کورس udemy – 100 days of code

Day001

میتونیم برای کمک گرفتن در کورس با #100daysofcode توی تویتر مطلب پیدا کنیم.



میخوایم the band name generator بسازیم.

Hello world رو رفت و داره استرینگ توضیح میده .

توی pycharm داره پیش میره ، و جالبیش اینه که توی Pycharm میتونیم اون هایلایتی که زیر چیزای اشتباه میندازه رو هاور کنیم و نکته ش رو ببینیم. فکر کنم این توی vscode نیست.

-نکته جدید: خط آخر توی اکثر زبان ها بهتره که خالی باشه. وگرنه ai که چک میکنه اشکال تشخیص میده و Char آخر رو Highlight میکنه .

-یادآوری: \n برای رفتن به خط بعدی در خروجی

Concat با +

یه نرم افزار معرفی کرد به نام thonny که میتونیم باهاش کدی که نوشتیم رو ببینیم مرحله به مرحله توسط کامپیوتر چطور اجرا میشه. این جالب میتونه باشه و در ادامه استفاده های زیادی ازش بکنیم . شاید Pycharm و vscode هم داشته باشنشون .

تا 012 Variables

ادامه

همونطور که میدونیم باید از underline استفاده کنیم به جای camelCase . (که در اصل اصن استفاده ای در پایتون از camelCase نداریم، یا snake\_case هست یا PascalCase )

یادآوری: len() برای گرفتن تعداد char ها.

-توی vscode توی اجرای کد هایی که توی هرجایی بجز c ذخیره شده به مشکل خوردیم . توی همون pycharm پیش میریم. یا عشقمون کشید با jupyter میریم . (در انتها سلطان قلب ها vscode)

تا انتهای day001

ادامه

Day002

Len() برای Int عمل نمیکنه و میگه که no length . باید conversion انجام بدیم روی Int ها وعدد ها اگر میخوام Len بگیریم . (که یعنی اگر میخوایم بفهمیم چند رقم داره باید حتما convert کنیم و بعد از len استفاده کنیم تا تعداد ارقام رو بده، که البته اگر اعشار باشه اینطوری به مشکل میخوریم و باید دقیق تر عمل کنیم. (که geminiمیگه که باید از تبدیل به int() و سپس convert کردن و سپس len() گرفتن استفاده کنیم. ))

با گذاشتن براکت جلوی هر چیزی که Length داره میتونه خروجی Index مربوطه رو داشته باشیم

Index منفی هم میتونیم استفاده کنیم و از آخر میشماره

Concatenation

نکته جالب: میتونیم بین اعداد Underline بزاریم به جای کاما تا خوندشون راحت تر بشه

Type() برای چک کردن data type

Type conversion در یسری حالتا غیر قابل انجامه.



یادآوری: اولی Float میده به صورت پیشفرض ، دومی ~~ceil~~ میکنه (درعمل ceil اتفاق نمیفته، بلکه به سمت پایین ترین عدد صحیح گرد میشه)

\*\* توان

Round() روی اعداد >> گرد میکنه بسته به مقدار اعشار به بالا و یا به پایین 5 بشه به بالا

Round(x,x) مقدار دوم میتونیم اعلام کنیم تا چند رقم اعشار گرد کن.

Fstring : print(f”string{value}”

تمرین امروز



تا وسطای 009 Day 2 Project Tip Calculator

ادامه

ما برای تمرین روز دوم از number: .2f استفاده کردیم تا اعشارش رو محدود کنیم ولی خودش از round(x, x) استفاده کرده. اولی متغیری هست که میخوایم روند شه ، دومی مقدار روند کردنشه. (البته باید دقت داشت که استفاده از number: .2f برای تغیر در اعداد اعشار باید حتما و حتما داخل fstring اتفاق بیوفته .)

Day003 treasure-island-end

تا 003 Control Flow with if else and Conditional Operators

ادامه از 003 Control Flow with if else and Conditional Operators

Modulo operator یا همون % خودمون. باقیمانده.

تمرین:



Nested if/else

تودرتو









-داخل pycharm یه سیستم تشخیص simplify داریم که میتونیم روش کلیک کنیم و همون کدی که نوشتیم رو ساده ترش رو بهمون بده .



<https://ascii.co.uk/art>

یادآوری: برای اینکه بتونیم یچیزی مثل صندوق ابتدای تمرین نهایی رو print() کنیم باید به جای “” از ‘’’ ‘’’ استفاده کنیم که 3 تا Single quotation میشه. این اجازه میده چنیدن خط رو یکجا پرینت بگیریم.

پروژه نهایی هم در انتها اینه که یه بازی ابتدا بر اساس diagram زیر بسازیم. بعدش اگر حست کشید بزرگتر و بسط داده ترش کن یا یکی دیگه بساز با یه diagram دیگه .



