاس پایه به صورت `abstract` یا `pass` تعریف شوند و در کلاس‌های فرزند پیاده‌سازی شوند.

2. ایجاد کلاس‌های فرزند:

- سه کلاس `Circle`، `Rectangle` و `Triangle` از کلاس Shape مشتق شوند.

- هر کلاس باید متدهای `calculate_area()` و `calculate_perimeter()` را بر اساس فرمول‌های مربوط به هر شکل پیاده‌سازی کند.

3. استفاده از اصول شی‌گرایی:

- از وراثت برای جلوگیری از تکرار کدها استفاده شود.



- از چندریختی (Polymorphism) استفاده کنید تا متدهای `calculate_area()` و `calculate_perimeter()` در کلاس‌های فرزند، رفتار خاص خود را داشته باشند.

#### 4. ورودی از کاربر:

- ابتدا نوع شکل (دایره، مستطیل یا مثلث) از کاربر دریافت شود.
- سپس مقادیر مربوط به آن شکل دریافت شود.
- خروجی شامل مساحت و محیط شکل باشد.

#### 5. فرمول‌های محاسبه:

- دایره:

$$\pi * r^2 = \text{مساحت}$$

$$2 * \pi * r = \text{محیط}$$

- مستطیل:

$$\text{length} * \text{width} = \text{مساحت}$$

$$2 * (\text{length} + \text{width}) = \text{محیط}$$

- مثلث (با استفاده از فرمول هرون برای مساحت):

$$\text{sqrt}(s * (s - a) * (s - b) * (s - c)) = \text{مساحت}$$



$$s = (a + b + c) / 2 \text{ (که در آن)}$$

$$a + b + c = \text{محیط}$$

### ورودی

ورودی تنها شامل یک خط است که ابتدا نوع شکل و سپس پارامترهای موردنیاز آن آورده شده است.

### ♦ قالب ورودی:

- اگر شکل دایره (Circle) باشد، یک عدد شامل شعاع  $r$  دریافت می شود.

Circle  $r$

اگر شکل مستطیل (Rectangle) باشد، دو عدد شامل طول  $l$  و عرض  $w$  دریافت می شود.

Rectangle  $l$   $w$

اگر شکل مثلث (Triangle) باشد، سه عدد شامل سه ضلع  $side1$   $side2$   $side3$  دریافت می شود.

Triangle  $side1$   $side2$   $side3$

### خروجی

خروجی برنامه ۲ خط است:

1. مساحت شکل با ۶ رقم اعشار

2. محیط شکل با ۶ رقم اعشار

🚀 نکات:



- اعداد خروجی دقیقاً ۶ رقم اعشار داشته باشند.

- هیچ متن اضافی نباید چاپ شود!

## 02. عملیات منطقی (مردی)

### هدف تمرین:

هدف این تمرین آشنایی بیشتر با مفاهیم شی گرای و استفاده از کلاس‌ها و متدهای مختلف برای انجام عملیات منطقی است. شما باید یک سیستم ساده برای انجام عملیات منطقی با استفاده از کلاس‌ها پیاده سازی کنید و سپس با استفاده از دستورات ورودی، آن‌ها را ارزیابی کنید.

### شرح تمرین:

در این تمرین شما باید دو کلاس زیر را پیاده سازی کنید:

### کلاس اول Boolean :

این کلاس برای انجام عملیات منطقی مختلف تعریف می شود. هر شی از این کلاس نمایانگر یک مقدار منطقی ( True یا False ) است و باید متدهایی برای انجام عملیات منطقی مختلف (مثل AND ، OR ، XOR و غیره) داشته باشد.

### ویژگی ها و متدها:

1. ویژگی value : این ویژگی مقدار منطقی شی را ذخیره می کند که می تواند True یا False باشد.

2. متدهای عملیات منطقی:



○ And : پیاده سازی عملگر AND

○ Nand : پیاده سازی عملگر NAND

○ Or : پیاده سازی عملگر OR

○ Nor : پیاده سازی عملگر NOR

○ Not : پیاده سازی عملگر NOT

○ Xor : پیاده سازی عملگر XOR

○ Xnor : پیاده سازی عملگر XNOR

3. متدهای مقایسه و نمایش:

○ \_\_eq\_\_: برای مقایسه دو شی Boolean با هم.

○ \_\_str\_\_: برای نمایش مقدار شی به صورت رشته.

