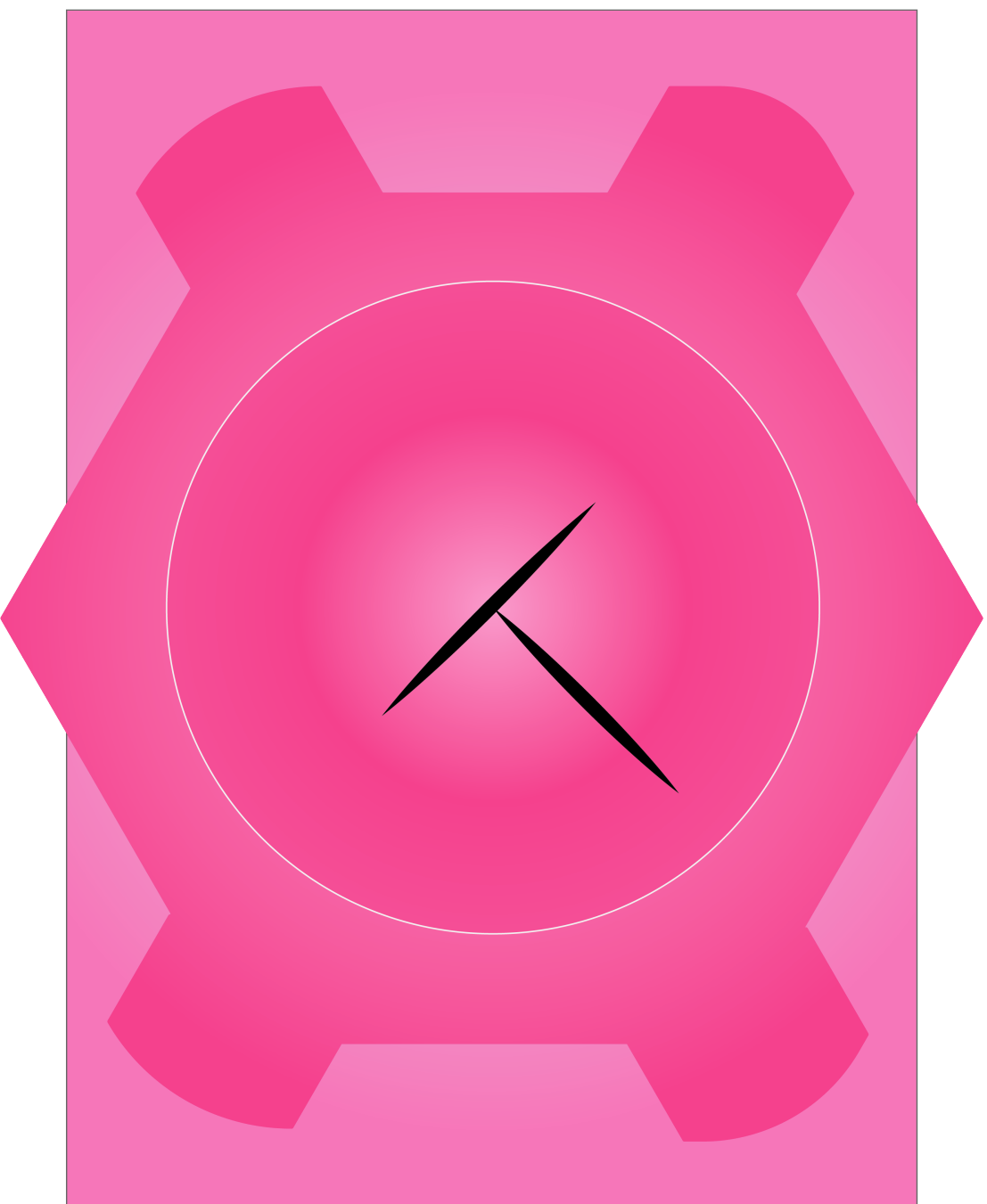


بازی سازی و برنامه نویسی گرافیکی پایه

چوبک بیدپا



عرضه شده تحت لیسانس MIT
 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 فایلها 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 و درضمن، در صورت بیشتر شدن تعداد خریده‌ها، قیمت کاهش میابد.