پایان day002

(پروژه day003 بازی سنگ کاغذ قیچیه. که بیشتر با مفهوم randomization و Python Lists قراره کار کنیم.)

ادامه

Day004

پروژه نهایی ساخت یک بازی سنگ کاغذ قیچی

Randomization

برای اینکه بتونیم از مقادیر random استفاده کنیم باید از module اضافه کنیم به برنامه :

Import random

که از.random() و .randint() استفاده میکنیم برای گرفتن یک مقدار random که البته فرق کوچیکی دارن که با یه سرچ در میاد (مثلا random خودِ صفر رو در نظر میگیره تا اون عددی که بهش دادی)

مفهوم module : میتونیم بخش هایی از برنامه رو



میتونیم ضرب هم کنیم که دو parameter رو ضرب میکنه از اونجایی که یکیشون صفره اونیکی فقط اعمال میشه عملا

یه برنامه نوشتیم واسه "شیر یا خط" که یه سکه پرت میکنیم.

Python list

یه data structure عه .



یادآوری: با Index میتونیم یک item رو return کنیم. با مقدار منفی از انتها شمرده میشه.

aList.add() و aList.extend() و...

میتونیم با random.choice() از item های یک List به صورت random returnداشته باشیم.

Nested list : میتونیم داخل یک List ، List تعریف کنیم (داخل index ها)

تا 007 day 004 03:17

ادامه

برای اینکه بتونیم برنامه ای که نوشتیم رو کاری کنیم دائم در حال اجرا باشه و در صورت اینکه کاربر قصد پایان داشت خاتمه پیدا کنه باید از ساختار کلی زیر در پایتون استفاده کنیم (که روی پروژه روز 4 سنگ کاغذ قیچی پیاده ش کردیم) :



حالا باید بریم ببینیم روش خودش چیه

1-ascii art هارو داخل یک List ذخیره کرد که با مقادیر وارد شده توسط user مطابقت داره که باعث میشه فقط بنویسه List\_name [user\_choice] که برای پرینت کردن شکل ها خیلی راحت تره. (کاری که ما کردیم این بود که یه module براش ساختیم و هر دفعه برای پرینت کردن هرکدوم باید این رو مینوشتیم: module\_name.scissors و یدور هم نیاز به Import module\_name داشت. که خب کاری که خودش کرده راحت تره)

ادامه

جلسه قبل در طی ور رفتن با گیت فایل هایی که نوشته بودیم پرید ، ولی از دوباره یه repository ساختیم که اونجا Push کنیم و نیاز نیست از دوباره کد هارو بزنیم مربوط به ابتدا رو ، فقط از اینجا به بعد رو انجام میدیم .

Day005

پروژه امروز PyPassword generator هست .



جالب: یه سایت معرفی کرد برای اینکه چک کنیم email رو و اگر شرکتی مورد هک واقع شده و اطلاعات شما هم اونجا بوده و رفته بهتون بگه. ([link](https://haveibeenpwned.com/))

خب ما دوتا راه داریم این کارو انجام بدیم: easy version و hard version

ابتدا میریم سراغ easy .

در easy به ترتیبِ حروف ، سمبل، عدد رفتیم ولی در hard باید بدون ترتیب مشخص پسوورد ساخته میشد. که ما از خودمون با random درستش کردیم. ولی داخلِ خودِ random یه method هست به نام shuffle() که همین کار رو انجام میده .

Random.shuffle(List\_name) یک Object رو به صورت درهم خروجی میده.

در hard version میتونستیم از روش های مختلفی بریم. از اونجایی که easy version رو رفته بودیم پس یعنی تعداد در اون درسته ، فقط میمونه ترتیب رو به هم بریزیم که با random.shuffle انجام میشه . چیزی که ابتدا تو ذهنم بود این بود که برای هر جایگاه به صورت تصادفی از یکی از لیست ها انتخاب بشه و بعد به صورت اتفاقی از داخل همون لیست انتخاب بشه و این بره تا جایی که ورودی های کاربر تشکیل بشه. یعنی وقتی تعداد Number هایی که کاربر میخواست touch شد دیگه اون لیست رو داخل امکانات نزاره.

ادامه

قبل از اینکه ادامه بدیم برای اینکه این چیزایی که بلدیم رو سریع بزنیم بره بعدی، چطوره همون ابتدا سعی کنیم پروژه پایانی رو بسازیم. اگر تونستیم میتونیم اون روز رو Skip کنیم.

Day006

با یه تمرین باحال روبروییم اینجا:

<https://reeborg.ca/reeborg.html?lang=en&mode=python&menu=worlds%2Fmenus%2Freeborg_intro_en.json&name=Hurdle%201&url=worlds%2Ftutorial_en%2Fhurdle1.json>

که باید با کد های پایتون این رباته رو ببریم جایی که پرچم هست.