### کلاس دوم BooleanInterpreter :

این کلاس مسئول پردازش دستورات ورودی است. دستورات شامل تعریف متغیرها، انجام عملیات منطقی و چاپ نتایج هستند. این کلاس باید قابلیت ارزیابی و اجرای دستورات منطقی را داشته باشد.

### ویژگی ها و متدها:

1. ویژگی `var_dict`: یک دیکشنری برای ذخیره سازی متغیرهای منطقی که توسط دستور Define

تعریف می شوند.

2. متد **process\_statement** : این متد مسئول پردازش دستورات ورودی است. دستورات ورودی

شامل سه نوع اصلی هستند:

○ **Define** : برای تعریف یک متغیر منطقی (مثال : Define X True)

○ **Let** : برای انجام عملیات منطقی روی متغیرهای موجود و ذخیره نتیجه در یک متغیر جدید

(مثال : Let A = and(X, Y))

○ **Print** : برای چاپ مقدار یک متغیر منطقی (مثال : Print A)

### ورودی

ورودی شامل چند دستور است که هر کدام باید توسط کلاس **BooleanInterpreter** پردازش شود. اولین خط ورودی عدد  $n$  است که تعداد دستورات ورودی را نشان می‌دهد. سپس  $n$  خط دستور وارد می‌شود که می‌تواند شامل دستورات **Define**, **Let** یا **Print** باشد.

### خروجی

دستورات **Print** باید خروجی منطقی متغیر مورد نظر را به صورت **True** یا **False** چاپ کنند. به این توجه داشته باشید که خروجی‌ها باید در آخر کار و پس از اتمام دریافت ورودی چاپ شوند.

## 03. مدیریت فروشگاه آنلاین (مردی)

شما به عنوان توسعه‌دهنده یک سیستم ساده برای مدیریت فروشگاه آنلاین وظیفه دارید سیستمی طراحی کنید که بتواند محصولات مختلف را مدیریت کند، موجودی هر محصول را تغییر دهد و امکان خرید و ثبت





سفارش برای کاربران را فراهم نماید. همچنین کاربران باید بتوانند اعتبار خود را افزایش دهند و از آن برای خرید محصولات استفاده کنند.

الزامات و وظایف سیستم:

### 1. مدیریت محصولات:

✓ هر محصول دارای ویژگی‌های زیر است:

- product\_id (شناسه منحصر به فرد محصول)
- product\_name (نام محصول)
- price (قیمت هر واحد محصول)
- count (تعداد موجودی اولیه محصول)

✓ عملیات مرتبط با محصولات:

- اضافه کردن محصول جدید به سیستم.
- افزایش موجودی محصول در صورت نیاز.
- فروش محصول و کاهش موجودی هنگام ثبت سفارش.

### 2. مدیریت کاربران:

✓ هر کاربر دارای ویژگی‌های زیر است:

- user\_id (شناسه منحصر به فرد کاربر)

- username (نام کاربری)
- credit (میزان اعتبار مالی کاربر)
- orders (لیست سفارشات ثبت شده توسط کاربر)

✓ عملیات مرتبط با کاربران:

- ثبت کاربر جدید در سیستم.
- افزایش اعتبار مالی کاربر برای خرید محصولات.

### 3. مدیریت سفارشات:

✓ هر سفارش شامل اطلاعات زیر است:

- order\_id (شناسه منحصر به فرد سفارش)
- user\_id (کاربری که سفارش را ثبت کرده است)
- products (لیستی از محصولات سفارش داده شده)
- product\_counts (تعداد هر محصول در سفارش)

✓ عملیات مرتبط با سفارشات:

- ثبت سفارش جدید برای یک کاربر.
- بررسی اعتبار مالی کاربر پیش از ثبت سفارش (در صورت کافی نبودن اعتبار، سفارش ثبت نمی شود).
- بررسی موجودی کافی محصولات (اگر محصولی به مقدار کافی در انبار نباشد، سفارش ثبت نمی شود).



- در صورت تأیید سفارش، کاهش موجودی محصولات و کاهش اعتبار کاربر.

### ورودی

1. دستوراتی که در طول اجرای برنامه وارد می‌شوند، به صورت زیر هستند:

- `register <user_id> <username>`: ثبت یک کاربر جدید.
- `add product <product_id> <product_name> <price> <count>`: اضافه کردن یک محصول جدید.
- `add stock <product_id> <count>`: افزایش موجودی یک محصول.
- `add credit <user_id> <credit>`: افزایش اعتبار یک کاربر.
- `order <order_id> <user_id> <n>`: ثبت یک سفارش برای کاربر خاص.
- `Exit`: برای خروج از برنامه.

