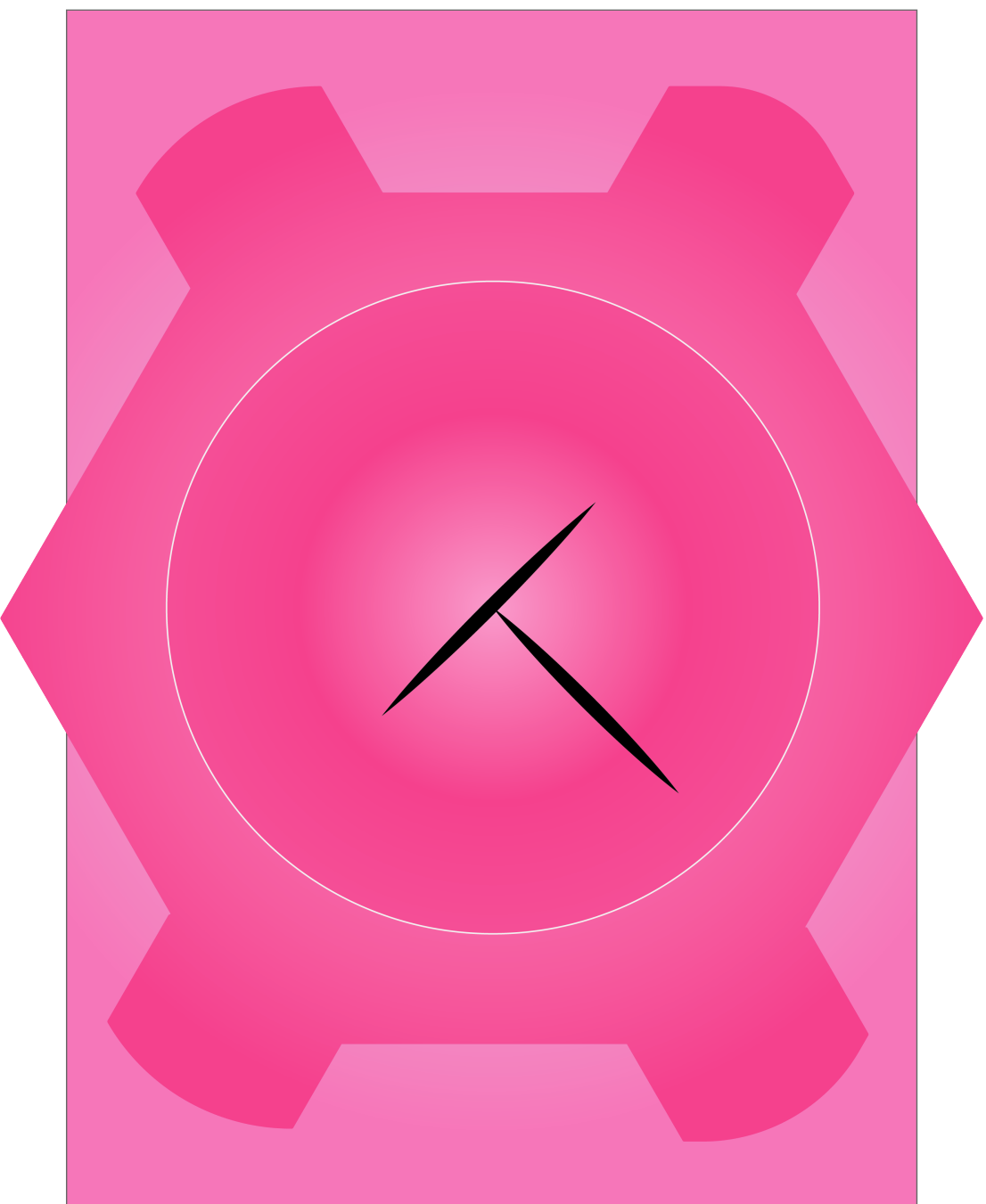


بازی سازی و برنامه نویسی خلاقانه پایه

چوبک بیدپا

ج.



عرضه شده تحت لیسانس MIT

۱۳۹۷

نویسنده: چوبک بیدپا

سال عرضه: ۱۳۹۷

کاملاً رایگان

جهت برقراری ارتباط با چوبک بیدپا از ایمیل chubakbidpaa@riseup.net استفاده نمایید.

به خاطر اینکه بخش قابل توجهی از این کتاب، از آموزشهایی که تحت لیسانس Creative Commons، MIT و GPL عرضه شده اند استفاده میکند، استفاده، تکثیر و آموزش این کتاب، به شرط نام بردن نویسنده یعنی شخص حقیقی چوبک بیدپا، آزاد میباشد.

توجه داشته باشید که فایلهایی که همراه کتاب به فروش گذاشته شده اند، غیرقابل تکثیر بوده، و آپلود آنها توسط شخص حقیقی چوبک بیدپا قابل قبول میباشد.

تحت قوانین Transference، شما جهت استفاده از بخشهای این کتاب که توسط افراد دیگر نگاشته شده اند احتیاجی به اجازه از آنها ندارید.

در آخر، قابل توجه باشد که این کتاب یک پروژه ی اشتیاقی^۱ می باشد، و نه پروژه ای که برای به دست آوردن پول نوشته شده است. برای همین، مرام را به جای آورد و آنرا در جای دیگر آپلود نکنید.

لطفاً فایلهایی که همراه کتاب خریده اید نیز جایی کپی نکنید. قیمت فایلهای با توجه به استطاعت خوانندگان، با الگوریتمی پیچیده^۲ تعیین شده است. برای همین همه میتوانند آن را بخرند. آپلود فایلهای کتاب در جای دیگر، پائرسی حساب میشود و از لحاظ اخلاقی، کاریست نپسندیده.

اما تکثیر خود کتاب با ذکر منبع آزاد است.

نکته: کدهای کتاب در فایل خریداری شده حی و حاضر آماده ی کپی میباشد.

^۱ Passion Project

^۲ با بیشتر شدن تعداد خریدارها، قیمت کاهش میابد.

قراردادهای کتاب

۱. بخشهای کتاب: این کتاب به دو بخش برنامه نویسی خلاقانه^۳، و بازی سازی تقسیم شده است.



۲. نکته: نکته های خاص کتاب در به این صورت مشخص شده اند:

نکته اینجاست که....

۳. استفاده از فایلها: اگر فایلی لازم باشد، نام و آدرس آن در فایل زیپ دانلود شده نوشته خواهد شد.

۴. یو آر الها به صورت `https://google.com` نوشته خواهند شد.

۵. متن پررنگ: وقتی لازم است روی کلمه ای تاکید کنم، یا کلمه جدید است و قبلا استفاده نشده، از **متن پررنگ** استفاده خواهم کرد.

۶. دکمه ها به صورت آیکون تعریف شده اند. یادتان باشد که دکمه ی  نشانگر دکمه ی شیفِت میباشد. دکمه ی بالا  میباشد.

۷. کد: کدهایی لازمه به صورت زیر نوشته خواهند شد.