قراردادهای کتاب

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



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

نکته اینجااست 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');
```

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
۱۴	۶.۲ پایان فصل ریاضی
۱۵	۳ نگاهی کوتاه به برنامه نویسی
۱۵	۱.۳ برنامه نویسی فانکشنال
۱۷	۲.۳ برنامه نویسی شیء گرا
۱۷	۳.۳ کتابخانه ها
۱۸	۴.۳ پایتان
۳۶	۵.۳ C++
۵۳	۶.۳ سیستمهای کنترل نسخه
۵۳	۷.۳ پایان فصل برنامه نویسی
۵۵	۴ برنامه نویسی گرافیکی، مفهومات، الگوریتمها

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 گرافیکی به صورت حرفه ای و پولساز خبر دارد. سعی من درینست که با نوشتن در مورد این دو دیسپلین دوست داشتنی، فرهنگ آنها را

GNU^۱

Linux^۲

Mozilla^۳

Firefox^۴

Freedom of Information^۵

در کشور اشاعه بدهم.

از سابقه ام در برنامه نویسی و بازی سازی بگویم. من از شانزده سالگی^۶ 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 ++C، پایتان، و C را میدانم و با زبان اسکریپت نویسی چندین نرم افزار آشنایی دارم. به علوم نرم مانند ادبیات انگلیسی آشنایی آکادمیک دارم و در حال حاضر دانشجوی برنامه نویسی ام.

سابقه ی من در برنامه نویسی گرافیکی کمتر است. دو سال پیش بود که با نرم افزار افتر افکتس^۷ آشنا شدم و به صرافت نوشتن پلاگین برایش افتادم، و طی این امر، با کتابخانه ی Cinder برای ++C آشنا شدم. و از آنجا بود که با زبان Processing و شیدرها آشنا گردیدم. الان تسلط کافی برای آموزش پایه ی شیدرها و زبانها و کتابخانه های برنامه نویسی گرافیکی دارم.

بگذارید در مورد چارچوب کتاب کمی صحبت کنم. در این کتاب، دو بخش داریم، برنامه نویسی گرافیکی، 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^۸ آموزش داده خواهد شد. اگر فرصت شد، آموزشی کوتاه 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 پر هیر و گیر، اگر عشقی^۹ به چیزی دارید که به شما آرامش میدهد، نیکوست. و اگر این کتاب برای پیدا کردن این عشق کمک میکند، خوشحالم.

چوبک بیدپا مشهد - ۱۳۹۷

^۶ الان بیست و پنج ساله ام.

^۷ After Effects

^۸ تلفظ این انجین، گدو میباشد.

^۹ Passion

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 میگوییم. 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:

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

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:

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

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

$$(f \times g)(x) = f(x) \times g(x)$$

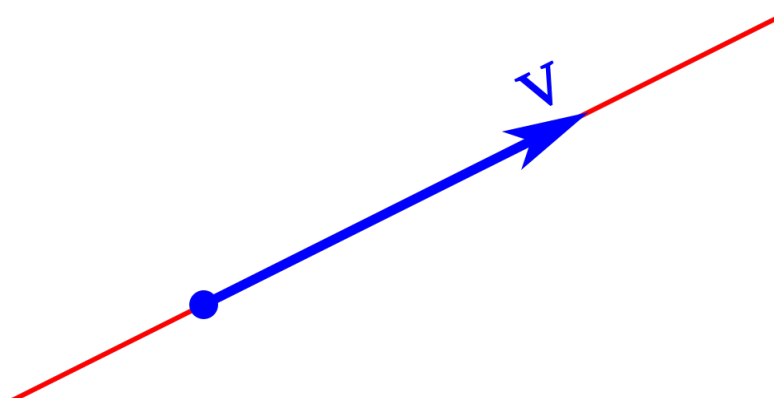
Function^۱
Inverse^۲

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

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 ۲ 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:

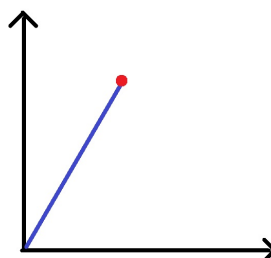
Domain^۳
Range^۴
Vector^۵



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:

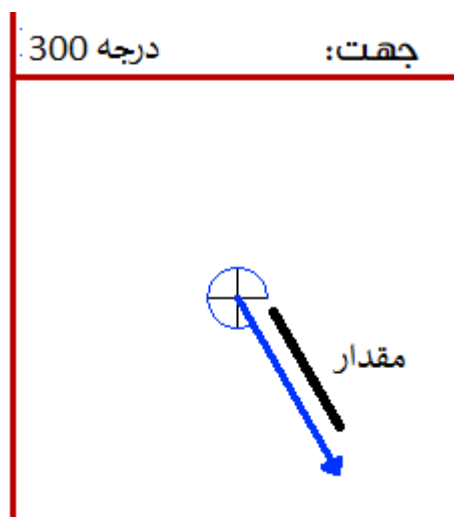
$$\vec{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$$

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



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:

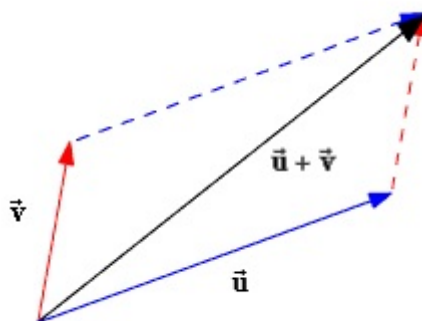


برای به دست آوردن مقدار بردار ازین فرمول استفاده میکنیم:

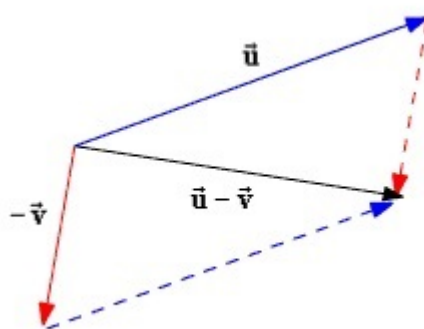
$$|\vec{V}| = \sqrt{x^2 + y^2}$$

دو بردار را میتوان به صورت زیر جمع کرد:

Direction^v
Magnitude[^]



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:

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

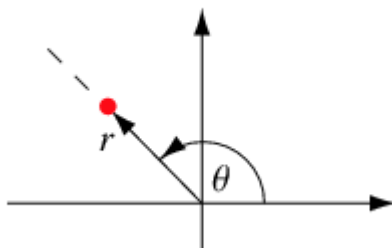
و ضرب صلیبی ازین فرمول استفاده میکنیم.

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

که \vec{n} برداری پایه^{۱۱} 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 پایه ی 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**^{۱۲} 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:

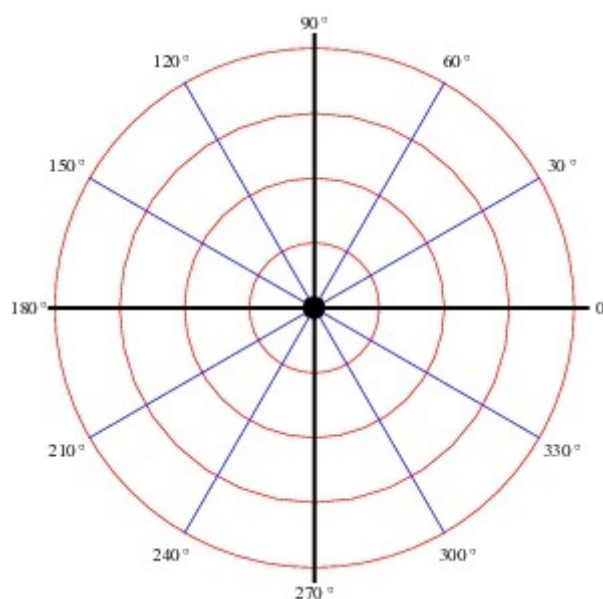


در برنامه نویسی گرافیکی، دستگاه مختصات قطبی کاربردهای زیادی دارد. اما در کامپیوتر، **پیکسلها**^{۱۳} در دستگاه مختصات دکارتی قرار دارند. حلال مشکلات ما، مثلثات است. یک دایره ی واحد را در دستگاه مختصات قطبی کنید که شعاعش ۱ میباشد:

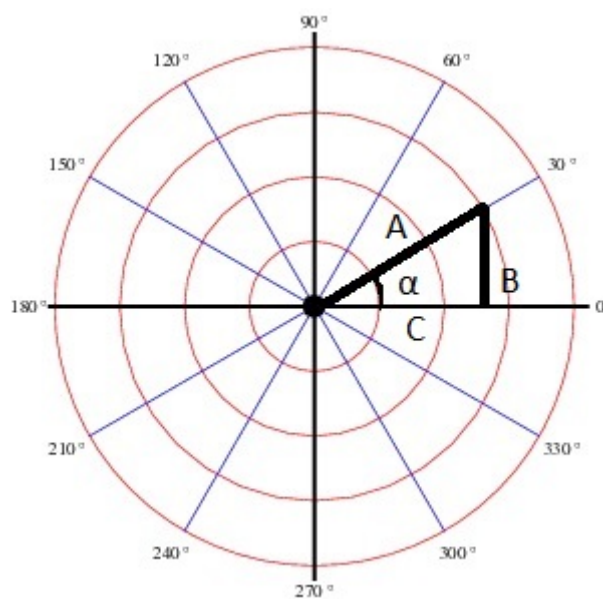
^{۱۱} Basis Vector

^{۱۲} Polar Coordinate System

^{۱۳} 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**^{۱۴} 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**^{۱۵} 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 گرافیکی و بازی سازی کلی.

^{۱۴} Radians

^{۱۵} Trigonometric Identities

۴.۲ ماتریسها

به آرایه هایی از اعداد که به صورت n سطر و m ستون به نمایش در می آیند، **ماتریس**^{۱۶} میگویند. یک ماتریس را به این صورت نشان میدهند:

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

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

$$A_{m,n} \pm B_{m,n} = \begin{bmatrix} a_{1,1} \pm b_{1,1} & a_{1,2} \pm b_{1,2} & \cdots & a_{1,n} \pm b_{1,n} \\ a_{2,1} \pm b_{2,1} & a_{2,2} \pm b_{2,2} & \cdots & a_{2,n} \pm b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} \pm b_{m,1} & a_{m,2} \pm b_{m,2} & \cdots & a_{m,n} \pm b_{m,n} \end{bmatrix}$$

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. مثلاً ضرب دو ماتریس 2×2 به صورت زیر است:

$$A_{2,2}B_{2,2} = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{2,1} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{2,1} + a_{2,2}b_{2,2} \end{bmatrix}$$

یکی دیگر از عملیتهای ماتریسی، **دترمینان**^{۱۷} است. برای احتساب دترمینان ماتریسهای بزرگتر از 3×3 الگوریتمهای زیادی مانند **دیکامپوزیشن**^{۱۸} وجود دارد که خود آن توسط افراد مختلفی در طول سالها بهسازی گشته است، اما راه ساده ای برای به دست آوردن دترمینان 2×2 وجود دارد که به شرح زیر است:

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

^{۱۶} Matrix^{۱۷} Determinant^{۱۸} Decomposition

$$|A| = AD - BC$$

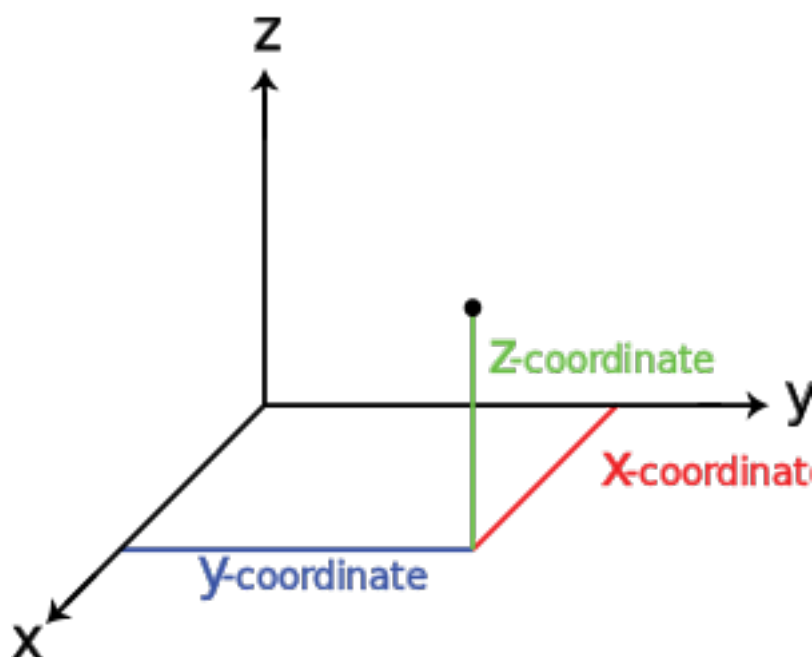
به I_n ماتریس هویت^{۱۹} میگویند و مثلاً I_3 به صورت زیر تعریف میشود:

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

ما در برنامه نویسی گرافیکی 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:



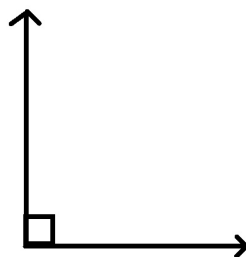
توجه کنید که در بعضی از نرم افزارها جای Y با Z عوض میشود.

^{۱۹}Identity

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**^{۲۰} 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 آن دو، 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:

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

^{۲۰} Orthogonality

۶.۲ پایان فصل ریاضی

کلید یادگرفتن ریاضی یک چیز است: ⁿتمرین! یادتان نرود که حافظه، چیز است فرار، و هر لحظه ممکن است همین چیزهای کمی که از بنده ی حقیر 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 المثال 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. بعد از آن، نگاهی می اندازیم به سینتکس^۵ زبان پایتان^۶ 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^۷ و لیسپ^۸ 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^۹ است که پارامتر^{۱۰} داده شده را با تغییرات، باز میگردانند. این تغییرات میتواند عملیتهای جمع و تفریق، ضرب و تقسیم، باقیمانده، و یا تغییر نوع پارامتر مثلا از عدد صحیح

^۱ Programming

^۲ Paradigm

^۳ Functional

^۴ Object Oriented Programming

^۵ به دستور زبانی یک زبان برنامه نویسی Syntax گفته میشود.

^۶ Python

^۷ FORTRAN

^۸ Lisp

^۹ Instructions

^{۱۰} Parameter

به عدد حقیقی، و یا هرچیز دیگری باشد. برای اجرای تابع، آنرا **میخوانیم**^{۱۱} و به پارامتری که به آن می‌دهیم، **آرگومان**^{۱۲} می‌گوییم. اینستراکشن **سِت** 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**^{۱۳} 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 نمی‌گردانند. به این نوع از توابع **ووید**^{۱۴} می‌گویند. بعضی از زبانها، **تایپ ثابت**^{۱۵} هستند و باید نوع ارزشهای باز گرداننده را مشخص کرد. C++ یکی ازین نوع زبانهاست. بعضی از زبانها **تایپ دینامیک**^{۱۶} 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 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**^{۱۷} هایی هستند که شامل توابع و کلاسها 3.2 و دیگر کدهایی هستند که به برنامه نویس کمک میکنند خود را تکرار نکند.

خود را تکرار نکنید.^{۱۸}

قانون پلاتینیوم برنامه نویسی، اینست. هرگز چیزی که در یک کتابخانه موجود است را ننویسید. مثلاً عوض اینکه در زبان **جاوا**^{۱۹} عوض نوشتن صدها خط کد برای به دست آوردن یک تابع ماتریس، میتوان از کتابخانه ی JAMA استفاده کرد.

Call^{۱۱}

Argument^{۱۲}

Factorial^{۱۳}

Void^{۱۴}

Statically Typed^{۱۵}

Dynamically Typed^{۱۶}

Library^{۱۷}

DRY - Don't Repeat Yourself^{۱۸}

Java^{۱۹}

۲.۳ برنامه نویسی شیء گرا

برنامه نویسی شیء گرا بر پایه ی کانسپت کلاس^{۲۰} 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 چنده هزار عابر پیاده که سوار ماشین میشوند. درین تشبیه، به ماشین **اسلوب**^{۲۱} و به عابر پیاده **خواص**^{۲۲} 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**^{۲۳} میگویند. مثلاً خیابان ولیعصر، یک شیء خیابان است. 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. یک کلاس میتواند فرزند یک کلاس دیگر باشد. در بعضی از زبانها، یک کلاس میتواند فرزند چندین کلاس باشد. C++ یکی ازین زبانهاست. 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**^{۲۴} انجام میشود. و یا به سادگی میخواهیم یک مدرسه ی قدیمی را بکوبیم و خراب کنیم. برای این کار از اسلوبی به نام **خراب کننده**^{۲۵} استفاده میکنیم. یگ کلاس میتواند چنین سازنده داشته باشد، ولی فقط یک خراب کننده میتواند داشته باشد. در بخش بعدی در مورد کتابخانه ها صحبت خواهیم کرد.

۳.۳ کتابخانه ها

به مجموعه توابع و کلاسهای از قبل آماده شده، کتابخانه میگویند.

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

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**^{۲۶} عرضه شوند. کتابخانه های پایتان متنی، و کتابخانه های C++ باینری هستند. گاهی کتابخانه هایی به صورت متنی نیز عرضه میشوند.

۴.۳ پایتان

زبان پایتان در سال ۱۹۹۹ برای بار اول عرضه شد و در هنگام نوشتن این کتاب، در ورژن ۳.۷.۱ به سر میبرد. درین فصل، فقط قطره ای از دریای این زبان را آموزش میدهم. برای آموزش بهتر زبان، به کتاب مخصوص مراجعه کنید.

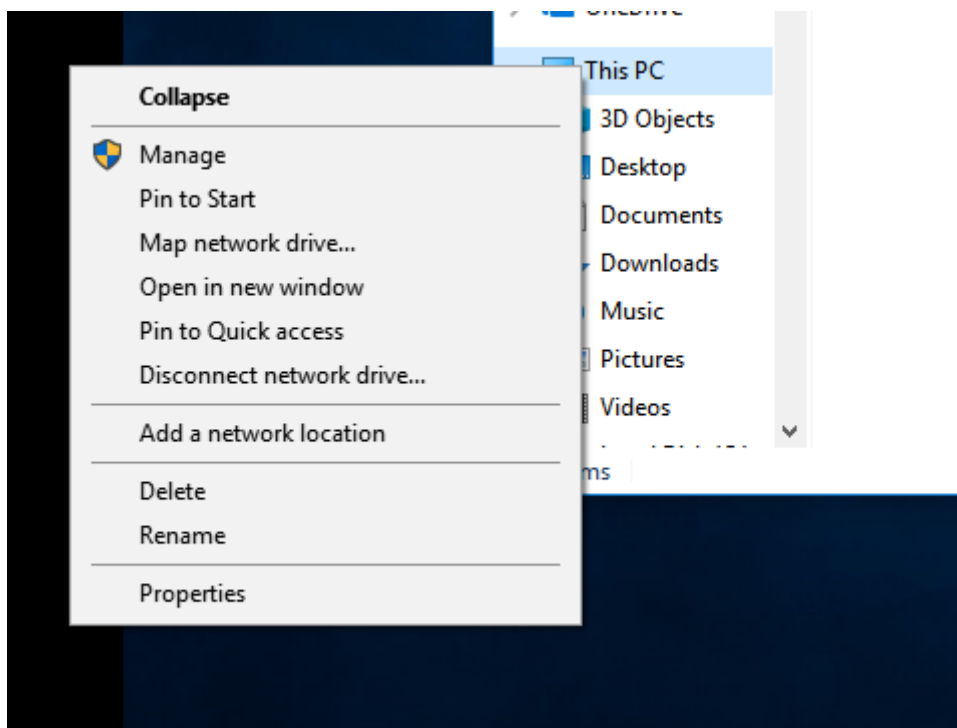
پایتان زبانی کاملاً مدرن، قابل انعطاف، یکدست، و جذاب و ساده میباشد که برای از اتوماسیون گرفته تا بازی سازی، کاربرد دارد. اینستراکشن های پایتان **ترجمه**^{۲۷} میشوند، یعنی لازم نیست که از قبل به زبان اسمبلی یا ماشین در بیایند، بلکه، در حین اجرا به زبانهایی مثل C یا Java ترجمه میشوند و بعد خط به خط اجرا میشوند. عرضه ی اصلی پایتان که ما از آن استفاده میکنیم، از C++ استفاده میکند.

نصب پایتان

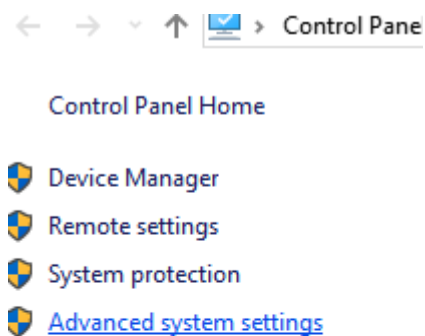
۱. به این صفحه بروید و پایتان را دانلود کنید: <https://www.python.org/downloads/>

۲. آنرا نصب کنید.

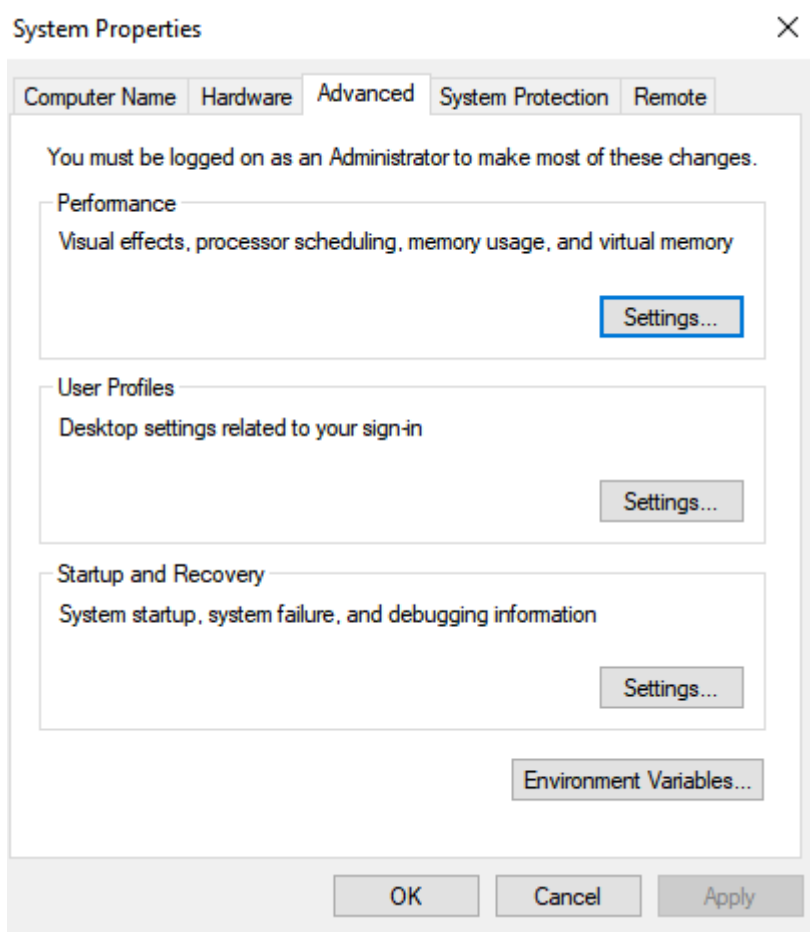
۳. از فایل اکسپلورر مانند زیر روی Properties کلیک کنید:



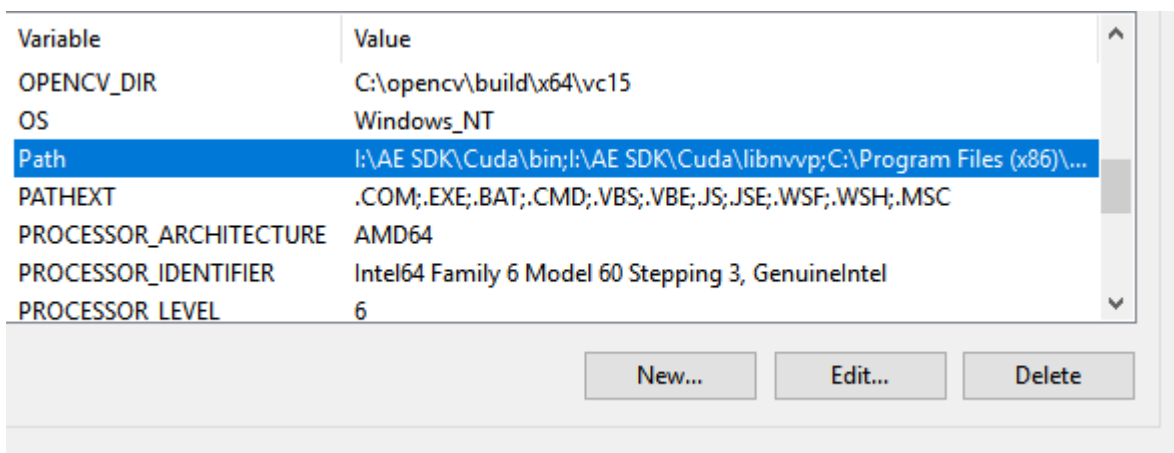
۴. روی Advanced System Settings کلیک کنید.



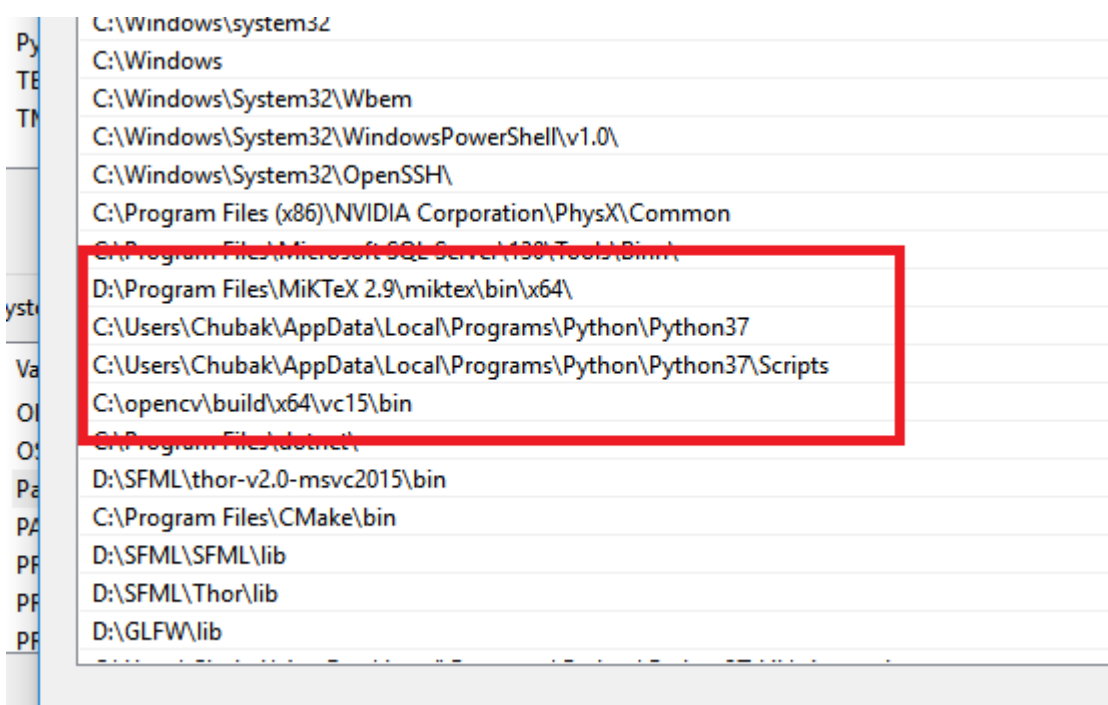
۵. روی گزینه Environment Variables کلیک کنید.



۶. روی Path کلیک کنید.



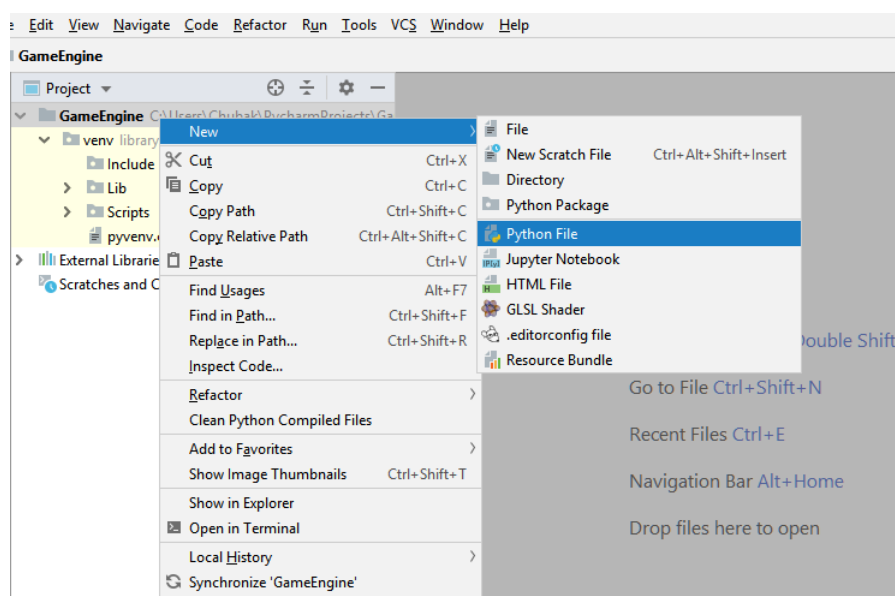
۷. دو گزینه ی زیر را به آن اضافه کنید.



برای نوشتن کد پایتان احتیاج به یک محیط گسترش مجتمع^{۲۸} دارید. من PyCharm Community را پیشنهاد میکنم که کاملاً مجانیست. آنرا میتوانید از <https://www.jetbrains.com/pycharm/> دریافت کنید.

^{۲۸}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 ^{۲۹}متغیر است. متغیرها مانند ظروفی هستند که میتوان در آنها هر چیزی ریخت. چون پایتان زبان قایپ امن^{۳۰} نیست، میتوان هر نوع دیتایی را داخل یک متغیر جا داد.

^{۲۹}Variable
^{۳۰}Type Safe

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

همانطور که میبینید، متغیرهای پایتان هر نوع دیتایی^{۳۱} را قبول میکنند، استرینگ، عدد صحیح، فلوت و کلس. بگذارید تمام این انواع دیتا را توضیح دهم.

- به متغیرهایی که دو حالت راست^{۳۲} و غلط^{۳۳} دارند، متغیر بولی^{۳۴} میگویند.
- به تکه های متنی که از الفباء تشکیل شده، رشته یا String میگویند. به متنی که به یک استرینگ میدهیم، لیترال^{۳۵} میگویند.
- به اعداد صحیح بدون اعشار اینتجر^{۳۶} یا به کوتاه int میگویند.
- به عدد حقیقی با اعشار فلوت^{۳۷} 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^{۳۸} نمایید.

اما اگر بخواهیم چندین نوع دیتا، یا متغیر، را در یک متغیر نگاه داریم چه؟ بخش ۴.۳ درین مورد صحبت خواهد کرد.

^{۳۱} دیتا تایپ
^{۳۲} True
^{۳۳} False
^{۳۴} Boolean
^{۳۵} Literal
^{۳۶} Integer
^{۳۷} Float
^{۳۸} Initialize

لیست، تاپل، دیکشنری

برای نگاه داشتن چندین نوع دیتا، یا چندین متغیر، در یک مکان، از **لیست**^{۳۹} 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 بعیارت دیگر، معکوس میکند.

- `list.copy` لیست را در لیست دیگر کپی میکند.

- `list.pop` در **ایندکس**^{۴۰} 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([])
```