2. در دستور `order <order_id> <user_id> <n>`، برای هر محصول در سفارش، باید ورودی به صورت زیر باشد:

- `<product_id> <count>`: شناسه محصول و تعداد آن محصول در سفارش.

### خروجی

برای هر دستور، سیستم باید پیام مناسبی را نمایش دهد، مانند:

- "User registered successfully"
- "Product added successfully"



- "Stock added successfully"
- "Credit added successfully"
- "Order completed successfully"
- پیغام خطا در صورت وجود به ترتیب : "Not" "product not found" "User not found" "Not enough stock" "Not enough credit"

## 04. حل مسئله ژوزفوس با شی گرای (زرگر)

سخت ترین سوالات ریاضی که برای حل آنها پاداش در نظر گرفته شده است ممکن در نگاه اول بسیار ساده به نظر بیایند؛ مسئله ژوزفوس بدنام یکی از همین دست مسائل در دنیای ریاضیات به شمار می رود.

این مسئله اسم خود را از نام تایتوس فلاویس ژوزفوس برگرفته که یک محقق یهودی در قرن اول بوده است. موضوع این مسئله از این قرار است که وقتی او توسط لشکر رم محاصره شده، با خود ۴۰ سرباز داشته است و سربازان به جای تسلیم شدن تصمیم گرفتند خودکشی دسته جمعی انجام دهند و این کار را به صورتی انجام دادند که افراد یکدیگر را می کشتند و هیچ کس خودکشی نمی کرد و این تصمیم را از آن جهت گرفتند تا مبادا در لحظات آخر فردی تصمیمش عوض شود و دست به خودکشی نزنند.

آنها یک حلقه زدند و اولین سرباز می بایست فرد سمت چپ خود را می کشت و سرباز زنده بعدی نیز باید فرد سمت چپ خود را می کشت و این رویه در کل حلقه باید ادامه پیدا می کرد.

وقتی حلقه مرگ به ابتدای شروع این کار می رسید، این فرایند می بایست با تعداد کمتری از افراد مجدداً انجام می گرفت و در نهایت آخرین فردی که زنده می ماند می بایست با شمشیر خودش، به زندگی خود پایان می داد.



در این بین ژوزفوس مشکلی داشت و مشکل این بود که او ترجیح می داد زنده بماند تا مثل بقیه به زندگی خود پایان دهد ولی از قراری که با بقیه سربازان نیز گذاشته بود گریزی نبود و در عین حال نیز نمی خواست هم رزمان خود از راز او مطلع شوند. به نظر شما او باید خود را در کدام قسمت این حلقه قرار می داد تا آخرین فردی باشد که هنوز زنده است؟

جواب جایگاه نوزدهم است. اما اینکه چطور به این عدد رسیدید و اینکه با تعداد مختلف سربازان به چه عددی می رسید مهم است؛ این دقیقاً کاری است که دنیل ارمان از دانشگاه ویسکانسین آن را تشریح کرده است. در دور اول این حلقه مشخص است هر فردی که در وضعیت جایگاه فرد قرار داشته باشد جان سالم به در برده است پس اگر می خواهید زنده بمانید در این دور، حتماً در جایگاه های فرد قرار بگیرید.

اما با شروع دور دوم و نفرات باقی مانده، افرادی که در جایگاه های زوج قرار گرفته اند همچنان سرنوشت مرگباری خواهند داشت.

الگوی مهمی که در این مسئله باید به آن توجه کنیم، این است که اگر تعداد سربازان دقیقاً توانی از ۲ باشند (۶۴، ۳۲، ۱۶، ۸، ۴، ۲، ...) جایگاه امنی که می توان در آن قرار گرفت همان جایگاه اول است چرا که همیشه شروع کننده حلقه مرگ او است. به مثال زیر توجه کنید:

دو نفر را در نظر بگیرید: فرد شماره ۱ قاعدتاً فرد شماره ۲ را می کشد.

۴ نفر را در نظر بگیرید: فرد شماره ۱، شماره ۲ را می کشد و فرد شماره ۳ شماره ۴ را می کشد و فرد شماره ۱ مجدداً فرد شماره ۳ را می کشد. وقتی تعداد نفرات توانی از ۲ باشند فرقی نمی کند که چه تعداد باشند چرا که فرد شماره یک همواره شروع کننده کشتار است.