تا مرحله hurdle4 میره و بعدش میره سراغ maze که همه رو حل کردیم کداش هست. (برای مرور خوبه یه دور چک کنی)

3 تا مسئله هم برای debug داره که اونا هم حل کردیم ولی یه راه خودش رفت که برای سه تاشون جواب بود و اون رو نمینویسم اینجا توی کد ها هم نیست. سعی کن بنویسیش توی مرور.

(



این قطعه کد رو میزاره قبل از تابع اصلی تا قبل از اینکه شروع به هرکاری کنه، مطمئن بشه دیوار رو پیدا میکنه چون الگوریتم بر این اساسه که دیوار سمت راستشو بگیره و بره. لازم به ذکره داریم فقط و فقط راجب maze صحبت میکنیم.

)

ادامه

Day007

جلسه قبلی project نداشت و فقط درس بود. اما این جلسه Hangman رو داریم.

طبق چیزی که جلسه قبلی گفتیم سعی میکنیم خودمون بزنیمش اگر نتونستیم میریم با خودش پیش میریم.

خب چیزی که میبینم اینه که به تنهایی از این یکی بر نمیایم. با خودش پیش میریم. (روی این با خودش پیش میریم خیلی تاکید میکنم، چیزی رو skip نکن اگر مردی سرعت ببخش بهش. توی این تمرین ها قراره مسیر رو یادمون بده و راه و روش رو. اگر یه بخشی رو بلدی دلیل نمیشه بقیه رو هم بلد باشی. )

جالب اینه که Flow chart کشیدن رو یه مرحله از برنامه نویسی میدونه و باید قبل از کد زدن انجامش داد. با ابزار draw.io آنلاین میتونیم این کار رو بکنیم.

توضیح بازی:

اینطوریه که یه کلمه باید حدس زده بشه. تعداد حدس ها هم یه تعداد محدودیه (هربار که اشتباه حدس زده بشه یه قسمت از شکل hangman (انسانی که به دار آویزونه) رو میکشه و هروقت شکل کامل بشه یعنی ما باختیم و اگر قبل از اینکه جونامون تموم بشه کلمه رو حدس بزنیم ، بردیم .

حالا باید با این توضیحات و چیزی که دیدیم یه flow chart بکشیم که این پروژه چطور کار میکنه Logically

Enumerate() میتونیم str بفرستیم داخلش و به عنوان خروجی بهمون index و char رو میده میتونیم استفاده کنیم . (که با استفاده از اون داخل یک for میتونیم iterate کنیم و برای هرکدوم داشته باشیمش. )

با ذهن خسته نمیشه کار کرد ، از یه مسیر خیلی ساده ای میشد مشکل حل بشه که داخل ویدیو خودش رفت و ما مدتی روش فکر کردیم ، نکته اصلی اینه که نباید سعی کنیم جلو جلو بریم ، وقتی داریم step step میریم باید باهاش پیش بریم و کاری که خواسته رو بکنیم.

باقیش برای فردا

تا انتهای 004 از day007

ادامه

-نکته: میتونیم برای اینکه در مراحل مختلف برنامه cls بگیریم داخل کد از :



استفاده کنیم. البته باید import subprocess هم بکنیم.

پیشنهاد: میتونیم برای تمرین یا مرور سعی کنیم از ابتدا project های هر روزی که هست میخوایم مرورش کنیم رو بزنیم از ابتدا. دفعه بعدی این کار رو بکن ببین efficient هست از لحاظ زمانی یا نه

تا انتهای day007

ادامه

باید بدو بدو بریم تا جلسه 15. در حداقلی ترین و سریع ترین حالت ممکن پیش میریم. بجز نکات مهم چیزی نمینویسیم اینجا .

Day008

Encode decode کردن با shift کردن به تعداد مشخص (Caesar cypher )

یادآوری: میشه در تعریف function به جای Positional arg از حالت parameter=”value” استفاده کرد که دیگه ترتیب مهم نباشه. ولی خب باید Parameter هارو بدونیم.



(عکس بالا برای شیف کردن استفاده شد که به جای اینکه چندین سری از alphabet رو اد کنیم توی List که جا داشته باشه و ادامه بده به میزان اون عدد ، اینکار باعث میشه user هر عددی خواست وارد کنه و معادلش در حالت شیف شده بدست بیاد. )

تا day008 007 12:31

ادامه

برای نوشتن Caesar() یه ترفند جالبی زد که با توجه به فرایند فقط نیاز بود shif \*= -1 بشه تا decrypt انجام بشه. همین. لازم نبود مثل چیزی که خودمون نوشتیم یه func جدا تعریف کنیم.