^{۳۹}Lists

^{۴۰}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**^{۴۱}، از **تاپل**^{۴۲} 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 موقعیت 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**^{۴۳} هستند. برای مقداردهی اولیه ی یک دیکشنری، از تفریق 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.

^{۴۱} Immutable
^{۴۲} Tuple
^{۴۳} 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**^{۴۴} پایتان صحبت کنیم.

بیانیه های شرطی

بیانیه های شرطی^{۴۵}، بخشهایی از پایتان هستند که به کد اجازه ی اجرا، یا در صورت عدم اجازه، اجازه ی اجرای کد دیگری را میدهند.

کلمات کلیدی^{۴۶} که ما برای شرط گذاشتن روی جریان اجرای برنامه استفاده میکنیم، **if** و **else** 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**^{۴۷} 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^{۴۴}
Conditional Statements^{۴۵}
Keywords^{۴۶}
Execution Flow^{۴۷}

برای شرط گذاشتن، از آپریتور^{۴۸} هایی مانند < استفاده میکنیم. به اعدادی که آپریتور روی آنها تاثیر میگذارد، آپرند^{۴۹} میگویند. آپریتورهای شرطی پایتان به شرح زیرند:

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

جدول ۱.۳: آپریتورهای پایتان

آپریتورهای دیگری نیز داریم مانند آپریتورهای **Bitwise** 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:
```

Operator^{۴۸}
Operand^{۴۹}

```
print("The area is less than 10")
else:
    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^{۵۰} بیانیه هایی هستند که تا شرط برقرار است، یک اصطلاح یا بیانیه را به صورت نا محدود^{۵۱} 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^{۵۲}، ارزش ۱ را به متغیر اضافه میکنیم. وقتی متغیر به ۵۰ رسید، چرخش تمام میشود و به بیانیه ی بعدی میرسد.

Conditional Loops^{۵۰}
Indefinitely^{۵۱}
Iteration^{۵۲}

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 3.4 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^{۵۳} دیگری، از `for` استفاده میکنیم. مثال برای لیست اینگونه است:

^{۵۳}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**^{۵۴} توانستیم یک لیست دو بعدی را روی صفحه پرینت بگیریم.