**توضیح مسئله:**



یک حلقه‌ی دایره‌ای از  $n$  نفر داریم که هر کدام دارای یک شماره صندلی و وضعیت زنده یا مرده هستند. بازی تا زمانی ادامه دارد که فقط یک نفر زنده بماند. در هر مرحله، فرد بعدی حذف می‌شود. هدف این است که این مسئله را با استفاده از شی‌گرایی و وراثت در پایتون حل کنیم.

### دستورالعمل پیاده‌سازی:

1. یک کلاس `Person` تعریف کنید که شامل ویژگی‌های زیر باشد:
  - شماره صندلی : شماره‌ای که به هر فرد اختصاص داده می‌شود.
  - وضعیت حیات : یک مقدار بولی که نشان می‌دهد فرد زنده است یا نه.
2. یک کلاس `JosephusGame` ایجاد کنید که شامل ویژگی‌های زیر باشد:
  - لیستی از افراد که شامل نمونه‌هایی از کلاس `Person` است.
  - روش اجرای بازی که به ترتیب افراد را حذف کند تا تنها یک نفر باقی بماند.
3. از حلقه‌ها و لیست‌ها برای مدیریت حذف افراد و حفظ ترتیب چرخشی آن‌ها استفاده کنید.
4. خروجی باید شماره‌ی صندلی فرد نهایی که زنده مانده است را نمایش دهد.

## 05. سیستم سفارش غذا (زرگر)

شما قرار است یک سیستم مدیریت سفارش غذا برای یک رستوران طراحی کنید. در این رستوران، غذاها به دو دسته کلی تقسیم می‌شوند:

1. فست‌فودها که دارای ۲۰٪ تخفیف هستند. 2. غذاهای سالم که تخفیف ندارند.




اما مهمی غذاها شامل ۹٪ مالیات هستند که در قیمت نهایی لحاظ می‌شود. این برنامه باید از کاربر سفارش دریافت کرده، قیمت نهایی را با تخفیف و مالیات محاسبه کند و در نهایت یک فاکتور نمایش دهد.

✓ لیست غذاها و قیمت اولیه آنها:

🍔 فست‌فودها (۲۰٪ تخفیف + ۹٪ مالیات)

قیمت پایه (تومان)	نام غذا
۵۰,۰۰۰	برگر 🍔
۸۰,۰۰۰	پیتزا 🍕
۴۰,۰۰۰	هات‌داگ 🌭
۳۰,۰۰۰	سیب‌زمینی سرخ‌کرده 🍟



غذاهای سالم (۹٪ مالیات بدون تخفیف) 

نام غذا	قیمت پایه (تومان)
سالاد سزار 	۴۰,۰۰۰
سوپ جو 	۳۵,۰۰۰
ماهی کبابی 	۷۰,۰۰۰





نمونه لیست غذاها:

نام غذا	نوع	قیمت پایه (تومان)
Burger 🍔	Fast Food	50000
Pizza 🍕	Fast Food	80000
Fries 🍟	Fast Food	30000
Hotdog 🌭	Fast Food	40000
Caesar Salad 🥗	Healthy	40000
Barley Soup 🍲	Healthy	35000
Grilled Fish 🐟	Healthy	70000



### شرح مسئله:

۱. لیستی از غذاهای فست فود و سالم در برنامه تعریف شده است.
۲. هر غذا باید به عنوان یک شیء (Object) از یک کلاس ساخته شود و دارای نام و قیمت پایه باشد.
۳. غذاهای فست فود شامل ۲۰٪ تخفیف هستند اما مالیات ۹٪ نیز روی قیمت نهایی اعمال می شود.
۴. غذاهای سالم هیچ تخفیفی ندارند اما مالیات ۹٪ شامل آنها می شود.
۵. کاربر می تواند چندین غذا را به لیست سفارش خود اضافه کند.
۶. اگر کاربر ok یا end را وارد کند، برنامه مبلغ نهایی را محاسبه و نمایش دهد.
۷. اگر غذایی اشتباه وارد شد، پیام خطا نمایش داده شود.

✗ Item not found, please try again.

### وظایف شما:

#### ۱. تعریف کلاس ها:

- کلاس **Food** را به عنوان کلاس پایه (Parent Class) ایجاد کنید که شامل نام و قیمت پایه غذا باشد.
- دو کلاس فرزند از آن مشتق کنید:
  - **FastFood** برای غذاهای فست فودی (با تخفیف ۲۰٪)
  - **HealthyFood** برای غذاهای سالم (بدون تخفیف)

## ۲. پیاده‌سازی منطق قیمت‌گذاری:

- متدی در کلاس **Food** تعریف کنید که قیمت نهایی را با احتساب ۹٪ مالیات محاسبه کند.
- در کلاس **FastFood** این متد را بازنویسی (Override) کنید تا ۲۰٪ تخفیف قبل از محاسبه مالیات اعمال شود.