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive');
```

Creative Programming^۳

Write('Pascal keywords.');

فهرست مطالب

ج	فهرست مطالب
۱	۱ چند کلمه با خواننده
۳	۲ نگاهی کوتاه به ریاضی لازمه
۳	۱.۲ توابع
۴	۲.۲ بردارها
۸	۳.۲ مثلثات
۱۱	۴.۲ ماتریسها
۱۲	۵.۲ قائمیت در فضای سه بعدی
۱۵	۶.۲ پایان فصل ریاضی
۱۷	۳ نگاهی کوتاه به برنامه نویسی
۱۷	۱.۳ برنامه نویسی فانکشنال
۱۹	۲.۳ برنامه نویسی شیء گرا
۲۰	۳.۳ کتابخانه ها
۲۰	۴.۳ پایتان

فصل ۱

چند کلمه با خواننده

«آزادی خود را گرامی بدارید، وگرنه آنرا از دست میدهید.» امروزه، جبهه های مختلفی هستند که بر آزادی اطلاعات عقیده دارند. یکی از آنها نهاد گنو^۱ است که کرنل سیستم عامل لینکس^۲ را در دست دارد. دیگری مازیلاست^۳، که مرورگر فایرفاکس^۴ را منتشر کرده است.

من به شخصه معتقدم آزادی اطلاعات از آزادی بیان مهمتر است، چون اگر اطلاعات را برای خود نگه داریم، کمتر کسی راههای اشاعه ی آزادی بیان را یاد خواهد گرفت، یا اصلا خواهد دانست که آزادی بیان چه هست.

این کتاب نه تنها بر پایه ی عقیده به آزادی اطلاعات^۵ مجانی است، بلکه یکی از دلایل مجانی بودن آن اینست که تمام آن مال من نیست، بلکه، حدود ۳۰٪ این کتاب، ترجمه ی آموزشهای اینترنت، با اجازه از صاحبان آنهاست. ۲۰٪ این کتاب، از داکيومنتشنهای رسمی برداشته شده و ۵۰ درصد باقی را خودم نوشته ام.

شاید برایتان سوال باشد چرا این کتاب را نگاشت کرده ام. دلیل اصلی آن اینست که دلیلی داشته باشم تا برنامه نویسی را ادامه دهم. بعضی ها پروژه مینویسند، بعضی ها کتاب مینویسند. من در لفافه ی کتاب، پروژه مینوسم. تمام پروژه های کتاب اریجینال بوده، و فایلهایی که همراه کتاب خریده اید، کار من هستند.

دلیل دیگری که این کتاب را نوشته ام، اینست که کتاب های بازی سازی به زبان فارسی کم هستند، و کمتر کسی در ایران از برنامه نویسی خلاقانه خبر دارد. سعی من درینست که با نوشتن در مورد این دو دیسپلین دوست داشتنی، فرهنگ آنها را در کشور اشاعه بدهم.

GNU^۱

Linux^۲

Mozilla^۳

Firefox^۴

Freedom of Information^۵

از سابقه ام در برنامه نویسی و بازی سازی بگویم. من از شانزده سالگی 25 کم و بیش در برنامه نویسی، و گهگاهی ساخت بازی، فعال بوده ام. مانند خیلی ها از نرم افزار Game Maker کارم را شروع کردم! و با آن چندین بازی مانند تتریس، بریک اوت و... ساختم. من چندین بازی تحت اسکی مانند بلک جک نیز نوشته ام. من زبانهای سی پلاس پلاس، پایتان، و سی را میدانم و با زبان اسکریپت نویسی چندین نرم افزار آشنایی دارم. درضمن نمره ی تافل در ۱۷ سالگی ۹۵ بوده پس به ترجمه ام اعتماد کنید.

سابقه ی من در برنامه نویسی خلاقانه کمتر است. دو سال پیش بود که با نرم افزار افتر افکتس^۶ آشنا شدم و به صرافت نوشتن پلاگین برایش افتادم، و طی این امر، با کتابخانه ی Cinder برای سی پلاس پلاس آشنا شدم. و از آنجا بود که با زبان Processing و شیدر ها آشنا گردیدم. الان تسلط کافی برای آموزش پایه ی شیدرها و زبانها و کتابخانه های برنامه نویسی خلاقانه دارم.

بگذارید در مورد چارچوب کتاب کمی صحبت کنم. در این کتاب، دو بخش داریم، برنامه نویسی خلاقانه، و بازی سازی که به دو بخش Asset و برنامه نویسی تقسیم میشود. در بخش آست سعی شده با استفاده از برنامه های مختلف، ساخت اسپریت، تایل، اسپریت شیت، تایل شیت، عکس پس زمینه، مدل سازی سه بعدی، و تکسچر و متریال را آموزش دهم. در بخش برنامه نویسی کتابخانه ی Arcade پایتان، کتابخانه ی SFML سی پلاس پلاس، و انجین Godot آموزش داده خواهد شد. اگر فرصت شد، آموزش کوتاهی برای ساخت انجین خودتان را خواهم نوشت.

قبل از هرچیزی دو چیز باید یادآوری شود: برنامه نویسی، و ریاضی. من زیاد در مورد این دو کانسپت حرف نمیزنم، چون وظیفه ی خود خواننده است که این دو را از قبل یاد داشته باشد، اما فقط در حد یادآوری، در مورد این دو حرف خواهم زد.

در آخر، در این دنیای پرهیر و گیر، اگر پشینی دارید که به شما آرامش میدهد، نیکوست. و اگر این کتاب برای پیدا کردن این پشین کمک میکند، خوشحالم.

چوبک بیدپا مشهد - ۱۳۹۷

فصل ۲

نگاهی کوتاه به ریاضی لازمه

۱.۲ توابع

یک تابع^۱ به صورت زیر نشان داده میشود:

$$y = f(x)$$

وظیفه ی یک تابع، تغییر عدد داده شده بر اساس قوانین داده شده است. به این قانون، تابع میگویند. مثلاً تابع $f(x) = x^2$ که به آن تابع مربع می گویند، وظیفه اش بردن عدد به توان دو است. به عکس تابع، تابع معکوس^۲ میگویند و به صورت زیر نشان داده میشود:

$$y = f(x)^{-1(y)}$$

مثلاً معکوس تابع مربع، تابع ریشه دو یعنی $f(x) = \sqrt{x}$ میباشد. میتوان دو تابع را با هم به صورت $f(g(x))$ ترکیب کرد که به آن تابع مرکب میگویند. از دیگر عملیاتها عبارت است از:

$$(f + g)(x) = f(x) + g(x)$$

$$(f - g)(x) = f(x) - g(x)$$

$$(f * g)(x) = f(x) * g(x)$$

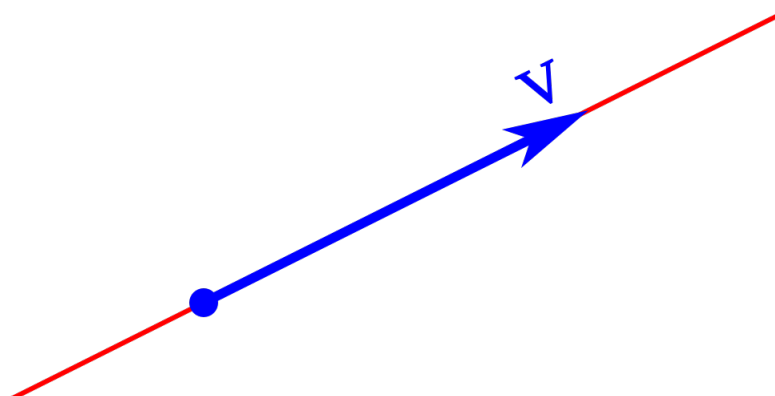
Function^۱
Inverse^۲

$$\left(\frac{f}{g}\right)(x) = \frac{f(x)}{g(x)}$$

به تمام اعدادی که تابع میپذیرد، دامنه^۳، و تمام اعدادی که تابع خارج میکند، برد^۴ خوانده میشود. دامنه ی یک تابع را ما تعیین میکنیم، اما برد آن را خود تابع تعیین میکند. در آخر، بگذارید بگوییم که تابع مانند یک ماشین است. ورودی آن x و خروجی آن $f(x)$ است. در برنامه نویسی از توابع استفاده ی زیادی میشود. در بخش برنامه نویسی خواهید خواند.

۲.۲ بردارها

اگر فضای دوبعدی را به دو بخش نقاط افقی و نقاط عمودی تقسیم کنیم، بردار^۵ خطی است که چهار نقطه را به هم وصل میکند.



بردار را به صورت زیر نشان میدهند:

$$\vec{V} = (x_1, y_1) + (x_2, y_2)$$

صفحه ی مختصات را به این صورت میکشیم:

Domain^۳
Range^۴
Vector^۵



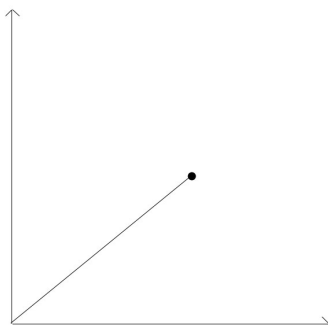