پایان day008

Day009

Dictionaries and nesting

Silent auction program

یادآوری dictionary\_name = {“key” : “value”, “key2”: “value”, … }

برای return کردن: dictionary\_name[“key”]

برای مقدار اضافه کردن : dictionary\_name[“new\_key\_name”]= “value”

برای تغیر یک key : dictionary\_name[“key\_for\_change”] = “changed value”

-در iteration ها key return میشه

-برای بیشترین مقدار max(dictionary\_name)

تا ابتدای day010

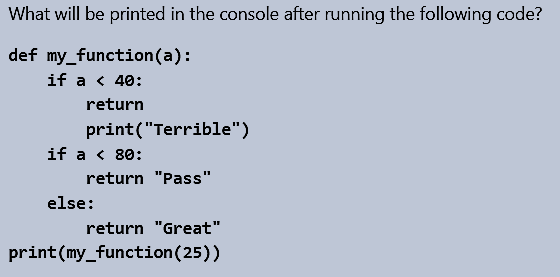
ادامه

‘string’.title() کاری میکنه string مون هرجوری از لحاظ بزرگی و کوچیکی بود اولین char بزرگ بشه بقیش کوچیک

-یادآوری: وقتی return انجام میشه func تموم میشه.

-یادآوری: برای function های داخلی python ما docstring داریم که وقتی روش hover میکنیم اون رو نشون میده و بهمون میگه که این function در واقع چکار میکنه چیا میگیره با چه data type ای و چی پس میده و ... ، ما برای Function هایی که خودمون میسازیم هم میتونیم این رو ایجاد کنیم با تنها اضافه کردن یک string به خط اول function مون. که اگر تعداد خطوط زیاده میتونیم از ‘’’’’’’ استفاده میکنیم. بعد وقتی Hover میکنیم اون چیزی که تو توضیحات نوشتیم نشون داده میشه. توی تمرینات exercism هم داشتیم میتونیم بریم اونجا نکات بیشتری یادمه داریم./

-نکته: در ازای این کد ها None میگیریم:



چرا؟ چون 25 که وارد میشه اولین return که جلوش هیچی نیست و میشه None اجرا میشه چراکه شرطش برقراره و سراغ باقی کد ها نمیره.

یادآوری: تغیر نام function چه داخلی چه چیزی که خودمون نوشتیم فقط باید یک variable بنویسیم و اسم اون function رو بدون پرانتز (چون پرانتز بزاریم call میشه) بزاریم اون داخل. از این به بعد با اسمی که برای اون variable گذاشتیم هم میتونیم function call کنیم که میشه توی ساختاری مثل این ازش استفاده کرد:



(فقط باید مواظب باشیم که نباید در حین این کار Function call انجام بشه ، یعنی نباید پرانتز بزاریم وقتی میخوایم تغیر نام بدیم . بدون پرانتز باید استفاده کنیم .)

تا day010-007-12:13

ادامه

Day011

Black jack game

این بازی رو قبلا ساختیم . ولی سراغ چیزایی که قبلا نوشتیم نمیریم و از ابتدا باید بسازیم

Flow chart رو ساختیم

تا day011 002 end

جلسه بعد از روی flow chart کد بزن

ادامه

الان میخوایم از روی flow chart خودمون کدش رو بزنیم

نکته: 

این رو AI داده برای ascii art هر ورق، شاید بعدا ادش کردم توش:



-برای اینکه جمع همۀ عناصر یک list رو داشته باشیم sum()

ادامه



ما به ترتیب به هردو کارت میدادیم ولی اینجا توی دوره اول به user کارت میده بعد شروع میکنه به pc کارت میده تا زمانی که pc < 17 باشه کارت دادن ادامه داره. البته واسه زمانیه که ما گفتیم دیگه کارت نمیخوایم به عنوان user. یعنی کارت دادن به pc بعد از user اتفاق میوفته.

ادامه

Day012

Number guessing project

درمورد Scope و اهمیتش گفت.

نکته یادآوری مهم : برای اینکه از داخل یک function بخوایم یک global var رو Modify کنیم باید یبار داخل Scope ای که برای function داریم ذکر کنیم که یک متغیری هست به این نام و global هست و ما میخوایم اون رو تغیر بدیم. که میشه global variable\_name و این باید داخل scope باشه.

-البته میگه که اینکارو نکنیم بهتره، چون باعث باگ میشه و دیباگ کردنش سخته . اگر خواستیم Modify انجام بدیم روی یچیز global با Function بهتره از return استفاده کنیم و یجوری return کنیم که وقتی برابر با اون متغیر قرارش میدیم اون Functionality ای رو داشته باشه که میخوایم . ولی درمورد const variable فرق میکنه .