## ۳. ایجاد کلاس مدیریت سفارش:

- کلاس **Order** را پیاده‌سازی کنید که لیستی از غذاهای سفارش داده شده را نگه دارد.
- امکان افزودن غذاهای مختلف به سفارش فراهم شود.
- قیمت نهایی کل سفارش محاسبه و نمایش داده شود.

توجه: مهم! حتما لیست منو داشته باشید...

### 📌 لیست غذاها (منو):

در این برنامه، یک دیکشنری به نام **menu** تعریف شده که شامل تمامی غذاهای موجود در سیستم است. هر آیتم در **menu** یک **Object** از کلاس **FastFood** یا **HealthyFood** است.

## ۴. نمایش خروجی:

- لیست غذاهای سفارش داده شده نمایش داده شود.
- قیمت هر غذا همراه با تخفیف و مالیات نشان داده شود.
- مبلغ نهایی کل سفارش محاسبه و نمایش داده شود.

## 06.مسأله اختلاط (کلاتری)

در این سوال شما لازم است که دو کلاس Computations, ComplexNumbers را به گونه ای پیاده سازی کنید که ComplexNumbers به صورت:

```
class ComplexNumber:
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary
    def __add__(self, other):
        """
        این متد باید با توجه به متد Add از کلاس Computations
        پیاده سازی شود
        """
        pass
    def __sub__(self, other):
        """
        این متد باید با توجه به متد Subtract از کلاس Computations
        پیاده سازی شود
        """
        pass
    def __mul__(self, other):
        """
        این متد باید با توجه به متد Multiplication از
        کلاس Computation پیاده سازی شود
        """
        pass
    def __div__(self, other):
        """
        این متد باید با توجه به متد Division
        از کلاس Computations پیاده سازی شود
        """
        pass
    def __eq__(self, other):
```



این تابع باید برابری دو عدد مختلط را نشان دهد#

```
pass
```

```
def __str__(self):
```

```
    """استفاده از تابع str ممنوع می باشد"""
```

```
pass
```

و کلاس Computations نیز به صورت زیر پیاده سازی شود:

```
class Computations:
```

```
    def __init__(self,a,b):
```

```
        pass
```

```
    def Add(self):
```

```
        pass
```

```
    def Subtract(self):
```

```
        pass
```

```
    def Multiplication(self):
```

```
        این تابع باید بدون نماد * پیاده سازی شود#
```

```
        pass
```

```
    def Division(self):
```

```
        این تابع نیز باید بدون نماد / یا // پیاده#
```

```
        سازی شود#
```

```
        pass
```

```
        خروجی کل توابع فوق تا 2 رقم اعشار باشد#
```

## ورودی ها

در 2 خط اول دو عدد  $a, b$  به صورت مختلط وارد میشود که شامل به ترتیب قسمت حقیقی و موهومی عدد مختلط می باشند. قسمت های حقیقی و موهومی به صورت اعشاری تا 2 رقم اعشار می باشند. خط سوم شامل یکی از عملیات های زیر میباشد که با استفاده از کتابخانه random پایتون باید یکی از آنها را انتخاب کنید.

1.add

2.subtract

3.multiply

4.division

5.equals

در این لینک میتوانید درباره کتابخانه random مطالعه بفرمایید [پیوند](#)

## خروجی ها

خروجی شامل یک خط است که در صورت valid بودن باید عدد مختلط حاصل را نشان دهد. اگر عبارت equals فراخوانی شود، باید ، True / False برگردانده شود. در غیر این صورت باید پیام invalid operations entered را نشان دهد.

## 07. پیاده سازی دیتابیس کارخانه و تحلیل داده ها (زرگر)

در این تمرین، شما باید یک دیتابیس ساده از کارکنان یک کارخانه را پیاده سازی کنید که شامل چندین نوع کارمند باشد. هدف، تمرین روی ارث بری، دسته بندی داده ها، جستجو و نمایش اطلاعات است.