تمام بیانیه های چرخشی را میتوان لانه کرد، اما اگر که اشتباهی صورت بگیرد، **سرریزی پشته**^a صورت میپذیرد. پشته بخشی از زبان C است که کامپایلر پایتان در آن نوشته شده است. ۵.۳ را بخوانید.

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;
```

Nested Statement^{۵۴}

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.

تابع در پایتان

در بخش پَرَدایم 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 بازگردانده میشود، یا هردو. 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 می گرداند. 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 می گرداند. و ابتدا شعاع دایره را به عنوان پارامتر میپذیرد. یک تابع را میتوان در خودش بخواند. به این امر **تابع بازگشتی**^{۵۵} میگویند. مثلاً برای بدست آوردن فاکتوریل یک عدد:

^{۵۵} Recursive Function

```
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. تا این حد بدانید که پایتان، **مدیریت حافظه**^{۵۶} را خودش انجام میدهد و نیازی به این کار توسط شما نیست. در C++ ما متغیرهایی داریم که به آدرس یک متغیر دیگر در RAM اشاره دارند، اما در پایتان، ما همچنین چیزی نداریم.

توجه داشته باشید که **اسکوپ**^a متغیرها در تابع، مخصوص خودشان است. اگر متغیری را در یک تابع تعیین کردید، نمیتوانید آنرا بیرون از تابع استفاده کنید. و همچنین اگر متغیری را داخل یک بلاک بیانیه ی شرطی یا چرخش شرطی تعیین کردید، آن متغیر، بیرون از آن بلاک، ارزشی ندارد.

Scope^a

فایل در پایتان

برای باز کردن یک فایل در پایتان، از روش زیر استفاده میکنیم:

```
f = open(filename, mode)
```

فایل حتما باید در فولدر اسکریپتی که داریم مینویسم باشد. نام فایل یک استرینگ لیترال است پس باید داخل علامت نقل قول باشد. حالت باز کردن فایل بستگی به عملیاتی که میخواهیم روی فایل انجام دهیم دارد. در جدول زیر، حالات باز کردن فایلها را ببینید:

باز کردن یک فایل متنی برای خواندن	'r'
باز کردن فایل متنی برای نوشتن	'w'
حالت ساخت فایل. اگر فایل وجود دارد، عملیات شکست میخورد.	'x'
باز کردن فایل برای اضافه کردن متن به فایل.	'a'
باز کردن در حالت متنی.	't'
باز کردن در حالت باینری. برای هر فایلی جز فایل متنی.	'b'

جدول ۲.۳: حالت‌های باز کردن فایل در پایتان

میتوانید با علامت بعلاوه، حالتها را با هم ترکیب کنید مثلاً `open("scene.jpg", 'w+b')` برای بستن فایل از اسلوب `file.close()` استفاده کنید. نوشتن روی فایلها به صورت زیر انجام میپذیرد:

```
with open("test.txt", 'w', encoding = 'utf-8') as f:
    f.write("my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")
```

کلاسهای پایتان

در مورد کلاسها در بخش پردایم شیء گرا حرف زدیم. در پایتان، کلاس مجموعه ای از توابع و متغیرهاست که به آنها اسلوب و خواص میگوییم. یک کلاس میتواند فرزند یک کلاس دیگر باشد. آنگاه کلاس فرزند تمام متدها و خواصهای کلاس مادر را به ارث میبرد. یک کلاس به صورت زیر درست میشود:

```
class Rectangle:
    def __init__(self, color, filled, width, length):
        self.__color = color
        self.__filled = filled
        self.__width = width
        self.__length = length
```

```

def get_color(self):
    return self.__color

def set_color(self, color):
    return self.__color = color

def is_filled(self):
    self.__filled

def set_filled(self, filled):
    return self.__filled

def get_area():
    return self.__width * self.__length

```

کلاس `rectangle` دارای یک اسلوب سازنده به نام `__int__` است که چهار خواص مستطیل را تعیین میکند. و دارای پنج اسلوب دیگر است که هر کدام یک عمل متفاوت انجام میدهند.

برای درست کردن یک فرزند از کلاس مستطیل، کافیت به صورت `class Square(Rectangle)` عمل کنیم. یکی از کارهایی که میشود در برنامه نویسی شی گرا کرد، چند ریخت^{۵۷} کردن است. که عبارتست از تعیین یک متد در کلاس فرزند که نام یک متد در کلاس مادر را دارد، ولی خواصش متفاوت است.

ماژولها و پکیجهای پایتان

پایتان، به خاطر پکیجهایی (یا همان کتابخانه ها) که برایش عرضه میشود، نامدار است. پایتان بدون پکیج، یک زبان متوسط است، مانند `Ruby`، `Ada`، `Lua`، `Perl` و... اما در به خاطر پکیجهایی که برای این زبان در طول سالها عرضه شده، و سادگی نصب این پکیجها، همه برای کارهای کوچک و بزرگ به پایتان روی می آورند.

فلسفه ی نرم افزار مدیریت پکیج پایتان، یعنی `pip` — «خود را تکرار نکنید»^{۵۸} است. برای نصب یک پکیج روی پایتان، کافیت:

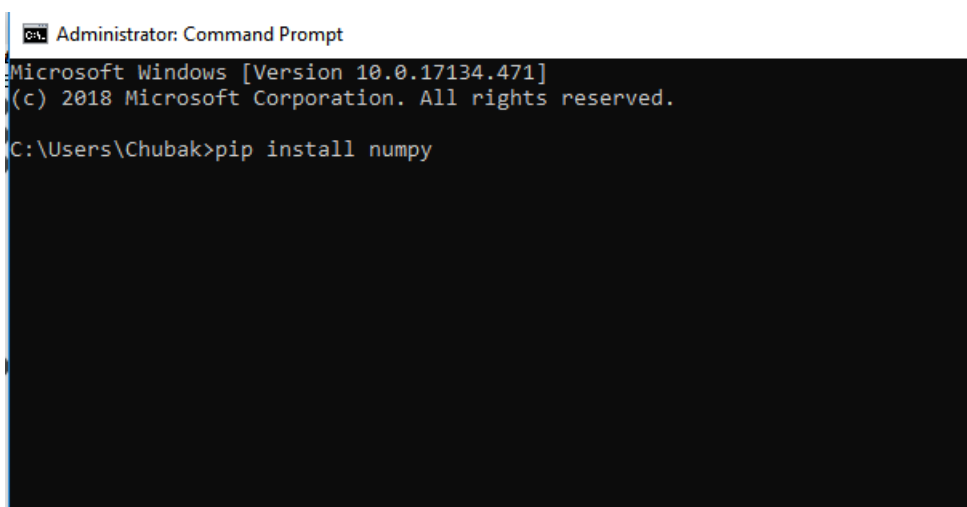
Polymorphism^{۵۷}
 DRY^{۵۸}

۱. CMD را باز کنید.

۲. بنویسید `pip install package-name`.

۳.  را بزنید.

۴. پکیج روی سیستم شما نصب خواهد شد.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.471]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Chubak>pip install numpy
```

وقتی پکیج روی کامپیوترمان نصب شد، میتوانیم با فرمانهای زیر، در اول یا هرجای فایل اسکریپت، پکیج را وارد^{۵۹} اسکریپتمان کنیم.

```
import numpy
from numpy import class
import numpy as math
```

چندین نوع وارد کردن وجود دارد. اولی، همانطور که میبینید، وارد کردن کل پکیج است. وقتی اینکار انجام شد. میتوانیم با استفاده از فرمان `numpy.method()` اسلوبها و خواصهای مورد نظر خود را استفاده نماییم. روش دوم، وارد کردن کلاسی خاص با استفاده از فرمان `from x import y` است. و فرمان سوم، نام پکیج مورد نظر را تغییر میدهد.

^{۵۹}Import

هر پکیج راهنمای خود را دارد که در اینترنت پیدا میشود. در این کتاب، وقتی از یک پکیج نام میبریم، دیگر نمیگوییم آنرا چگونه نصب کنید. وظیفه ی خودتان است که پکیج را نصب کرده و وارد نرم افزار کنید.

پایان بخش پایتان

با پایان رسیدن بخش پایتان، لازم است یادآوری کنم که من فقط پوسته ی پایتان را خراش دادم. اگر میخواهید بیشتر بدانید، کتابهای متعددی برای اینکار وجود دارند.

۵.۳ C++

زبان C++ در اوایل دهه ی هشتاد توسط دکتر بژورن اشتراشراب، مهندس دانمارکی، ابداع شد. فرق C++ با C در برنامه نویسی شیء گراست. این زبان، عوض پایتان، **کامپایل**^{۶۰} میشود. یعنی، توسط یک نرم افزار به نام کامپایلر که به زبان C نوشته شده، به زبان اسمبلی تبدیل میشود و سپس سیستم عامل آنرا به زبان ماشین تبدیل کرده و آنرا اجرا میکند.

کامپایلرهای زیادی برای C++ وجود دارند. دو تا از بزرگترین کامپایلرها، GCC روی لینکس و Microsoft Visual C++ روی ویندوز است. البته روی ویندوز میتوان از MinGW یا Cygwin هم استفاده کرد.

محیط گسترش مجتمع روی ویندوز، برای C++ زیاد است. اما ما از Microsoft Visual Studio استفاده میکنیم. پیشنهاد میکنم این نرم افزار را به صورت پائریت شده از بازار بخرید، بلکه، پیشنهاد میکنم نسخه ی مجانی Community آنرا از سایت مایکروسافت دانلود کرده و آنرا نصب کنید.

<https://visualstudio.microsoft.com/vs/community/>

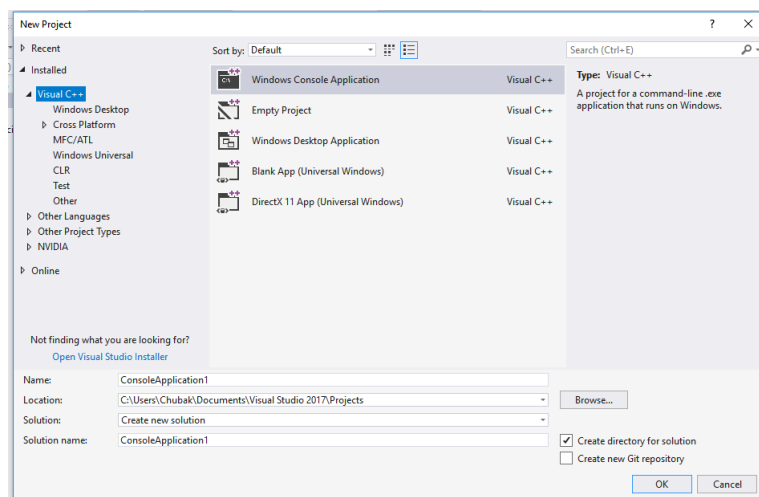
یادتان باشد هنگام نصب، Visual C++ Tools را نصب کنید. وگرنه از منوی Tools میتوانید پکیج را نصب کنید. یکی از نیکوییهای ویژوال استودیو، NuGet Package Manager است که به شما اجازه میدهد فایل‌های **سری**^{۶۱} کتابخانه ها را دانلود کنید. اما ما خودمان فایل‌های **سری** را دانلود کرده و **اضافه**^{۶۲} میکنیم. به معرفی C++ پردازیم. چون خیلی از چیزها تکرار از بخش پایتان میباشد، این بخش بسیار کوتاه خواهد بود.

^{۶۰}Compile

^{۶۱}Header

^{۶۲}Include

برای شروع یک پروژه ی جدید از بخش File -> New Project یک پروژه ی Windows Console Application بسازید.



سینتکس

یک برنامه ی C++ به صورت زیر نوشته میشود:

```
#include "pch.h"
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}
```

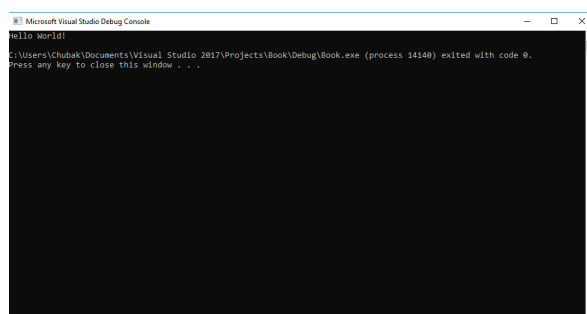
کلمه ی کلیدی `#include` فایل های سری برنامه را تعیین میکنند. **اعلامیات**^{۶۳} توابع، متغیرها، و کلاسهای برنامه هستند. **تعیینات**^{۶۴} برنامه در فایل های سورس برنامه قرار دارند. هر فایل سری با پسوند `.h` یک فایل سورس با همان نام با پسوند `.cpp` دارد که حاوی تعیینات برنامه است.

^{۶۳} Declaration
^{۶۴} Definitions

تابع `main` تابع اصلی برنامه است. اگر یک برنامه، تابع اصلی نداشته باشد، کتابخانه به حساب می آید. در `C++` برخلاف پایتان نمیتوانیم یک کد را خارج از تابع به اجرا درآوریم، و تابع اصلی برای همین کار است.

متد `std::cout` (تلفظش سی-اوت میباشد) یکی از اسلوبهای کتابخانه `STL`^{۶۵} میباشد. `std` که با دو تا دونقطه از اسلوب سی-اوت جدا شده، **فضانام**^{۶۶} اسلوب میباشد. فضانامها مجموعه ای از اسلوبها هستند که مسمای آنان، کتابخانه شان است. `std` مسمای اسلوبهای کتابخانه `STL` است.

از علامت `>>` تعجب نکنید، این یک آپریتور بیت وایز میباشد. اما درینجا معنی ”خروج” میدهد. با زدن `ctrl + F5` خروجی نرم افزار را خواهید دید.



تابع معکوس سی-اوت، `std::cin` (سی-این) میباشد. سی-این یک متغیر را گرفته، و به از کیبورد ارزش گرفته و به آن مقداردهی میکند.

```
int main()
{
    int variable = 0;
    std::cin >> variable;
    std::cout << variable << std::endl;
}
```

کد بالا، توسط سی-این، یک عدد به متغیر میدهد و بعد آنرا به نمایش میگذارد. `std::endl` که حرف آخرش `L` کوچک است، خط جدید ایجاد میکند. همانطور که توجه کرده اید، متغیرها

Standard Template Library^{۶۵}
Namespace^{۶۶}

در C++، نوع دارند چون C++ زبان تایپ امن است. زیر چند گونه از انواع دیتا در این زبان را مشاهده میکنید.

```
//integer
unsigned int;
signed int;

unsigned short;
signed short;

unsigned long long;
signed long long;

unsigned long;
signed long;

//decimal
float;
double;
long double;

//text
char;
wchar_t;

//rest
bool;
void;
```

تمام اعداد صحیح دو حالت دارند، signed و unsigned. همانطور که بر می آید، اگر یک عدد صحیح signed باشد، میتواند اعداد منفی نیست دریافت کند. اما اگر عدد صحیح unsigned باشد، بین ۰ و مکسیمم اعداد صحیح پذیرفته شده توسط پراسسور کامپیوتر است. مکسیمم short، یک عدد بسیار کوچکتر از مکسیمم int بوده، و مکسیمم long و long

long بسیار از مکسیم int بیشترند. اما long و long، long با استفاده از حقه های اتاق نشیمن^{۶۷} مکسیم بیشتری دارند. مکسیم واقعی اعدادی که یک کامپیوتر دریافت میکند، توسط واحد پراسسور مرکزی^{۶۸} تعیین میشود. مثلاً یک پراسسور ۳۲ بیتی میتواند بین - ۲،۱۴۷،۴۸۳،۶۴۷ و ۲،۱۴۷،۴۸۳،۶۴۷ عدد صحیح نگه دارد که مساوی 2^{31} میباشد. یک پراسسور ۶۴ بیتی 2^{63} مکسیم عدد نگه میدارد. علت اینکه ۳۱ و ۶۳ است اینست که، یکی از بیتها برای علامت منفی یا مثبت عدد نگه داشته میشود. اما معنی ۳۲ بیت و ۶۴ بیت چیست؟ در بخش ۵.۳ خواهید خواند. در C++، متغیر int، ۳۲ بیتی است. اما در کتابخانه ی استاندارد، int ۶۴ بیتی نیز یافت میشود.

متغیرهایی که عدد اعشار میپذیرند، سه نوعند. فلوت، که در پایتان با آن آشنا شدیم، دابل^{۶۹} که دقت^{۷۰} آن دو برابر فلوت است، و دابل طولانی، که دقتش چندین برابر دابل معمولی است.

شاید برایتان سوال باشد که از فلوت استفاده کنید یا از دابل. جواب، در احتیاجات شماست. مثلاً اگر π را حساب کنیم، و آنرا یک بار در فلوت قرار بدهیم، یکی بار در دابل، یک بار در لانگ دابل:

```
//float:
3.1415927410125732421875
//double:
3.141592653589793115997963468544185161590576171875
//long double:
3.14159265358979323851280895940618620443274267017841339111328125
```

char و wchar_t انواع کاراکتر در C++ هستند. اولی، فقط کاراکترهای اسکی^{۷۱} دومی کاراکترهای یونیکد^{۷۲} را قبول میکند. میتوان خود کاراکتر را به این متغیر داد، یا شماره ی آن در یونیکد یا اسکی را.

^{۶۷} Tricks Parlor

^{۶۸} Central Processing Unit or CPU

^{۶۹} Double

^{۷۰} Percision

^{۷۱} ASCII – کیبورد استاندارد آمریکا

^{۷۲} Unicode – کاراکترستی که تمام کاراکترهای دنیا، حتی خط میخی پارسی، را در بر دارد.

در C++ مانند پایتان، نوع استرینگ نداریم. اما در کتابخانه ی استاندارد استرینگ داریم که `std::string` نام دارد. برای استفاده از بخش استرینگ کتابخانه ی استاندارد مانند زیر عمل میکنیم:

```
#include <iostream>
#include <string>

int main()
{
    std::string myString = "Text";
}
```

با تایپ بولی نیز آشنا هستید. تایپ ووید، یعنی تایپ خالی. کاربرد آن، کم است. هر نوع تایپ را میشود ترکیب کرد و **ثابته**^{۷۳} ساخت. ثابت‌ها، برعکس متغیرها، هرگز تغییر نمیکنند. برای ساخت ثابت‌ها از نوع فلوت به صورت زیر عمل میکنیم:

```
const float pi = 3.1415
```

متغیرهای اشاره ای و مرجعی

مسلماً شما تابحال وقتی خواستید فایلی را دانلود کنید، به سائز آن فایل نگاه کرده اید. مثلاً ۱۰ مگابایت، ۲۰ گیگابایت، و یا سائز هارددیسک اکسترنال شما، مثلاً ۱ ترابایت. هر بایت، مختص از ۸ بیت است. هر بیت، یک اینستراکشن به پراسسور است: ۰ یا یک. هر بایت، یک عدد **دودویی**^{۷۴} است و هر بیت، یک رقم آن عدد است. اعداد دودویی یا باینری، عوض ارقام ۰ تا ۹، از ارقام ۰ و ۱ تشکیل شده اند. همچنین میتواند یک بایت را به صورت **شانزده** **شانزدهی**^{۷۵} نشان داد. ارقام شانزده شانزدهی از ۰ تا ۱۶ هستند. اما ما ارقام شانزده شانزدهی را با حروف الفبا نشان میدهیم. مثلاً FF مساوی ۲۵۵ است. از اعداد شانزده شانزدهی بگذریم و به اعداد دودویی بپردازیم. یک کامپیوتر، اینگونه عمل میکند:

Constant^{۷۳}

Binary^{۷۴}

Hexadecimal^{۷۵}

۱. ابتدا، سیستم عامل، دستورات را به صورت ۰ و ۱ به رم میفرسد.
 ۲. پراسسور، بسته به ساعت^{۷۶} خود، در بازی های زمانی ثابت، این دستورات را از رم واردبایس^{۷۷} خود میکند.
 ۳. دستورات باینری وارد دروازه های منطقی^{۷۸} میشوند.
 ۴. دستورات به اطلاعات تبدیل شده، و به دستگاههای خروجی داده میشوند.
- اطلاعات در رم، با یک آدرس حافظه ای^{۷۹} هستند. آدرس حافظه، در پایه ی شانزده شانزدهی نوشته میشود. این آدرس حافظه ای در پراسسورهای اولیه فقط ۸ بیت بود، و با گذر زمان، بیشتر شد. اکثر پراسسورهای امروزی ۶۴ بیت آدرس حافظه دارند. اما بیشتر ازین هم میشود. مثلاً پراسسور Playstation 2 ۱۲۸ بیت آدرس حافظه ای دارد.
- در C++، حافظه به دو بخش تقسیم میشود: پشته^{۸۰} و هرم^{۸۱}. پشته، توسط پراسسور کنترل میشود و اگر سائز آن از حدی بیشتر شود، سرریز^{۸۲} میشود. هرم، دینامیک است و توسط کاربر کنترل میشود. متغیرها را باید دستی از پشته به هرم برد.
- و اما متغیرهای اشاره ای^{۸۳}. متغیرهای اشاره ای، متغیرهایی هستند که به آدرس حافظه ی یک متغیر دیگر اشاره دارند و اینگونه درست میشوند:

```
#include "pch.h"
#include <iostream>
```

```
int main()
{
    int i = rand();
    int *ip = &i;
```

Clock^{۷۶}
 Bus^{۷۷}
 Logic Gates^{۷۸}
 Memory Address^{۷۹}
 Stack^{۸۰}
 Heap^{۸۱}
 Overflow^{۸۲}
 Pointers^{۸۳}


```
std::cout << "'i' is: " << i << "; " <<
"The memory address of it is" << ip << std::endl <<
"And by adding * to ip we 'dereference' it like so: " << *ip;
}
```

خروجی این نرم افزار، اینست:

```
'i' is: 41; The memory address of it is 00CFFCB8 And by adding * to ip
we 'dereference' it like so: 41
```

بگذارید این کد را مرحله به مرحله توضیح دهم:

۱. ابتدا، ما، یک متغیر به نام `i` درست میکنیم و یک عدد رندوم به آن میدهیم.

۲. سپس، ما یک متغیر اشاره ای به نام `ip` درست میکنیم. برای اینکه متغیر اشاره ای درست کنیم، از علامت ستاره^{۸۴} استفاده میکنیم. سپس با علامت امپرسند^{۸۵} آدرس `i` را به آن میدهیم.

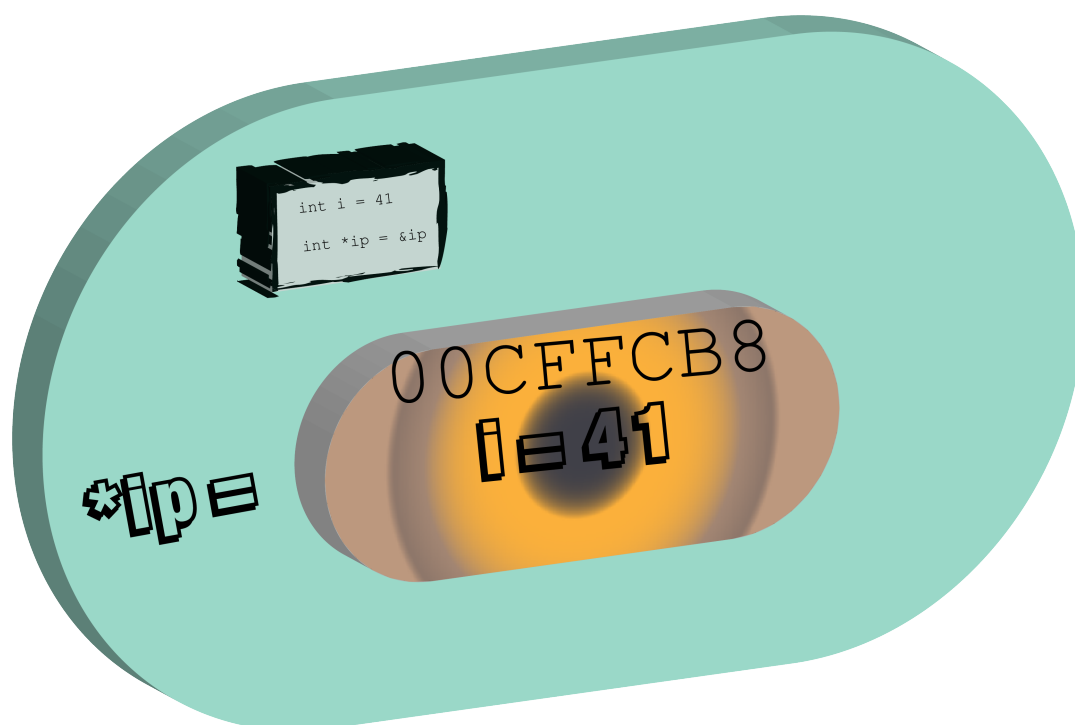
۳. سپس به کامپایلر میگوییم که اول، متغیر را پرینت کن. بعد، ارزش متغیر اشاره ای را پرینت کن، که آدرس متغیر اصلی در حافظه است. سپس، متغیر اشاره ای را **دیرفرنس**^{۸۶} کن. یعنی، ارزشی که در آدرس حافظه ای که به آن اشاره میکنی ر پرینت کن.

عکس زیر، گویای همه چیز است.

^{۸۴}Asterisk

^{۸۵}Ampersand

^{۸۶}Dereference



شکل ۱.۳: آدرس حافظه و دیرفرنس کردن

به `&i` اشاره‌ی مرجعی^{۸۷} به `i` می‌گوییم. کلاً برای اشاره مرجعی به هر متغیری، از علامت امپرسند استفاده می‌کنیم. در بخش‌های بعد، مصرف آنرا خواهید دید.

بیانیه‌های شرطی و چرخشی

بیانیه‌های شرطی و چرخشی، در C++ مانند همتایانشان در پایتان هستند. در زیر تنها به سینتکشان بسنده می‌کنیم:

```
int i = rand();
if (i > 20)
{
    std::cout << "i is greater than 20.";
}
```

Reference^{۸۷}

```

else if (i > 30)
{
    std::cout << "i is greater than 30.";
}
else
{
    std::cout << "i is smaller than 20 and 30.";
}
{
while (i > 0)
{
    std::cout << i << std::endl;
    i -= 0;
}

for (int h = i; h > 0; h--)
{
    std::cout << h << std::endl;
}

```

while و for یک کار را انجام میدهند، فقط، اینگونه است که ابتدا **بازتکرارکننده**^{۸۸} یک مقدار میگیرد، بعد شرط میگذاریم، و بعد میگوییم چقدر از مقدار کم، اضافه، ضرب یا تقسیم کن. به هرکدام ازین بیانیه هایی که در چرخش for داریم، **اصطلاح**^{۸۹} میگوییم. هر اصطلاح، یک بیانیه است، اما هر بیانیه ای، یک اصطلاح نیست.

درینجا، i در اسکوپ خارجی تابعی که کد را در آن اجرا میکنیم، قرار دارد. به اینگونه متغیرها، **جهانی**^a میگوییم. یک متغیر میتواند نسبت به کل کد، جهانی باشد. برای اینکار، کافیت متغیر را بیرون تابع اصلی تعیین، یا مقداردهی کنید. معمولاً ثابتها، نسبت به همه چیز، جهانی هستند.

Global^۱

Iterator^{۸۸}
Expression^{۸۹}

اما C++ دارای یک بیانیه ی چرخشی دیگر به نام **do...while** هستیم:

```
do
{
    i %= 10;
    i - 1;
}
while (i > 0)
```

فرق **do...while** با **while** اینست که **while** فقط در صورتی که شرط درست باشد، یک بیانیه را اجرا میکند. اما در **do...while** بیانیه ای که در برکت **do** قرار دارد، **یک بار** اجرا میشود تا شاید اگر لازم بود، شرط برقرار شود.

آرایه ها، بردارها، نقشه ها

در C++ برای نگه داشتن چندین متغیر معمولی یا اشاره ای، چندین نوع لیست محیا شده که پایه ی همه ی آنها **آرایه**^{۹۰} است. آرایه، یک گروه از دیتای **هم نوع** است. اندازه ی یک آرایه، از قبل تعیین شده است و نمیتواند کاهش یا افزایش یابد. اما آرایه، جهشی بوده و میتوان اعضای آنرا توسط ایندکس داده شده، تعیین کرد.