یادآوری: Const چیزیه که define میکنیم و نمیخوایم که تغیرش بدیم . در این صورت میتونیم global رو اعلام کنیم ولی باید این رو یادمون باشه که تغیر دادنش ممکنه، یعنی مثل باقی زبان ها نیست که نشه تغیرش داد، عملا همون متغیر عادیه. ولی با حروف بزرگ نوشته میشه که یادِ خودمون بمونه که این const عه . (و اینطوری هم هست که const میسازی و داخل function ها اگر خواستی استفاده کنی global میزاری پشتش و تمام. هر بلایی هم داخل سرش بیاری هرجایی برای ابتدا نوشتی ایجادش کردی اونجا هم در نظر گرفته میشه. )

-علاوه بر اون سایت که برای ascii art معرفی کرد که حاضر آماده بود ، این سایته هم هست:

<https://patorjk.com/software/taag/#p=display&f=Ghost&t=Type%20Something%20>

که میشه بری هرچیزی که مینویسی رو برات ascii art کنه بهت بده

انتهای day012

ادامه

Day013

درمورد روش های debug داره صحبت میکنه. چطور مشکل رو حل کنیم.

1.اگر مشکل دارید، اول باید سعی کنیم مشکل رو توضیح بدیم برای خودمون و برای خودمون حداقل روشن کنیم که قضیه از چه قراره. جمله ای که گفت : (تقریبا غیر ممکنه وقتی که خودتون از مشکل خبر ندارید بتونید حلش کنید)

2. چیزی که نوشتید ببینید دقیقا چیکار میکنه، چیکار میخواستید بکنه باید چه چیزی رو تغیر بدید

نکته مهمی هم که وجود داره اینه که مشکلی رو میشه حل کرد که بشه از دوباره خلقش کرد. به تعبیری بدونی در ازای چی ارور میگیری یا مشکل داری دراون صورت میتونی حلش کنی. با تغیر دادن تصادفی و بدون حساب کتاب گاها شاید مشکل حل بشه ولی توی روند فرسایشی میوفتیم .

یادآوری ساختار try except ، وقتی که یه ارور توی یک بخش مشخص از کد پیش میاد و میدونیم ارور چیه (مثلا ValueError) اون رو بگیریم و تبدیل کنیم به یه پیام مشخص تر یا یه عملیات کاربردی در طی پیش اومدن اون مشکل: (این نکته هم قابل توجهه که یک function داخلی هم از همین قضیه برای هندل کردن ارور های مختلف استفاده میکنه این رو توی step into فهمیدیم)



و این هم هست که در طی این try except کد در محل error بلاک نمیشه و روند تا انتها ادامه پیدا میکنه .

نحوه کار با debugger هم اینطوریه که شما ران میکنی و این خط به خط اجرای کد رو بهت نشون میده. یه step over داریم و step in و توی Pycharm یدونه داریم که داخل Module های Inner نمیره و فقط module های خودمون رو برسی میکنه . اینطوری میتوینم خط به خط ببینیم چی میشه و اگر یسری خطوط رو میخوایم بزنیم بره جلو از breakpoint همون نقطه قرمزه که بقل عدد های خطوط کد میشه اضافه ش کرد قرار میشه داد. از اونجایی که breakpoint هست Debug شروع میشه .

(پس این رو میتونیم به عنوان یه نکته از این به بعد داشته باشیم که 1.همیشه کامنت کن کاری که قراره توی اون بخش کد انجام بدی و قراره بنویسیش 2.اگر به مشکل خوردی از debugger استفاده کن تا مشکل رو پیدا کنی. )

اهمیت Debugger از اهمیت print() میاد، که وقتی خروجی میگیری از چیزی که نوشتی بهتر میتونی بفهمی چه اتفاقی داره میوفته. حالا debugger در عمل داره برای همه چی Print() استفاده میکنه و خط به خط نشونت میده چه اتفاقی داره میوفته. با استفاده کردن مکرر از debugger که در IDE های مختلف ساختارش یکیه، میتونیم مهارتمون رو هی بهتر و بهتر کنیم. سر کلاس های c++ , java بهش میگفتیم تحلیلِ برنامه ، که خط به خط میرفتیم و برسی میکردیم چه اتفاقی داره میوفته. جالبه

نکته بعد هم درمورد رفع مشکل و ارور، اینه که اگر هرکاری میکنی نمیشه ، شاید اون روز روزش نیست و باید استراحت کنی و یه مقدار زمان بدی بهش و بعد بیای سراغش. ما این رو میدونیم که بعد از گذروندن زمان و چرت زدن و خوابیدن و ... مغز ما این مهلت رو پیدا میکنه که بین اطلاعات و بخش های مختلف ارتباط برقرار کنه و اینطوریه که ما چیز هارو میفهمیم و به خاطر میسپاریم، این که کد زدنه درمورد ورزش و فعالیت های فیزیکی هم قضیه همینه، پس زمان دادن به مغز برای رفع مشکل خیلی مهمه.