که به آن **دستگاه مختصات دکارتی**^۶ میگویند. در دستگاه مختصات دکارتی، دو محور X و Y به ترتیب محور افقی و عمودی ما را تشکیل می دهند. ما مختصات یک نقطه را در پراکنش به صورت (X, Y) نشان میدهیم. همانطور که گفته شد، خطی که دو نقطه را به هم وصل کند، بردار نام دارد.

در جبر خطی، بردار به صورت:

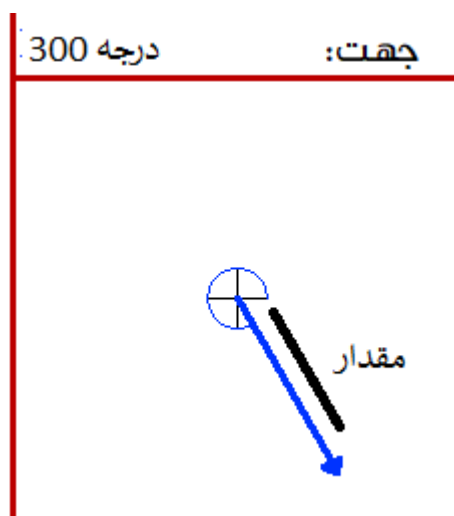
$$\vec{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$$

نشان داده میشود و نقطه ی اول آن، مسقط الرأس دستگاه مختصات یعنی $(0, 0)$ میباشد.

^۶ Cartesian Coordinate System



به برداری که مختصات افقی، یا عمودی آن، یک باشد بردار واحد میگویند. بردار واحد افقی را \vec{i} و بردار افقی عمودی را \vec{j} میگویند. بردارها را میتوان به صورت مضربی از بردار واحد نشان داد مثلاً بردار $\vec{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$ را میتوان به صورت $X\vec{i} + Y\vec{j}$ نشان داد. یک بردار دارای دو خصیصه می باشد. جهت^۷ و مقدار^۸. که به صورت زیر نشان داده میشود:

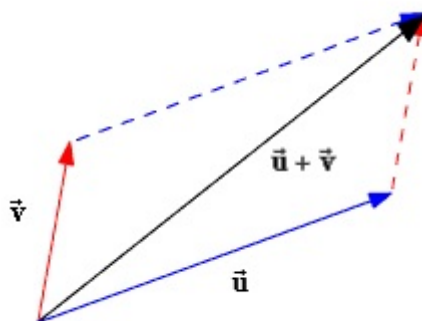


برای به دست آوردن مقدار بردار ازین فرمول استفاده میکنیم:

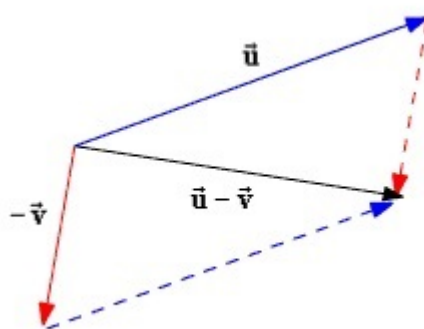
$$|\vec{V}| = \sqrt{x^2 + y^2}$$

دو بردار را میتوان به صورت زیر جمع کرد:

Direction^۷
Magnitude^۸



و به این صورت تفریق کرد:



اما دو نوع ضرب برداری داریم. ضرب نقطه ای^۹ و ضرب صلیبی^{۱۰}. قبل ازین که پیش بروید، قسمت مثلثات 2.3 را بخوانید. فرض کنید دو بردار به صورت زیر هستند:



ضرب نقطه ای به صورت زیر تعریف میشود:

$$\vec{A} \cdot \vec{B} = |\vec{A}| |\vec{B}| \cos \alpha$$

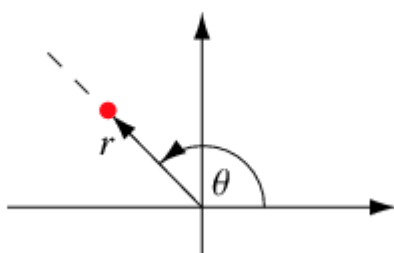
و ضرب صلیبی ازین فرمول استفاده میکنیم.

$$\vec{A} \times \vec{B} = |\vec{A}| |\vec{B}| \sin \alpha \vec{n}$$

که \vec{n} برداری واحد عمود بر دو بردار است. برای به دست آوردن \vec{n} کافیست از انگشتان وسط، اشاره، و شصت خود استفاده کنید. انگشت شصت شما، همواره بردار واحد عمود است، که مضربی از ضرب صلیبی دو بردار می باشد.

۳.۲ مثلثات

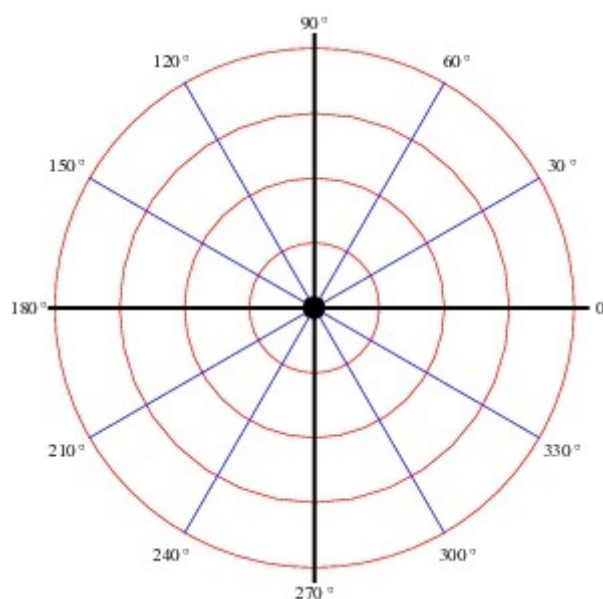
مثلثات بحثیست پیچیده. و من نیز ریاضیدان نیستم پس به کمی در مورد این مبحث قناعت میکنیم. قبل از هرچیزی، بگذارید در مورد **دستگاه مختصات قطبی**^{۱۱} حرف بزنم. دستگاه مختصات قطبی، مانند دستگاه مختصات دکارتی، دارای دو محور عمودی و افقی است. اما در این دستگاه مختصات، ما یک نقطه را، عوض X و Y توسط یک زاویه α و یک بردار شعاع \vec{r} نشان میدهیم:



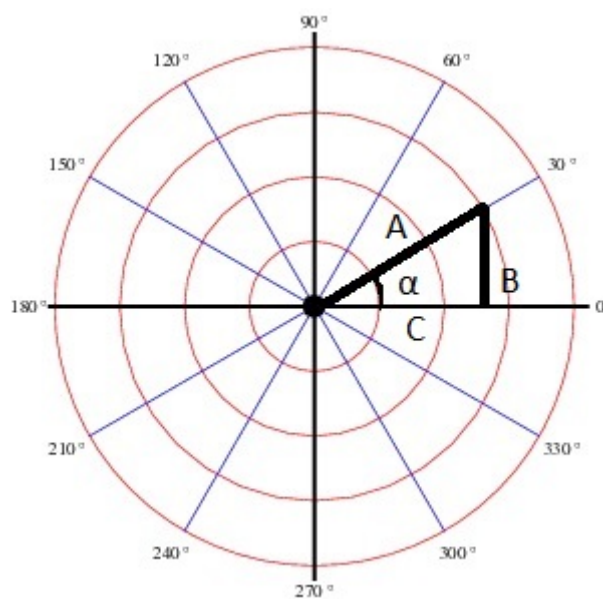
در برنامه نویسی خلاقانه، دستگاه مختصات قطبی کاربردهای زیادی دارد. اما در کامپیوتر، **پیکسلها**^{۱۲} در دستگاه مختصات دکارتی قرار دارند. حلال مشکلات ما، مثلثات است. یک دایره ی واحد را در دستگاه مختصات قطبی کنید که شعاعش ۱ میباشد:

^{۱۱} Polar Coordinate System

^{۱۲} در مورد پیکسلها به وفور حرف خواهیم زد.



اگر زاویه ی 30° را انتخاب کرده و یک مثلث قائم الزاویه دور آن بکشیم:



سینوس و کسینوس زاویه 30° درجه که اینجا α نامیده میشود، به صورت زیر تعریف میگردد:

$$\sin \alpha = \frac{\text{مقابل}}{\text{وتر}}$$

و:

$$\cos \alpha = \frac{\text{مجاور}}{\text{وتر}}$$

همچنین تانژانت و کتانژانت به صورت زیر تعریف میشوند:

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha}$$

و

$$\cot \alpha = \frac{\cos \alpha}{\sin \alpha}$$

درجه، تنها واحد اندازه گیری زاویه نیست. واحد دیگر، **رادیان**^{۱۳} نام دارد. یک زاویه در رادیان، بین ۰ و 2π قرار دارد. ارزش π حدود ۳.۱۴۱۵۹۲۶۵۳۵ است. ما برای کار با پیکسلها، ارقام اعشار بیشتر ازین نیز نیازمندیم. برای تبدیل درجه به رادیان:

$$n^{\circ} \times \frac{\pi}{180}$$

، و بالعکس:

$$n\text{rad} \times \frac{180}{\pi}$$

توابع مثلثاتی توسط **هویتهای مثلثاتی**^{۱۴} به هم ربط داده میشوند. بعضی ازین هویتها عبارتند از:

$$\sin^2 \alpha + \cos^2 \alpha = 1$$

$$\sin(-\alpha) = -\sin \alpha$$

$$\cos(-\alpha) = \cos(\alpha)$$

$$\sin(\alpha \pm \beta) = \sin \alpha \cos \beta \pm \cos \alpha \sin \beta$$

$$\cos(\alpha \pm \beta) = \cos \alpha \cos \beta \pm \sin \alpha \sin \beta$$

اینها تقریباً تمام سرفصلهایی هستند که شما برای این کتاب لازم دارید. توجه کنید، این کتاب، نه برنامه نویسی خلاقانه و بازی سازی کلی.

^{۱۳} Radians

^{۱۴} Trigonometric Identities

۴.۲ ماتریسها

به آرایه هایی از اعداد که به صورت n سطر و m ستون به نمایش در می آیند، **ماتریس**^{۱۵} میگویند. یک ماتریس را به این صورت نشان میدهند:

$$M_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

در ساخت بازیهای کامپیوتری و برنامه نویسی خلاقانه ما بیشتر نیاز به ماتریسهای 2×2 ، 3×3 و 4×4 داریم. جمع و تفریق ماتریسها به صورت همسان انجام میشود:

$$M_{m,n} \pm N_{m,n} = \begin{bmatrix} m_{1,1} \pm n_{1,1} & m_{1,2} \pm n_{1,2} & \cdots & m_{1,n} \pm n_{1,n} \\ m_{2,1} \pm n_{2,1} & m_{2,2} \pm n_{2,2} & \cdots & m_{2,n} \pm n_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} \pm n_{n,1} & m_{n,2} \pm n_{n,2} & \cdots & m_{n,n} \pm n_{n,n} \end{bmatrix}$$

ضرب ماتریسها به این روش صورت میپذیرد که، هر سطر با یک ستون. پس تا سطرها و ستونهای دو ماتریس با هم مساوی نباشند، ضرب صورت نمیپذیرد.

$$M_{1,n} \times N_{m,1} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,m} \end{bmatrix} \times \begin{bmatrix} n_{1,1} \\ n_{2,1} \\ \vdots \\ n_{m,1} \end{bmatrix} = (m_{1,1} \times n_{1,1}) + (m_{1,2} \times n_{2,1}) + \cdots + (m_{1,n} \times n_{m,1})$$