## ✓ مفاهیم کلیدی:

- ارث‌بری در کلاس‌ها
- پردازش داده‌ها با **Pandas**
- عملیات فیلتر و جستجو
- استفاده از **Matplotlib** برای نمایش اطلاعات

## 📌 شرح:

### 1. تعریف کلاس‌ها:

- کلاس **Employee** شامل اطلاعات پایه (نام، کد ملی، حقوق، دپارتمان).
- کلاس‌های فرعی:

- **Manager**: مدیران هر دپارتمان
- **Engineer**: مهندسان با تخصص مشخص
- **Worker**: کارکنان ساده بدون مهارت خاص


### 2. داده‌های اولیه:

- اطلاعات ۲۰ کارمند در قالب یک لیست از لیست‌ها ذخیره شده و باید در برنامه خوانده شود.

### 3. امکانات برنامه:

✓ نمایش تمام کارکنان در قالب جدول ✓ نمایش لیست کارکنان یک دپارتمان خاص ✓ پیدا کردن

مدیر هر کارمند ✓ نمایش بصری حقوق کارکنان یک دپارتمان

داده‌های اولیه: 

یک مینی دیتابیس در بخش راهنمایی 1 تحت عنوان data مشهود است که عملیات های گفته شده قرار است روی آن انجام شوند.

🌀 وظایف شما:

۱. پیاده‌سازی کلاس‌های **Worker** ، **Engineer** ، **Manager** ، **Employee** با ویژگی‌های مناسب.

۲. پیاده‌سازی کلاس **FactoryDatabase** برای مدیریت کارکنان.

۳. بارگذاری داده‌ها از data

۴. پیاده‌سازی توابع جستجو و نمایش اطلاعات.

۵. رسم نمودار حقوق کارکنان یک دپارتمان.

✓ دستورات پیشنهادی برای اجرا در برنامه:

```
db = FactoryDatabase()
```

```
db.show_all() # نمایش تمام کارکنان
```

```
db.get_manager_of("Sara") # پیدا کردن مدیر یک کارمند
```

```
db.show_department("IT") # نمایش کارکنان یک دپارتمان به صورت نمودار
```



```
class FactoryDatabase:

    def __init__(self):

        self.employees = []

        self.load_data()

        # Load data

    def load_data(self):

        data = [

            ["Ali", 1234, 15000, "IT", "Manager", None],

            ["Sara", 5678, 12000, "IT", "Engineer", "Software"],

            ["Hamed", 9876, 10000, "IT", "Engineer", "Network"],

            ["Reza", 3456, 8000, "IT", "Worker", None],

            ["Mina", 1122, 16000, "HR", "Manager", None],

            ["Pouya", 3344, 11000, "HR", "Engineer", "Recruitment"],

            ["Nima", 5566, 9000, "HR", "Worker", None],

            ["Laila", 7788, 17000, "Finance", "Manager", None],

            ["Arman", 9900, 14000, "Finance", "Engineer", "Auditing"],

            ["Shayan", 2233, 9500, "Finance", "Worker", None],

            ["Farid", 4455, 18000, "Production", "Manager", None],

            ["Niloofer", 6677, 13000, "Production", "Engineer", "Manufacturing"],

            ["Saeed", 8899, 10500, "Production", "Engineer", "Quality Control"],

            ["Kaveh", 1010, 8700, "Production", "Worker", None],

            ["Homa", 2020, 17500, "R&D", "Manager", None],

            ["Mehran", 3030, 13500, "R&D", "Engineer", "AI Research"],

            ["Elham", 4040, 11000, "R&D", "Engineer", "Data Science"],

            ["Babak", 5050, 9700, "R&D", "Worker", None],

            ["Kamran", 6060, 8800, "Logistics", "Worker", None],

            ["Taraneh", 7070, 15500, "Logistics", "Manager", None],

            for item in data:

                name, id, salary, department, role, specialty = item
```



```
if role == "Manager":  
    self.employees.append(Manager(name, id, salary, department))  
elif role == "Engineer":  
    self.employees.append(Engineer(name, id, salary, department, specialty))  
else:  
    self.employees.append(Worker(name, id, salary, department))
```

لازم به ذکر است که Employee برای Manager و Engineer و Worker یک super به حساب می آید. (نسبت فرزند-والد)

```
super().__init__(name, id, salary, department)
```

راهنمایی 2

معرفی pandas و matplotlib

برای کار با داده‌ها در پایتون، دو کتابخانه‌ی بسیار پرکاربرد وجود دارند

pandas : برای مدیریت و تحلیل داده‌ها

Matplotlib : برای ایجاد نمودارهای گرافیکی و نمایش داده‌ها

1. نصب کتابخانه‌ها

اگر این دو کتابخانه روی سیستم نصب نیستند، ابتدا باید آن‌ها را نصب کنید:

```
pip install pandas matplotlib
```

2. معرفی pandas (مدیریت داده‌ها)

pandas به ما اجازه می‌دهد تا داده‌ها را به صورت جدول (**DataFrame**) مدیریت کنیم.