در مرحله بعدی هم اگر نتونستی باید از یه ذهنِ انسانی دیگه بپرسی

نکته بعدی هم این که بعد از اینکه بخشی اضافه شد به کد و نوشتیم و پیاده ش کردیم ، بهترین کار همینه که همون لحظه یه اجرا ازش بگیریم و ببینیم چی پیش میاد.

باگ ها پیش میان، نکته اصلی اینه که حلش کنی و مهمتر اینکه راه های مختلف برای حل کردنش رو بلد باشی، چه مستقیم چه غیر مستقیم .

انتهای Day013

Day014

Higher lower game

این یه بازیه که باید خودمون از صفر تا صد بسازیم.

تا الان flow chart رو کشیدیم ، کدش هم زدیم ، مونده debug کردنش، مشکلش هم نوشتم توی calendar در مراحل پایانی مشکل داریم ، دونه دونه هم مرحله هایی که درست انجام دادیم رو commit کردیم . از این جهت مشکلی نداریم.

جلسه بعدی اول باید کاری کنیم این بازی درست کار کنه که کار زیادی نمیبره و بعدش ویدیو هاشو میبینیم.

ادامه

الان قراره که debug کنیم. مشکل چیست؟

مشکل هایی که حل کردیم اولیش این بود که debug کردیم مشکلِ حذف شدن score درست بود. یعنی نمیرفت بشینه جای A و در عمل از ابتدا تشکیل میشد. و یسری قابلیت های کلی به برنامه اضافه کردیم و عملا نکته خاصی نبود و در انتها بازی باحالی شد (البته اگر خودمون بریم یسری data درست و درمون بزاریم توی database ش . )

Day015

Coffee machine project

001 رو دیدیم توضیحات اولیه بود برای برنامه coffee machine .

جلسه بعدی pdf رو چک میکنیم، flowchart میکشیم و کد میزنیم.

ادامه

یه وقفه طولانی مدت داشتیم سر مسائل درسی و کاری و ...

قرار بر این بوده که بریم یه سر سراغ Pdf و خودمون Flowchart بکشیم و ادامه ماجرا ولی چون فاصله افتاده قصد دارم قسمت اول رو دوباره ببینم.

این اطلاعات یه Coffee machine ساده س:



این اطلاعات 3 حالت مختلفیه که باید تولید کنه:



منابعی که در ابتدا در اختیار داره:



ارزش سکه ها:



طبق اطلاعات بالا و چیزایی که اد کردیم تو کدش همین الان باید جلسه بعدی پیش بریم و سعی کنیم که کدی بنویسیم که وقتی با PDF این قسمت چکش میکنیم درست کار کنه. ادامه ش جلسه بعد.

ادامه

درحال تکمیل پروژه coffee machine هستیم ، خیلی طولانی شده ، ولی بالاخره انجاامش میدیم.

خب در حال حاضر coffee machine رو انجام دادیم تا حدی که میخواست داره کار میکنه، البته خودم شاید بعدا یسری چیزا بهش اضافه کنم.

در ادامه باید با Pdf ای که گذاشته برنامه رو چک کنیم و چیزایی که میخواد رو اضافه کنیم و چیزایی که نداره رو اصلاح کنیم. که باشه برای جلسه بعد.

ادامه

باید مرحله به مرحله گزینه های pdf این قسمت (برای یادآوری الان day015 ایم. ) رو چک کنیم و باید طوری باشه که برنامه طبق اون چیزی که توی هر مرحله گفته درست عمل کنه .

Off رو اضافه کردیم.

درحال اضافه کردن کد های درست برای پیام out of resource بودیم که نت رفت. باقیش بعدا.

ادامه

درحال درست کردن این هستیم که وقتی یچیزی از منابع کمه اون رو توی پیام اطلاع بده، نه اینکه مثل برنامه خودمون سر راست بگه که منابع کافی نیست، یعنی بنویسه water کافی نیست. در طی این میخواستیم که عناصر یک list رو کنار هم پرینت کنیم که باید از

استفاده کنیم تا بدون داشتن [()] پرینت بشن . (کاری که این میکنه اینه که عناصر رو به هم دیگه متصل میکنه ، چیزی که در ابتدا به عنوان string گذاشتیم رو بینشون قرار میده و همه رو در قالب یک string ریترن میکنه. )