یکی دیگر از عملیتهای ماتریسی، **دترمینان**^{۱۶} است. برای احتساب دترمینان ماتریسهای بزرگتر از 3×3 الگوریتمهای زیادی مانند **دیکامپوزیشن**^{۱۷} وجود دارد که خود آن توسط افراد مختلفی در طول سالها بهسازی گشته اند، اما راه ساده ای برای به دست آوردن دترمینان 2×2 وجود دارد که به شرح زیر است:

Matrix^{۱۵}
Determinant^{۱۶}
Decomposition^{۱۷}

$$A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$|A| = AD - BC$$

به I_n ماتریس هویت^{۱۸} میگویند و مثلاً I_3 به صورت زیر تعریف میشود:

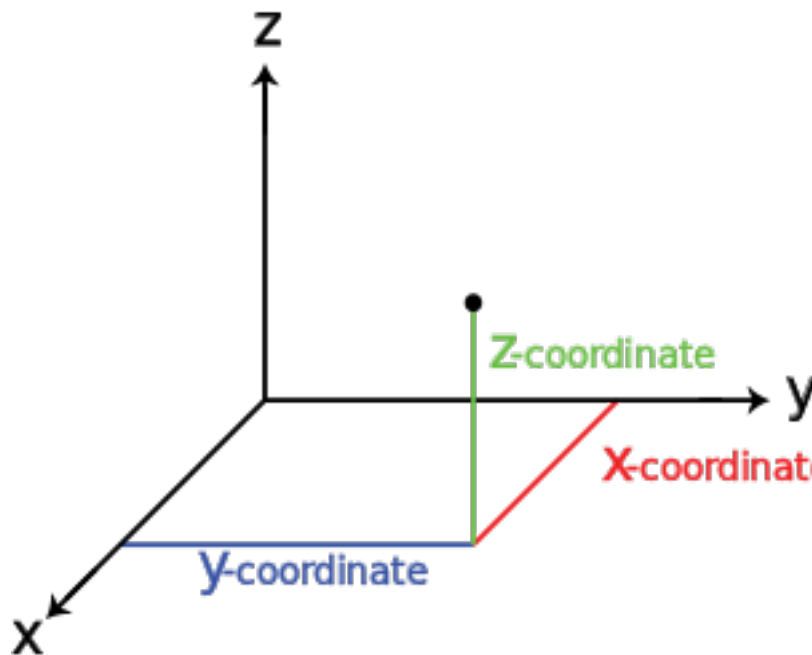
$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ما در برنامه نویسی خلاقانه و بازی سازی از ماتریسها استفاده های زیادی خواهیم برد.

۵.۲ قائمیت در فضای سه بعدی

ما در بخش بردار دیدیم که دستگاه مختصات دکارتی شامل دو محور عمودی و افقی است. اما همیشه اینگونه نیست، بلکه، میتوان با اضافه کردن یک بردار اضافه که نام آن Z است به دستگاه سه بعدی دست پیدا کنیم. این دستگاه را به صورت \mathbb{R}^3 نشان میدهند و در تصویر زیر میتوانید محور Z را مشاهده کنید:

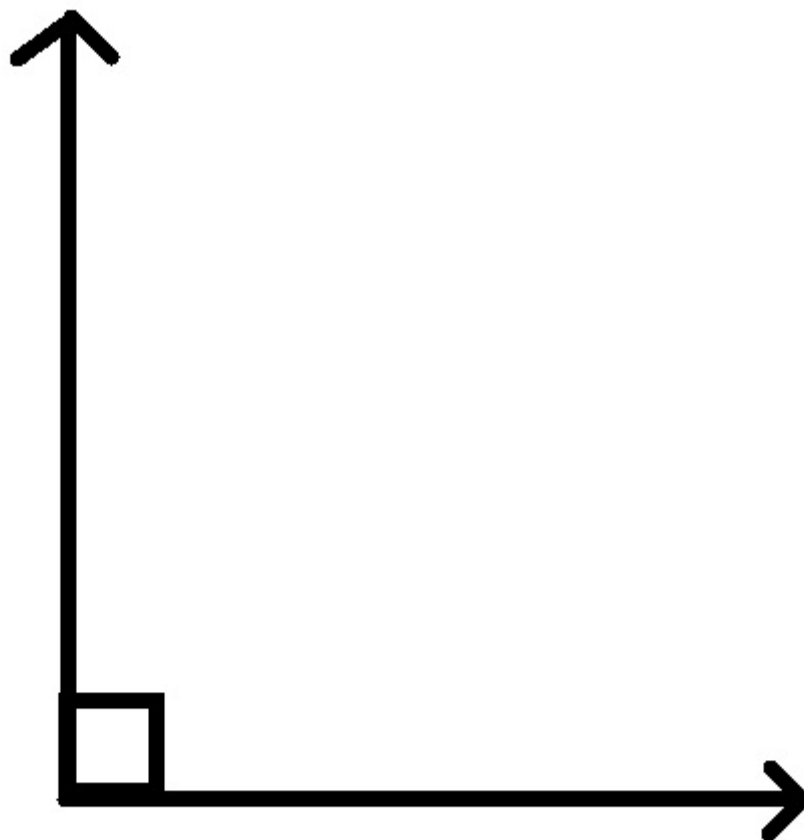
^{۱۸}Identity



توجه کنید که در بعضی از نرم افزارها جای Y با Z عوض میشود. یک بردار را در فضای R^3 به صورت زیر نشان میدهیم:

$$\vec{V} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

همه ی قوانین R^2 برای R^3 برقرار است. مثلاً به بردار واحد محور Z ، \vec{k} میگویند. غرض از این بخش، اینست که **قائمیت**^{۱۹} در فضای سه بعدی را معرفی کنم. زیرا برای بازیهای دو و نیم بعدی، دوربین باید قائم بر فضای R^3 باشد.



در کل، دو بردار وقتی بر هم قائمند که حاصلضرب نقطه ای آنها صفر باشد:

$$\vec{A} \cdot \vec{B} = 0 \text{ اگر و تنها اگر } \vec{B} \text{ قائم است بر } \vec{A}$$

اگر فرمول ضرب نقطه ای یادتان باشد، و در اینترنت کسینوس ۹۰ را خوانده باشید، میدانید که:

$$\vec{A} \cdot \vec{B} = |A||B| \cos \alpha$$

و $\cos(90) = 0$ پس:

$$|A||B| \cos(90) = 0$$

۶.۲ پایان فصل ریاضی

کلید یادگرفتن ریاضی یک چیز است: n تمرین! یادتان نرود که حافظه، چیزست فرار، و هر لحظه ممکن است همین چیزهای کمی که از بنده ی حفر آموخته اید، که مطمئنم برای بیشتر شما یک یادآوری ساده و کوتاه بوده، و برای خیل عظیمی از شما فوت آب بوده، و فقط یک لیست است، سریع از حافظه ی شما رخت برمیبندند. تمرین کنید، نوت برداری کنید، و یادتان نرود که روم را در یک روز نساخته اند. این ضرب المثل را چندین بار در طول کتاب تکرار خواهم کرد. یادگیری طول میکشد. و یادتان نرود که هیچکس استعداد چیزی را ندارد، و همه چیز با تمرین میسر میشود.

در فصل بعد، در مورد برنامه نویسی، زبان پایتان و C++ حرف خواهیم زد.

فصل ۳

نگاهی کوتاه به برنامه نویسی

درین فصل نگاهی کوتاه می اندازیم به برنامه نویسی^۱. ابتدا به دو پردایم^۲ برنامه نویسی فانکشنال^۳ و شیء گرا^۴. و بعد پیچیدگی زمانی^۵ را توضیح خواهیم داد. بعد از آن، نگاهی می اندازیم به سینتکس^۶ زبان پایتان^۷ و C++.

۱.۳ برنامه نویسی فانکشنال

از بین تمام روشها، یا به عبارتی، پردایمهای برنامه نویسی، برنامه نویسی فانکشنال یا تابعی ساده ترین، و پر مصرف ترین آنهاست. اکثر اشخاصی که برنامه نویسی را شروع میکنند، از برنامه نویسی فانکشنال شروع میکنند. زبانهای قدیمی مانند فورترن^۸ و لیسپ^۹ همه فانکشنال هستند. با توابع در فصل ریاضی آشنا شدیم. توابع کامپیوتری نیز با توابع ریاضی فرق زیادی ندارد، همه ی آنها یک ماشین هستند که ورودی را به خروجی تبدیل میکنند. یک تابع، مجموعه ای از دستورات^{۱۰} است که پارامتر^{۱۱} داده شده را با تغییرات، باز میگردانند. این تغییرات میتواند

^۱ Programming

^۲ Paradigm

^۳ Functional

^۴ Object Oriented Programming

^۵ Time Complexity

^۶ به دستور زبانی یک زبان برنامه نویسی Syntax گفته میشود.

^۷ Python

^۸ FORTRAN

^۹ Lisp

^{۱۰} Instructions

^{۱۱} Parameter

عملیاتهای جمع و تفری، ضرب و تقسیم، باقیمانده، و یا تغییر نوع پارامتر مثلاً از عدد صحیح به عدد حقیقی، و یا هرچیز دیگری باشد. برای اجرای تابع، آنرا **میخوانیم**^{۱۲} و به پارامتری که به آن میدهیم، **آرگومان**^{۱۳} میگوییم. اینستراکشن ست زیر را در نظر بگیرید:

۱. عدد n را بگیر.

۲. برای n بار، n را ضربدر $n - 1$ کن.

۳. جواب را برگردان.

به این تابع، تابع **فاکتوریل**^{۱۴} میگویند. توابع زیادی هستند، پیچیده و ساده، مهم اینجاست که از آنها درست استفاده کنید. بعضی از توابع، پارامتر قبول نمیکنند. بعضی از توابع، ارزشی را باز نمی گردانند. به این نوع از توابع **ووید**^{۱۵} میگویند. بعضی از زبانها، **تایپ ثابت**^{۱۶} هستند و باید نوع ارزشهای باز گرداننده را مشخص کرد. C++ یکی ازین نوع زبانهاست. بعضی از زبانها **تایپ دینامیک**^{۱۷} هستند و لازم نیست نوع ارزش بازگرداننده را در آنها مشخص کرد. پایتان یکی ازین زبانهاست. هردو زبان پایتان و C++ هم فانکشنال هستند، هم شیء گرا. در مورد پردایم شیء گرا در بخش بعد صحبت خواهیم کرد. هرزبان مقداری تابع از پیش تعیین شده دارد، اما بقیه ی تابع ها را خودتان باید تعیین کنید. اینکه در چه زمانی باید تابع تعیین کرد، قانون طلایی اینست که هرگاه دیدید عملی را دارید بیشتر از یک بار انجام میدهید، وقت تعیین کردن یک تابع است. همه ی زبانها دارای **کتابخانه**^{۱۸} هایی هستند که شامل توابع و کلاسها 3.2 و دیگر کدهایی هستند که به برنامه نویس کمک میکنند خود را تکرار نکنند.^{۱۹}

Call^{۱۲}

Argument^{۱۳}

Factorial^{۱۴}

Void^{۱۵}

Statically Typed^{۱۶}

Dynamically Typed^{۱۷}

Library^{۱۸}

DRY - Don't Repeat Yourself^{۱۹}