## فراخوانی کتابخانه pandas

```
import pandas as pd
```

ساده ایجاد یک DataFrame

```
data = {  
    "Name": ["Ali", "Sara", "Reza"],  
    "Salary": [15000, 12000, 8000]  
}  
df = pd.DataFrame(data) # تبدیل داده‌ها به یک جدول  
print(df) # نمایش جدول
```

خروجی: 

```
.# Name Salary
```

```
0 Ali 15000
```

```
1 Sara 12000
```

```
2 Reza 8000
```

3 معرفی matplotlib (رسم نمودارها)

matplotlib برای نمایش گرافیکی داده‌ها استفاده می‌شود.

فراخوانی matplotlib

```
import matplotlib.pyplot as plt
```



تنظیم اندازه‌ی نمودار

```
plt.figure(figsize=(6, 4)) # عرض 6 اینچ، ارتفاع 4 اینچ
```

رسم یک نمودار میله‌ای

```
plt.bar(df["Name"], df["Salary"], color="skyblue")
```

تنظیم برچسب‌های محورهای X و Y

```
plt.xlabel("Employee Name") # نام کارکنان
```

```
plt.ylabel("Salary ($)") # حقوق به دلار
```

```
plt.title("Employee Salaries") # عنوان نمودار
```

نمایش نمودار

```
plt.show()
```

✂ توضیح بخش‌های کد نمودار در matplotlib

در کد زیر، نمودار حقوق کارکنان در یک دپارتمان خاص رسم شده است:

تنظیم اندازه نمودار به  $8 \times 5$  اینچ

```
plt.figure(figsize=(8, 5))
```

```
plt.barh(names, salaries, color="skyblue") # رسم نمودار میله‌ای افقی با رنگ آبی روشن
```

```
plt.xlabel("Salary ($)") # نمایش مقدار حقوق روی محور X
```

```
plt.ylabel("Employee Name") # نمایش نام کارکنان روی محور Y
```

```
plt.title(f"Salaries in {department} Department") # عنوان نمودار بر اساس دپارتمان انتخابی
```

```
plt.grid(axis="x", linestyle="--", alpha=0.7) # افزودن خطوط راهنما روی محور X
```

```
plt.show() # نمایش نمودار
```

## 08. جنگ شور و شیرین (کلانتری)

جنگی رخ داده است. تیم حلیم با شکر (شکریا) در یک سمت و تیم حلیم با نمک (نمکیا) در سمت دیگر قرار دارند. آن ها برای جلوگیری از ناجوانمردی یک سری قوانین برای جنگ وضع کردند. آن ها قرار گذاشتند که تن به تن در یک صفحه‌ی ۱۰ در ۱۰ با هم مبارزه کنند. قوانین مسابقه به صورت زیر است:

- مبارز تیم شکریا در نقطه‌ی (0,0) قرار دارد (گوشه چپ بالا) و مبارز تیم نمکیا در نقطه‌ی (9,9) (گوشه راست پایین) ایستاده است.
- مبارزه نوبتی انجام می‌شود که ابتدا تیم شکریا حرکت اول را انجام می‌دهد.
- در هر حرکت اعمال مجاز عبارتند از:
  - حرکت به اندازه یک واحد به چهار جهت r, l, u, d (که r به معنای right و l به معنای left و u به معنای up و d به معنای down است)
  - شلیک به چهار جهت sr, su, sl, sd (که sr به معنای shoot right و sl به معنای shoot left و su به معنای shoot up و sd به معنای shoot down است)
- شلیک‌ها به این صورت هست که تیرها سرعت بی‌نهایت دارند و در لحظه‌ی شلیک به هدف برخورد می‌کنند و همچنین تیرها تا بی‌نهایت در حرکتند به طور مثال اگر مبارز شکریا در (0,7) باشد و مبارز نمکیا در (0,1) باشد، و مبارز شکریا su را اجرا کند، آنگاه چون هردو در یک ستون قرار دارند و مبارز نمکیا بالاتر از مبارز شکریا است پس تیر در همان لحظه به مبارز نمکیا اصابت می‌کند. حال تیم برگزاری جنگ از شما خواسته تا کدی برای داوری مبارزات بنویسید و پس از ورودی گرفتن حرکت‌های هر بازیکن در هر نوبت، نتیجه‌ی مبارزه را خروجی دهید. قوانین پیروزی یک طرف به قسم زیر است:



- اگر به هر مبارز تیر اصابت شود، مبارز کشته شده و طرف مقابل پیروز می‌شود.
- اگر هر مبارز از زمین مسابقه خارج شود، طرف مقابل پیروز می‌شود.
- اگر مبارزها در یک خانه به هم برخورد کنند، آن مبارزی که زودتر در آن خانه حضور داشته پیروز می‌شود.