نکته: یه راه حل ai گفت متاسفانه نتونستیم انجامش بدیم. مثل اینکه با گذاشتن یک \* پشت اسمِ list میتونیم تمامی عناصرش رو پرینت بگیریم. ولی خب چرا توی اون فرمتی که ما میخواستیم نشد رو نمیدونم (الان تستش کردم ، انجام میده ولی با f-string مشکل داره، دلیلشم اینه که میشه String رو با string کانکت کرد ولی نمیشه با یه list این کار رو انجام داد از همین جهته که Python مشکل داره باهاش. باید با همون join ابتدا خروجی str بگیریم و بعد استفاده کنیم. )



(این هم در نوع خودش چیز جالبیه، الان به کارمون نمیاد ولی در آینده شاید یکی از راه حل های ساخت class باشه برامون. )

-نکته: یه چیز جالب یاد گرفتیم. توی vscode و البته pycharm میتونیم با نگه داشتن shift + alt و تکون دادن موس از یه ستون مشخص انتخاب کنیم و همشون رو یکجا تغیر بدیم. جالب بود.

-الان داریم برسی میکنیم که ببینیم اون چطوری نوشته و ما چطوری نوشتیم و کد زدن و کارای خودمون تموم شده، همش هم با Pdf چک کردیم درست بوده .

یه فرقی که داره اینه که من برای پرینت کردن report حلقه نوشتم و for و همه چی رو با منطق چفت کردم که هر resource دیگه ای هم اضافه کنی بهش درست کار میکنه، ولی خودش فقط از Print() کردنشون استفاده کرده. که خب راحت تر و ساده تره ولی کاربردی نیست. بگذریم . (البته الان که دارم فکر میکنم همچین کار شاقی هم نکردیم، چراکه فقط برای water, coffee, milk, money درست عمل میکنه، خب در این حالت روش دوره راحت تر و درست تره)

فرق دیگه اینه که ما تلاش کردیم کد برسی کنه کدوم resource کمه و اون رو دقیق توی همون خط پرینت کنه که توی خودِ دوره خط به خط پرینت میکنه مثل اینکه. چراکه توی loop گذاشتش. بگذریم.

فرق دیگه: برای Process coins یدونه function قرار داد. ما توی خودِ کد انجامش دادیم. هرچند منتقل کردنش راحته. (تا به اینجا لزومی نمیبینم بخوایم function ش کنیم. البته در OOP و دیدگاهش باید این کار رو بکنیم تا بعدا هم بتونیم استفاده کنیم ولی خب)

فرق دیگه: برای اینکه in\_change رو نشون بده از round(,) استفاده کرده که ما از در نظر نگرفتن یسری از واحد های اعشار در حالت float استفاده کردیم :

که خط کد بالا باعث میشه فقط دو عدد اعشارش نمایش داده بشه و وقتی هم که عدد اعشار نداره صفر میزاره به جاش (که در این حالت روند میشه به سمت پایین و عملا floor() انجام میشه. )

Day016 – OOP programming

الان میخوایم بریم سراغ oop .

به نوعی از برنامه نویسی که تا اینجای کار انجام دادیم میگن procedural programming ، که فکر میکنم فارسیش میشه ساخت یافته. که میشه ترکیبی از Process function که وسطاش call میشه و ... و باعث میشه در برنامه های complex شبیه کلاف تو در تو بشه و به یادآوردن ارتباط function ها و کلا همه چی سخت بشه. چیزی نیست که امروزه روز استفاده بشه.

برای یادآوری باید بگم که مفهوم OOP برای این به کار میره که قابلیت scale خیلی زیادی داره، که در نوع قبلی این وجود نداره به اون صورت. به صورت بخش بخش و Object object وقتی به یک مسئله بزرگ نگاه کنی میتونی بهتر عمل کنی.

یادآوری: attributes همون variable هست که مربوط به یه object بخصوصه . method همون function عه که مخصوصه یه object خاصه.

یادآوری: class به حالت کلی برای Object های مختلف میگن. در عمل object ها از روی class و با پر شدن اونا به طرق مختلف بسته به نوع نیاز از اون class ساخته میشن. که Object های ساخته شده از یک class دارای attribute و method های یکسان هستن ولی با data متفاوت

مثل اینکه اسمِ class هارو در پایتون به صورت camelCase نام گذاری میکنیم. برای اینکه یادمون باشه این class عه . (این درسته کاملا با ai چک کردم، همچنین کامل با حروف بزرگ نوشتن برای const ها و همون حالت عادی برای variable ها و function های عادیه. camelCase هم برای class ها. )

یه چیز جالب و هیجان انگیز اینه که ما قراره از اینجا به بعد حالت های گرافیکی داشته باشیم و از کدِ خالص در بیایم. با استفاده از یه library به نام turtle graphic که گفته شد به صورت پیش فرض روی همه Python هایی که نصب میکنیم هست.