```
int integerArray[3];
int integerArray[] = { 1, 2, 3 };
int integerArray[3] = { 0 };

std::cout << integerArray[0];

int myArray[20];
for (int i = 0; i < 20; i++)
{
    myArray[i] = rand();
}
```

Array^{۹۰}

۱. ابتدا ما یک آرایه با ۳ عضو میسازیم. این آرایه، خالی است.

۲. سپس ما یک آرایه با ۳ عضو ساخته به و به آن اعضای ۱، ۲ و ۳ را می‌دهیم.

۳. سپس یک آرایه ی ۳ عضوی میسازیم که همه ی اعضای آن ۰ است.

۴. در مرحله ی بعد، یک آرایه ی ۲۰ عضوه ساخته، و توسط بیانیه ی چرخشی for، به هر عضو آن یک عدد تصادفی می‌دهیم. یادتان باشد که ایندکس اعضا از ۰ شروع میشود و تا سائز منهای یک ادامه دارد.

اما نمیتوان به آرایه عضوی اضافه و کم کرد. پس برای اینکار از چه استفاده کنیم؟ جواب، استفاده از یک لیست زنجیره ای^{۹۱} به نام بردار^{۹۲} میباشد. بردار، یکی از بخشهای کتابخانه ی استاندارد است. برای استفاده از بردار به صورت زیر عمل میکنیم:

```
#include "pch.h"
#include <iostream>
#include <vector>

int main()
{
    std::vector<int> myVector;

    for (int i = 0; i < 20; i++)
    {
        myVector.push_back(rand());
    }

    for (auto &i : myVector)
    {
        std::cout << i << std::endl;
    }
}
```

```
return 0;
}
```

۱. ابتدا با فرمان `#include`، فایل‌های کتابخانه را وارد کدمان می‌کنیم.

۲. سپس، یک بردار با نوع `int` می‌سازیم.

۳. بعد از آن، بردارمان را برای ۲۰ باریک عدد تصادفی واردش می‌کنیم. اسلوب `push_back()` برای اینکار است.

۴. سپس با استفاده از یک چرخش `for` برداری^{۹۳} تمام اعداد را روی صفحه پرینت می‌کنیم.

در پایتان دیدیم که اگر بخواهیم یک لیست داشته باشیم که هم‌تا به هم‌تاست، باید از دیکشنری استفاده کنیم. اما در C++ از یک نقشه‌ی آمیزشی^{۹۴} به نام نقشه^{۹۵} استفاده می‌کنیم. دو نوع نقشه داریم، ترتیبی و غیرترتیبی. ما از نقشه‌ی غیرترتیبی استفاده می‌کنیم.

```
#include "pch.h"
#include <iostream>
#include <map>

int main()
{
    std::map<int, char> myCharMap;
    char string[] = "Hello World!";

    for (int i = 0; i < 12; i++)
    {
        myCharMap[i] = string[i];
    }
}
```

Ranged^{۹۳}
 Hashmap^{۹۴}
 Map^{۹۵}

```

for (auto &i : myCharMap)
{
    std::cout << i.first << " : " << i.second << std::endl;
}

return 0;
}

```

۱. ابتدا، فایل سری نقشه را وارد میکنیم.
۲. سپس، یک نقشه میسازیم که ایندکسش `int` و ارزشش `char` باشد.
۳. سپس یک آرایه ی کاراکتری میسازیم و به آن یک متن میدهم.
۴. سپس یک چرخش به اندازی ی سائز آرایه کاراکتری میسازیم و به هر ایندکس نقشه، یک کاراکتر از آرایه را میدهم.
۵. در آخر، با استفاده از خواصهای `first` و `second` در یک `for` برداری، ایندکسها و ارزشها را پرینت میکنیم.

C++، تاپل نیز دارد. و همچنین چندین نوع دیتای دیگر. برای اطلاعات بیشتر در مورد هر چیزی از این زبان، میتوانید از اینترنت کمک بگیرید.

توابع در C++

توابع در C++ یک نوع برگشت دارند، و چندین پارامتر با انواع مختلف میپذیرند. تابع `main` یک تابع `int` است چون یک عدد صحیح باز میگرداند. توابع در دو مرحله ساخته میشوند، اعلامیه و تعیینیه. اکثر اوقات، اعلامیه در فایلهای سری انجام میپذیرد و تعیینیه در فایلهای سورس. اما اعلامیه همواره لازم نیست، و میتوان بدون اعلام کردن یک تابع، آن را تعیین کرد. تعیین کردن یک تابع اینگونه صورت میپذیرد:

```
int numberOfDigits(int num)
{
    std::vector<int> digits;

    do
    {
        digits.push_back(num % 10);
        num /= 10;
    } while (num > 0);

    return digits.size();
}
```

این تابع، تعداد ارقام یک عدد صحیح را باز میگرداند.

در تابع قبلی، از یک الگوریتم^{۹۶} استفاده کردیم. به یک سری دستور که باهم، یک کار خاص را انجام میدهند، الگوریتم میگویند. زبان C++ دارای یک کتابخانه به نام `<algorithm>` است که الگوریتمهای لازمه برای کار روی لیستها را به ما میدهد. در بخشهای بعد از الگوریتمهای گرافیک کامپیوتری، برنامه نویسی گرافیکی، و بازی سازی حرف خواهیم زد.

Algorithm^{۹۷}

اگر نوع دیتایی که تابع باز میگرداند، با نوع دیتایی که الو تابع تعیین کرده باشید، یکی نباشد، **خطای زمان کامپایل**^{۹۶} میگیرید. یکی دیگر از انواع خطا، **خطای زمان اجرا**^{۹۷} میباشد. برای جلوگیری از این نوع خطای زمان اجرا، از **Try...Catch** و **throw** استفاده میکنیم. به این منوال، **کنترل استثنائات**^{۹۸} میگویند.

```
int division(int a, int b)
{
```

Compile Time Error^{۹۶}

Error Run-Time^{۹۷}

Exception Handling^{۹۸}


```

        if (b == 0)
        {
            throw "Division by Zero!"
        }

        return a / b;
    }

int main()
{
    int a = 10;
    int b = 0;

    try
    {
        std::cout << a / b;;
    }
    catch (const std::exception& e)
    {
        std::cerr << e;
    }
}

```

سی-اریا `std::cerr` بخشی از کتابخانه ی استاندارد است که وظیفه ی آن نمایش استثنائات انداخته شده توسط برنامه است.

کلاسهای C++

کلاسها، مهمترین بخش C++ هستند. اصلا این زبان از اول برای برنامه نویسی شیء گرا درست شد. یک کلاس به صورت زیر است:

```
class Ancestor
```

```

{
public:
    Ancestor() = default;
    Ancestor(int x, int y, int z) { x = x; y = y; z = z };

    int returnX() { return x };

    int doSomething() override;

private:
    int x;
    int y;
    int z;

};

class Descendent : Ancestor
{
    int doSomething() { return 2*2 };
};

```

در بخش پایتان در مورد اسلوبها و خواصها حرف زدیم. آیا میتوانید اسلوبها و خواصهای کلاس Ancestor را پیدا کنید؟

یکی از تابعهای Ancestor با کلمه ی کلیدی override مشخص شده است. اینکار برای اینست که میخواهیم کلاسهای فرزند این کلاس، این تابع را تعیین کنند. به این کار، همانطور که در بخش پایتان گفتیم چندریختی گری یا پولی مورفیسم میگوییم.

خیلی کم اتفاق می افتد که یک اسلوب را داخل همان کلاس تعیین کنیم. معمولاً کلاسها را در فایل سری اعلام، و اسلوبهایش را در فایل سری تعیین میکنیم. در طول کتاب به مراتب این کار را انجام خواهیم داد.

پایان بخش C++

من اصلاً پیشنهاد نمیکنم که به این توضیحات کم قناعت کنید. حتماً یک کتاب بخرید و آنرا بخوانید، و یا از منابع اینترنتی استفاد کنید. هرکار میکنید، حتماً مطالعه ی زبانی خود را گسترش دهید.

۶.۳ سیستمهای کنترل نسخه

سیستمهای کنترل نسخه^{۹۹} سیستمهایی هستند که با استفاده از آنها میتوانید کدهای خود را آرگانیزه کرده و در اینترنت یا کامپیوتر خود ذخیره کنید. معروفترین سیستمهای کنترل ورژن، Git و SVN هستند.

سایت Github که بزرگترین سایت Git میباشد، میتواند یک ریپازیتوری^{۱۰۰} برای فایل های متنی شما درست کند و با سیستم کنترل ورژن، Git، فایلها را Commit و Push کند. کافیت نرم افزار دسکتاپ گیتهاب را دانلود کرده، یک ریپازیتوری بسازید، کدهای خود را در آن قرار بدهید و چند وقت یک بار، Commit Pull و Push کنید. توضیحات بیشتر را میتوانید از خود سایت بیابید. من اصلاً کار بدون یک سیستم کنترل ورژن را پیشنهاد نمیکنم. من در حالی که این کتاب را مینویسم، دارم هرزگاهی تغییراتم را گردآوری کرده، و کامیت، و پوش میکنم. ریپازیتوری این کتاب را میتوانید در لینک زیر بیابید.

<https://github.com/Chubek/Book>

۷.۳ پایان فصل برنامه نویسی

خوب، فصل برنامه نویسی هم به پایان رسید. همانطور که بارها در طول فصل گفتم، این به مثابه ی این نیست که برنامه نویسی را یاد گرفته باشید. یادگرفتن برنامه نویسی سالها وقت میبرد. اما تا وقت هست، وقت تمرین هست. تمرین کنید تا یاد بگیرید. و در حین تمرین، یاد خواهید گرفت. هرگز ناامید نشوید چون هربار که زمین بخورید، دوبار بلند خواهید شد. کدهای خود را به اشتراک گذاشته و از سوال پرسیدن نترسید.

میتوانید با استفاده از سایتهایی مثل Wars Code و Leetcode به چالشهای برنامه نویسی دست پیدا کنید تا برنامه نویسیتان قویتر شود.

فصل ۴

برنامه نویسی گرافیکی، مفهومات، الگوریتمها