قانون پلاتینیوم برنامه نویسی، اینست. هرگز چیزی که در یک کتابخانه موجود است را ننویسید. مثلاً عوض اینکه در زبان جاوا^{۲۰} عوض نوشتن صدها خط کد برای به دست آوردن یک تابع ماتریس، میتوان از کتابخانه ی JAMA استفاده کرد.

۲.۳ برنامه نویسی شیء گرا

برنامه نویسی شیء گرا بر پایه ی کانسپت کلاس^{۲۱} میجرخد. گفتیم توابع مانند ماشینهایی هستند که اطلاعات را از حالتی به حالت دیگر تغییر میدهند. اگر تابع، ماشین است، کلاس، یک خیابان پر از ماشین است که در آن هزاران ماشین وجود دارد، و همچنین چندده هزار عابر پیاده که سوار ماشین میشوند. درین تشبیه، به ماشین اسلوب^{۲۲} و به عابر پیاده خواص^{۲۳} میگویند. اگر ایده ی کلاس، یعنی یک خیابان پر از عابر پیاده و ماشین را داشته باشیم، با آن میتوانیم هزاران هزار خیابان بسازیم. به هر خیابانی که ما میسازیم، شیء^{۲۴} میگویند. در یک کتابخانه مانند کتابخانه ی JAMA که از آن نام بردیم، یک کلاس به نام ماتریس وجود دارد و این کلاس چندین اسلوب و چندین خواص دارد. یکی از آن اسلوبها، ارزش سطر و ستون داده شده را به کاربر برمیگرداند. بعضی از متدها و خواصها، خصوصی اند، یعنی جز سازنده ی کلاس خیابان، کسی اجازه ی عوض کردن آن را ندارد. اما بعضی از اسلوبها و متدها قابل تغییرند. فرض کنیم یک کلاس داریم به نام مدرسه. برای ساختن یک شیء مدرسه از آن، باید به آن ارزشهایی مانند آدرس مدرسه، تعداد کلاسها، اینکه مدرسه بوفه داشته باشد یا نه، نام مدرسه، اینکه دبیرستان است یا ابتدایی، و... را بدهیم تا از آنها، خواص مدرسه را تعیین کند. این کار توسط اسلوب سازنده^{۲۵} انجام میشود. و یا به سادگی میخواهیم یک مدرسه ی قدیمی را بگوییم و خراب کنیم. برای این کار از اسلوبی به نام خراب کننده^{۲۶} استفاده میکنیم. یگ گلاس میتواند چنین سازنده داشته باشد، ولی فقط یک خراب کننده میتواند داشته باشد.

در بخش بعدی در مورد کتابخانه ها صحبت خواهیم کرد.

-
- Java^{۲۰}
 - Classes^{۲۱}
 - Method^{۲۲}
 - Properties^{۲۳}
 - Object^{۲۴}
 - Constructor^{۲۵}
 - Destructor^{۲۶}

۳.۳ کتابخانه ها

به مجموعه توابع و کلاسهای از قبل آماده شده، کتابخانه میگویند. هر کدی که در صورت اجرا، عملیات خاصی انجام نداده، و به کدهای دیگر برای اجرا وابسته باشد، کتابخانه نام میگیرد. اکثر زبانها برای کتابخانه های خود دارای یک دیتابیس هستند، که زبان پایتان جزء آنهاست، اما بعضی از زبانها برای کتابخانه های خود دیتابیس ندارد، مانند C++. دلیل آن اینست که اکثر کتابخانه های C++، متن بسته و پولی هستند. کتابخانه ها میتوانند به صورت فایل متنی، یا فایل **باینری**^{۲۷} عرضه شوند. کتابخانه های پایتان متنی، و کتابخانه های C++ باینری هستند. گاهی کتابخانه هایی به صورت متنی نیز عرضه میشوند. در این کتاب آموزش ساخت یک کتابخانه ی برنامه نویسی خلاقانه با پایتان را خواهیم داد.

۴.۳ پایتان

زبان پایتان در سال ۱۹۹۹ برای بار اول عرضه شد و در هنگام نوشتن این کتاب، در ورژن ۱.۷.۳ به سر میرد. درین فصل، فقط قطره ای از دریای این زبان را آموزش میدهم. برای آموزش بهتر زبان، به کتاب مخصوص مراجعه کنید.

پایتان زبانی کاملاً مدرن، قابل انعطاف، یکدست، و جذاب و ساده میباشد که برای از اتوماسیون گرفته تا بازی سازی، کاربرد دارد. اینستراکشن های پایتان **ترجمه**^{۲۸} میشوند، یعنی لازم نیست که از قبل به زبان اسمبلی یا ماشین در بیایند، بلکه، در حین اجرا به زبانهایی مثل C یا Java ترجمه میشوند و بعد خط به خط اجرا میشوند. عرضه ی اصلی پایتان که ما از آن استفاده میکنیم، از C++ استفاده میکند.

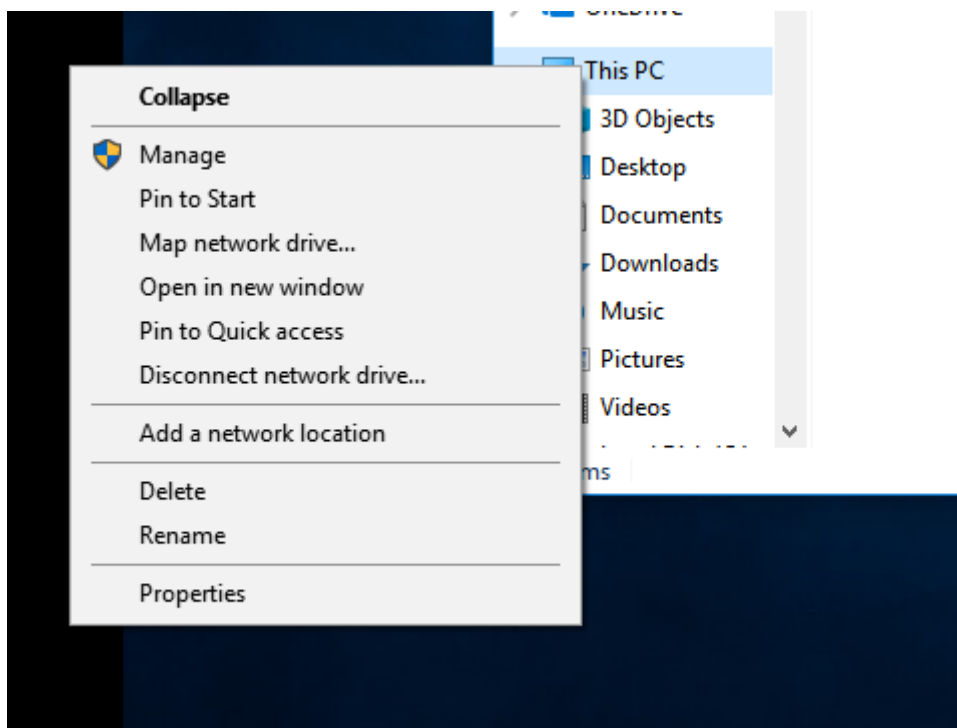
نصب پایتان

۱. به این صفحه بروید و پایتان را دانلود کنید: <https://www.python.org/downloads/>

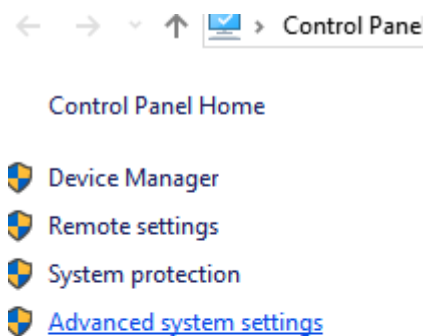
۲. آنرا نصب کنید.

۳. از فایل اکسپلورر مانند زیر روی Properties کلیک کنید:

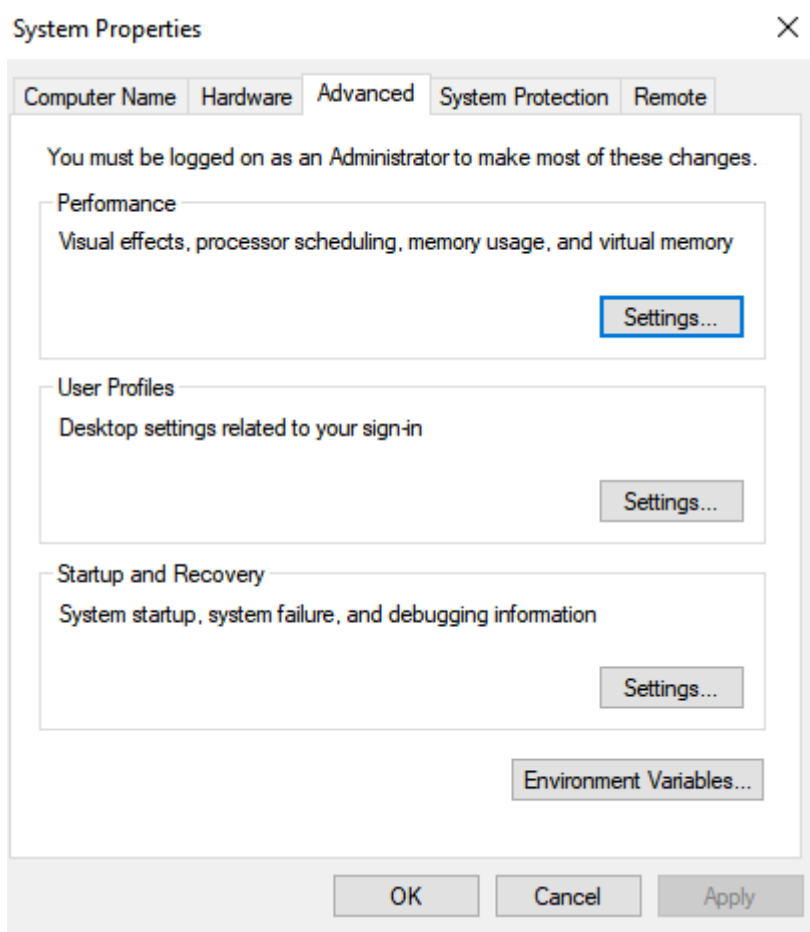
^{۲۷} Binary
^{۲۸} Interpret



۴. روی Advanced System Settings کلیک کنید.



۵. روی گزینه Environment Variables کلیک کنید.



۶. روی Path کلیک کنید.

Variable	Value
OPENCV_DIR	C:\opencv\build\x64\vc15
OS	Windows_NT
Path	I:\AE SDK\Cuda\bin;I:\AE SDK\Cuda\libnvvp;C:\Program Files (x86)\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6

New... Edit... Delete

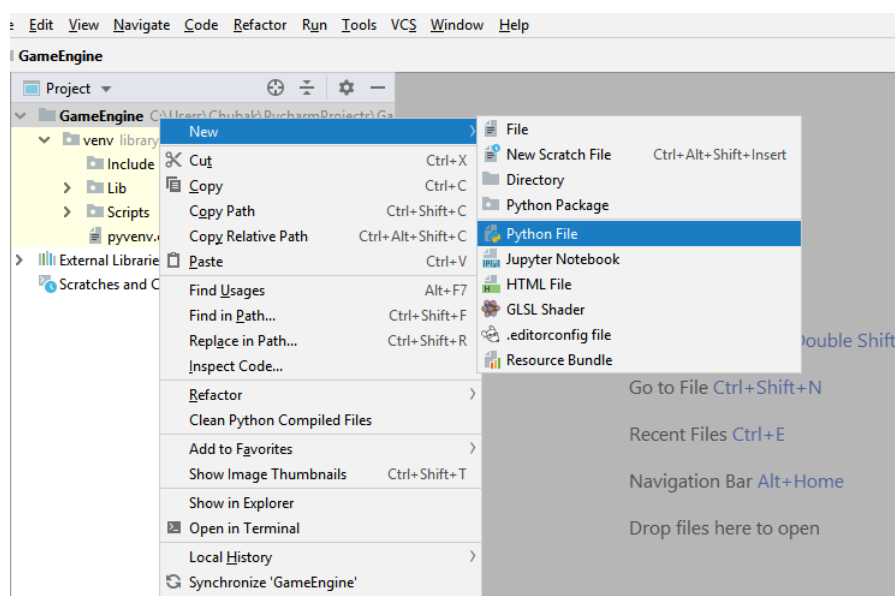
۷. دو گزینه ی زیر را به آن اضافه کنید.

Py	C:\Windows\system32
TE	C:\Windows
TM	C:\Windows\System32\Wbem
	C:\Windows\System32\WindowsPowerShell\v1.0\
	C:\Windows\System32\OpenSSH\
	C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
	C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
yste	D:\Program Files\MiKTeX 2.9\miktex\bin\x64\
Va	C:\Users\Chubak\AppData\Local\Programs\Python\Python37
Ol	C:\Users\Chubak\AppData\Local\Programs\Python\Python37\Scripts
Os	C:\opencv\build\x64\vc15\bin
Pa	C:\Program Files\dotnet\
PA	D:\SFML\thor-v2.0-msvc2015\bin
PF	C:\Program Files\CMake\bin
PF	D:\SFML\SFML\lib
PF	D:\SFML\Thor\lib
PF	D:\GLFW\lib

برای نوشتن کد پایتان احتیاج به یک محیط گسترش مجتمع^{۲۹} دارید. من PyCharm Community را پیشنهاد میکنم که کاملاً مجانیست. آنرا میتوانید از <https://www.jetbrains.com/pycharm/> دریافت کنید.

^{۲۹}Environment Development Integrated

چهارم و باز کردن یک پروژه ی جدید، از طریق زیر یک فایل پایتان بسازید: [com/pycharm/download/#section=windows](https://www.jetbrains.com/pycharm/download/#section=windows) دانلود کنید. بعد از باز کردن پای



حال وقت نوشتن اولین کد ماست. بنویسید:

```
new_variable = "A Portal to the World of Python"
print(new_variable)
```

و **ctrl** + **↑** + **F10** را بزنید. در پایین صفحه، کد شما اجرا میشود.

میتوانید از محیطهای گسترش دیگر نیز استفاده کنید. مانند Beans IDE، Spy - PyDev، Cloud9 IDE و درضمن، Microsoft Visual Studio نیز یک پکیج پایتان دارد.

متغیرها

در بخش قبل، `new_variable` یک **متغیر**^{۳۰} است. متغیرها مانند ظروفی هستند که میتوان در آنها هر چیزی ریخت. چون پایتان زبان قایپ امن^{۳۱} نیست، میتوان هر نوع دیتایی را داخل یک متغیر جا داد.

^{۳۰} Variable
^{۳۱} Type Safe

```
var = True #boolean
var = "String" #string
var = 1 #integer
var = 1.0 #float
var = 1.00 #double
var = Class() #Class
```

همانطور که میبینید، متغیرهای پایتان هر نوع دیتایی^{۳۲} را قبول میکنند، استرینگ، عدد صحیح، فلوت، دابل، و کلس. بگذارید تمام این انواع دیتا را توضیح دهیم.

- به متغیرهایی که دو حالت راست^{۳۳} و غلط^{۳۴} دارند، متغیر بولی^{۳۵} میگویند.
 - به تکه های متنی که از الفباء تشکیل شده، رشته یا String میگویند. به متنی که به یک استرینگ میدهیم، لیترال^{۳۶} میگویند.
 - به اعداد صحیح بدون اعشار اینتجر^{۳۷} یا به کوتاه int میگویند.
 - به اعداد حقیقی با اعشار که دقت کمتری دارند، فلوت^{۳۸}.
 - به اعداد حقیقی با اعشار که دقت بالاتری دارند، دابل^{۳۹} میگویند.
 - از هر کلاسی می توان یک متغیر ساخت. درین صورت، نام کلاس، نوع متغیر میشود.
- اینها فقط چند نوع از متغیرهای پایتان هستند. همانطور که گفته شد، لازم نیست نوع متغیر را مشخص کنید چون پایتان، تایپ امن نیست. اما لازم است برای تعیین نوع آن، آنرا مقداردهی اولیه^{۴۰} نمایید.
- اما اگر بخواهیم چندین نوع دیتا، یا متغیر، را در یک متغیر نگاه داریم چه؟ بخش ۴.۳ درین مورد صحبت خواهد کرد.

^{۳۲} دیتا تایپ
^{۳۳} True
^{۳۴} False
^{۳۵} Boolean
^{۳۶} Literal
^{۳۷} Integer
^{۳۸} Floating
^{۳۹} Double
^{۴۰} Initialize

لیست، تاپل، دیکشنری

برای نگاه داشتن چندین نوع دیتا، یا چندین متغیر، در یک مکان، از لیست^{۴۱} استفاده میکنیم. لیستها به صورت زیر مقداردهی اولیه میشوند:

```
a_list = []
a_list = [1, 2, 3, 4]
a_list = ["Hello World", "Goodbye World"]
a_list = [1, 2, 3, 4, "Hello", "World"]
```

یک لیست وقتی مقداردهی اولیه شد، نباید با علامت مساوی، به آن مقدار اضافه کرد. بلکه، باید از `list.append` استفاده کرد:

```
a_list = [1]
print(a_list)
a_list.append(2)
print(a_list)
```

را بزنید تا نتیجه را ببینید. در زیر چند اسلوب لیست را میخوانید. `ctrl` + `↑` + `F10`

- `list.reverse`: لیست را برعکس یا به بعارت دیگر، معکوس میکند.
 - `list.copy` لیست را در لیست دیگر کپی میکند.
 - `list.pop` در ایندکس^{۴۲} داده شده، عضو را پاک میکند.
 - `list.sort` لیست را بر اساس الگوریتم **Merge Sort** مرتب میکند.
 - `len(list)` سائز لیست را به دست می آورد.
- یک لیست، میتواند لیستهای دیگری نیز در بر بگیرد:

```
multi_dimensional_list = [[], [], []]
multi_dimensional_list.append([])
```

^{۴۱} Lists

^{۴۲} Index - به شماره ی عضو ایندکس میگویند.

برای به دست آوردن عضو خاصی از لیست، از ایندکس آن استفاده میکنیم.

```
my_list = [2, 4, 6, 8, 10]
print(my_list[0])
```

ایندکسها از ۰ شروع میشوند تا سائز لیست منهای یک ادامه دارند. برای بدست آوردن چندین عضو از لیست، از علامت دو نقطه استفاده میکنیم:

```
my_list = [2, 4, 6, 8, 10]
print(my_list[0:3])
```

در کل، لیستها بهترین روش برای نگه داشتن دیتاهای زیاد هستند. اما برای دیتای کم و **غیر جهشی**^{۴۳}، از **تاپل**^{۴۴} استفاده میکنیم. تاپلها میتوانند به اندازه ی لیست دیتا نگه دارند، اما نمیتوان از آنها دیتا کم و زیاد کرد. تاپلها بدین صورت تعریف مقداردهی اولیه میشوند:

```
my_tuple = (R, G, B)
print(my_tupe[0:1])
```

مقلا برای رنگ یا موقیت یک فرگمنت در یک تصویر رستر از تاپل استفاده میشود. مانند لیستها، میتوان از `len()` برای بدست آوردن سائز تاپل استفاده کرد. دیکشنریها نیز مانند لیستها، برای نگه داشتن مقدار زیادی دیتا استفاده میشود. اما در دیکشنری، ایندکسها، عوض شماره، دارای نام هستند. دیکشنریها **همتا به همتا**^{۴۵} هستند. برای مقداردهی اولیه ی یک دیکشنری، از سینتکس زیر استفاده میکنیم:

```
my_dict = {"Name" : "Chubak",
           "Last Name" : "Bidpaa"}
print(my_dict["Name"])
```

در تمام زبانهای برنامه نویسی، زدن ↵ وسط خط کد اشکالی ندارد.

^{۴۳} Immutable
^{۴۴} Tuple
^{۴۵} Peer to Peer

مهمترین اسلوب دیکشنری `dictionary.items()` میباشد که آیتم های دیکشنری را برمیگرداند. در بخش لوپ در موردش صحبت خواهیم کرد صحبت از لوپ شد، وقت آن است که در مورد **بیانیه های**^{۴۶} پایتان صحبت کنیم.

بیانیه های شرطی

بیانیه های شرطی^{۴۷}، بخشهایی از پایتان هستند که به کد اجازه ی اجرا، یا در صورت عدم اجازه، اجازه ی اجرای کد دیگری را میدهند. **کلمات کلیدی**^{۴۸} که ما برای شرط گذاشتن روی جریان اجرای برنامه استفاده میکنیم، **if** و **elif** هستند. همیشه لازم نیست از دوتای دوم استفاده کرد، اما پیشنهاد میشود اگر مستلزم است، حتما از آنها استفاده کنید. کد زیر را ببینید:

```
pi = 3.14
r = 10
area= pi*r*r

if area > 20:
    print("The area is greater than 20.")
else:
    print("The area is not greater than 20.")
```

این کد، مساحت یک دایره با شعاع ۱۰ را حساب میکند و اگر این مساحت، بیشتر از ۲۰ است، میگوید مساحت بیشتر از ۲۰ است، وگرنه میگوید مساحت بیشتر از ۲۰ نیست. به همین سادگی، ما **جریان**^{۴۹} اجرای کد را تغییر دادیم.

پای چارم خودش اینکار را میکند، اما بین اول خط کلمه ی **if** و **else** و **اصطلاح**^a شرط، باید چهار فاصله باشد.

Expression^a

Statements^{۴۶}
Statements Conditional^{۴۷}
Keywords^{۴۸}
Flow Execution^{۴۹}

برای شرط گذاشتن، از آپریتور^{۵۰} هایی مانند < استفاده میکنیم. به اعدادی که آپریتور روی آنها تاثیر میگذارد، آپرند^{۵۱} میگویند. آپریتورهای شرطی پایتان به شرح زیرند:

بعلاوه	+
منها	-
ضرب	*
تقسیم	/
باقیمانده	%
بزرگتر	<
کوچکتر	>
بزرگتر مساوی	<=
کوچکتر مساوی	>=
مساوی	==
نا مساوی	!=
و	and
یا	or

جدول ۱.۳: آپریتورهای پایتان

آپریتورهای دیگری نیز داریم مانند آپریتورهای **Bit-wise** ولی الان به کار ما نمی آیند. میتوانید از کلمه ی کلیدی elif که مخفف If Else است برای افزایش شروط استفاده کنید:

```
if area > 20:
    print("The area is bigger than 20.")
elif (area < 15):
    print("The area is less than 15")
elif area < 10:
    print("The area is less than 10")
else:
```

Operator^{۵۰}
Operand^{۵۱}

```
print("The area is not bigger than 20.")
```

اما این فقط تنها بیانیه ی شرطی پایتان نیست. دو بیانیه ی شرطی دیگر داریم، که با if فرق زیادی دارند.

شرط، میتواند یک متغیر بولی باشد. مثلاً `bool = 1 < 10`. این متغیر، راست (True) می باشد چون ۱ کوچکتر از ۱۰ است.

بیانیه های چرخشی شرطی

بیانیه های چرخشی شرطی^{۵۲} بیانیه هایی هستند که تا شرط برقرار است، یک اصطلاح یا بیانیه را به صورت **نامحدود**^{۵۳} اجرا میکنند. گاهی این چرخش، بینهایت است. اما اکثر اوقات، شرط تمام شده و غلط (False) میشود. وقتی شرط، غلط میشود، چرخش تمام شده و بیانیه ی بعدی اجرا میشود. همچنین میتوانیم خودمان جریان چرخش را کنترل کرده، و به میل خود چرخش را تکرار کرده و یا بشکنیم.

دو کلمه ی کلیدی برای اینکار استفاده میشود، `for` و `while`. اولی مصارف دیگری هم دارد که به آن میپردازیم. اما بگذارید اول به `while` بپردازیم. سینتکس آن اینگونه است:

```
i = 0

while i < 50:
    print(str(i))
    i += 1
```

ابتدا ما به متغیر `i` عدد ۰ را میدهم. بعد میگوییم تا این متغیر، از ۵۰ کوچکتر از، ارزش متغیر را روی صفحه نمایش بده و در هر **بازتکرار**^{۵۴}، ارزش ۱ را به متغیر اضافه میکنیم. وقتی متغیر به ۵۰ رسید، چرخش تمام میشود و به بیانیه ی بعدی میرسد.

Conditional Loops^{۵۲}
Indefinitely^{۵۳}
Iteration^{۵۴}

تابع `print()` نمیتواند جز استرینگ لیترال و استرینگ، چیز دیگری را در صفحه به نمایش بگذارد. با استفاده از تابع `str()` متغیرهای عددی را به استرینگ لیترال تبدیل میکنیم.

میتوانید با استفاده از کلمه ی کلیدی `and` یک شرط دیگر اضافه کنید:

```
i = 0

while i < 50 and i < 25:
    print(str(i))
    i += 1
```

اینگونه، فقط در صورتی متغیر روی صفحه پرینت میشود که بین ۲۵ و ۵۰ باشد. `or` هم دو یا چند شرط را در صورتی اجرا میکند که یکی از آنها، راست باشد. کلمه ی کلیدی بعدی که داریم، `for` میباشد. این کلمه بیشتر برای دسترسی به لیست، دیکشنری و تاپل به کار میرود؟؟ اما برای چرخش برای `n` بار از سینتکس زیر استفاده میکنیم:

```
for i in range(n):
    print(str(i))
```

تابع `range(m, n)` یک لیست قابل بازتکرار بین `m` و `n` ایجاد میکند. اگر پارامتر اول را به آن ندهیم، یک لیست قابل بازتکرار بین ۰ و `n` ایجاد میکند. و متغیر ارزش `i` را در هر بازتکرار، بر اساس لیست ساخته شده مشخص میکند. این بیانیه ی چرخشی شرطی نیست، بلکه بیانیه ی چرخشی بازتکراریست. در بخش بعد، از کلمه ی کلیدی `for` برای دسترسی به اعضای لیست، تاپل، و دیکشنری استفاده میکنیم.

دسترسی به لیست، دیکشنری و تاپل

برای دسترسی به اعضای یک لیست، تاپل، دیکشنری، یا هر شیء قابل بازتکرار^{۵۵} دیگری، از `for` استفاده میکنیم. مثال برای لیست اینگونه است:

^{۵۵}Iterable

```

one_dim_list = [1, 2, 3]
two_dim_list = [[1, 2, 3], [4, 5, 6]]

for i in one_dim_list:
    print(i)

for list in two_dim_list:
    for i in list:
        print(i)

```

همانطور که مشاهده میکنید، ما با استفاده از کلمه ی کلیدی `in` توانستیم به اعضای لیست `one_dim_list` دسترسی پیدا کنیم و آنها را روی صفحه پرینت کنیم. سپس، ما با استفاده از یک بیانیه ی **لانه ای**^{۵۶} توانستیم یک لیست دو بعدی را روی صفحه پرینت بگیریم.

تمام بیانیه های شرطی را میتوان لانه کرد، اما اگر که اشتباهی صورت بگیرد، سوریزی پشته^a صورت میپذیرد.

Stack Overflow^a

ما میتوانیم با استفاده از دو کلمه ی کلیدی `break` و `continue` بر جریان چرخشمان تاثیر بگذاریم.

```

n = 0
while True:
    n += 1

    if (n > 20):
        break

```

این کد، همیشه صحیح است، پس همواره اجرا میشود. اما اگر متغیر ما، بیشتر از ۲۰ شود، زنجیر میشکند و چرخش پایان میابد. `continue` نیز مانند `break` است، فقط حلقه را نمیشکند، بلکه کاری میکند که حلقه دوباره بازتکرار شود.

Nested Statement^{۵۶}

تابع در پایتان

در بخش پردایم فانکشنال، در مورد توابع در برنامه نویسی صحبت کردیم. در پایتان، تابع بلوکه ای از کد است که با خواندن آن، یک یا یک امر صورت میپذیرد، یا یک ارزش باز گردانده میشود، یا هر دو. پایتان دارای ۶۸ تابع از پیش ساخته شده است، و ما خودمان میتوانیم تا هرچقدر لازم داریم، تابع بسازیم. برای اینکار، از کلمه ی کلیدی `def` استفاده میکنیم:

```
def first_functions():
    pi = 3.14
    r = 10
    area = pi * r * r

    print(area)
```

```
def second_function(r):
    pi = 3.14
    area = pi * r * r

    return area
```

همانطور که مشاهده میکنید، تابع اولی، نه پارامتر قبول میکند، نه ارزشی را باز میگرداند. اما یک عملیات پرینت انجام میدهد. به این نوع توابع، همانطور که گفتیم، ووید میگویند. تابع دوم یک تابع فلوت است، چون یک ارزش فلوت باز میگرداند. و ابتدا شعاع دایره را به عنوان پارامتر میپذیرد. یک تابع را میتوان در خودش خواند. به این امر **تابع بازگشتی**^{۵۷} میگویند. مثلاً برای بدست آوردن فاکتوریل یک عدد:

```
def factorial(n):
    if n == 1:
        return n
    else:
```

```
return n*factorial(n - 1)
```

اگر شرایط بازگشت در تابع محیا نباشد همانطور که در بخش قبل گفتیم، سرریزی پشته صورت میگیرد. در مورد پشته و هرم در بخش C++ صحبت خواهیم کرد.