توجه کنید که باید از کلاس استفاده کنید

### ورودی ها

ورودی شامل یک  $n$  که تعداد کل حرکات را نشان می‌دهد و سپس در  $n$  خط بعد به ترتیب حرکات مبارز شکریا و مبارز نمکیا را نشان می‌دهد. (خطوط فرد حرکات مبارز شکریاست و خطوط زوج حرکات مبارز نمکیا)

$$1 \leq n \leq 100$$

### خروجی ها

در صورت پیروزی مبارز شکریا shekaria won در صورت پیروزی مبارز نمکیا namakia won و در صورت برابری دو مبارز draw نمایش داده شود.

## 09. دوز بی دوز (کلانتری)

شما باید در این سوال بتوانید که بازی دوز را با استفاده از قوانین oop در محیط ترمینال پیاده سازی کنید  
محتوای این سوال شامل 3 کلاس زیر با قابلیت های آن می باشد.

1. کلاس Board : برای مدیریت وضعیت صفحه بازی (خانه‌ها، ابعاد) و ارائه متدهایی برای انجام حرکت، بررسی برد و بررسی تساوی.

2. کلاس Player : برای نمایش بازیکنان با ویژگی‌هایی مانند نماد (X یا O) و احتمالاً استراتژی برای بازیکن کامپیوتر.

3. کلاس Game : برای کنترل جریان کلی بازی، شامل مدیریت نوبت‌ها، دریافت ورودی از کاربر و تعیین نتیجه بازی.

در این بازی، بازیکنان به نوبت نمادهای خود را در خانه‌های خالی قرار می‌دهند. اولین بازیکنی که بتواند سه نماد خود را به صورت افقی، عمودی یا مورب در یک ردیف قرار دهد، برنده است.

اگر تمام خانه‌ها پر شوند و هیچ بازیکنی نتواند برنده شود، بازی مساوی اعلام می‌شود.

برخی از قابلیت‌هایی که باید کد‌های شما داشته باشد:

1. اندازه متغیر صفحه و شرط برد:

- امکان تعیین اندازه صفحه بازی بالاتر از  $3*3$  توسط کاربر در ابتدای بازی.

- امکان تعیین تعداد نمادهای مورد نیاز برای برنده شدن (مثلاً 3، 4، و غیره) توسط کاربر در ابتدای بازی.

2. تاریخچه حرکات و بازپخش:

- قابلیت ذخیره‌سازی تمام حرکات انجام شده در طول بازی.

- امکان بازپخش بازی پس از اتمام، به صورت گام به گام در ترمینال، نمایش وضعیت صفحه پس از هر حرکت.

### 3. دادن قابلیت های هوش مصنوعی به بازیکن کامپیوتر:

- این هوش مصنوعی باید ابتدا بررسی کند که آیا می تواند در حرکت بعدی برنده شود، سپس بررسی کند که آیا حریف می تواند در حرکت بعدی برنده شود و در صورت لزوم حرکت او را مسدود کند، و در نهایت بر اساس یک استراتژی ساده دیگر (مانند اولویت دادن به خانه وسط، سپس گوشه ها) حرکت کند.

از بین موارد بالا باید حداقل 2 مورد را انتخاب کنید. همچنین با استفاده از کتابخانه OS باید بتوانید محیط ترمینال را کنترل کنید. در پایان باید وضعیت بازی نشان داده شود. اگر بازی مساوی نشد باید برنده و بازنده بازی نیز مشخص شود.

این مساله به صورت دستی تصحیح می شود.





نمونه هایی از محیط اجرای بازی در ترمینال:

```
0 | 1 | 2 | 3
---+---+---+---
0   |   |   |
---+---+---+---
1   |   |   |
---+---+---+---
2   |   |   |
---+---+---+---
3   |   |   |
---+---+---+---
```

```
0 | 1 | 2
---+---+---
0 x | o |
---+---+---
1   |   | o
---+---+---
2   | x | o
```