برای ساخت Object، میریزیمش توی یک متغیر دلخواه . (درست ترش اینه که MyClassName() رو میریزیم داخل یک متغیر و داخل پرانتزش بهش مقادیر لازم برای اون Object بخصوص رو میدیم. )

وقتی پرانتز داریم یعنی method داریم.

این یه window باز میکنه که توش یه فلش هست که میشه بهش شکل داد رنگ داد و کلا همه بلایی سرش اوورد و طوری که توی خود vscode نوشته بود یچیزیه که برای آموزش کد نویسی به بچه ها درست شده ، ولی خب جالبه دیگه

میتونیم از آدرس : docs.python.org/3/library/turtle.html

بیشتر وارد جزئیات این library بشیم ولی فعلا همین چیزاییه که گفتم . اینم قبلتر گفت ولی من فکر کردم داره برای ساختنش میگه نگو کلا اینطوریه، گفت فکر کنید یه لاکپشته که میشه شکلشو تغیر داد به کلی و یه قلمو بسته شده به پشتش و توی صفحه بالا میاد و میتونیم بهش دستور بدیم و چیز میزاشو تغیر بدیم و این حرکت کنه و در انتها یچیزی روی اون صفحه ای که بالا اومده بکشه.

باقیش جلسه بعد. (فکر میکنم باید تلاش میکردیم رنگ قرمز بدیم به لاکپشته.)

ادامه

نکته: درمورد module, package, library : یک فایل از کد ها و function ها و variable ها میشه یک module. که میشه در باقی برنامه ها Import کرد. یک package میشه یک پوشه یا بهتره بگیم یه مجموعه ای از Module های به هم مربوط برای مصرفی خاص. که معمولا یک فایل با نام:

داره که به خودِ python میفهمونه این باید به عنوان یک Package شناخته بشه. و در انتها Library هم میشه مجموعه ای package ها و Module های به هم مربوط برای یک مصرف خاص که از قبل آماده شده میشه import کرد و استفاده کرد. به نحوی باید گفت که کدیه که زده شده دیگه لزومی نداره باقی افراد هم بزنن . نکته دیگه ای هم که هست اینه که مثلا این رابطه در همه زبان ها وجود داره ولی شاید اسم های مختلفی بهشون اطلاق کنن مثلا در ruby به package ها میگن gems . جالبه.

-نکته بعدی اینکه داره درمورد یه موضوعی صحبت میکنه که بتونه باهاش استفاده از یک Package رو توضیح بده . مثلا پرینت کردن یک table به شکل ascii که اگر بخوایم خودمون این کار رو انجام بدیم خیلی نفس گیر خواهد بود. رسید به اینکه یه library یا package معرفی کرد به نام pypi که در اون علاوه بر اینکه میشه package های مختلف با کاربرد های مختلف پیدا کرد و document شو خوند و استفاده کرد به رایگان، یدونه برای این کار وجود داره به اسم PrettyTable . چون داره از روی pycharm پیش میره روش pycharm ش رو گفت که میری از تو تنظیمات و سرچش میکنی و Install رو میزنی و ... ولی ما که روی vscode ایم و با Pip اشنایی داریم تست کردیم و بله ... با کد زیر نصب کردیم آخرین حالت رو:

و تمام. پس روش نصب کردن یک package اینه که اسمشو بدونیم و pip install packagename

-(درمورد Pypi باید بگیم که این یه Library محسوب میشه که عمومیه و همه داخلش کدهاشونو اضافه میکنن و همه هم میتونن بردارن و استفاده کنن، و مستقیم مثل اینکه وصله به Pip ، این رو نمیدونم که Pip به چندتا Library دیگه وصله.)

خب الان تا انتهای قسمت 7 از روز 16 پیش رفتیم، و الان قراره با فایل های آماده ای که خودِ امروز داره و استفاده از oop ، همون پروژه دیروزی رو بسازیم. بریم

تا ابتدای 008

ادامه

کداشو به اصطلاح refactor کردیم، همونطوری کار میکنه (که در روش procedural programing کار میکرد) فقط با کدهای خودِ دوره انجام شده .

نکته خاصی نداره به اون صورت، بیشتر نکته همون استفاده از کدها و package های ساخته شده توسط دیگری با توجه به document (این کدهایی که نوشته بود یه فایل pdf به عنوان document داشت که از روش بتونی بسازی) ای هست که درموردش در اختیار داریم. یعنی اینطوریه که مثل یه ماشین لزومی نداره بدونیم چه اتفاقی اون داخل میوفته، فقط لازمه ورودی و نوعش رو بدونیم و بدونیم در انتها چه خروجی ای در ازای ورودی و با چه نوعی بهمون میده (نحوه کار کردن با اون رو باید یاد بگیریم.)

پایان روز 16

Day017

Creating classes

Project : quiz game

قراره بریم سراغ ساخت class های خودمون. در day قبلی داشتیم استفاده کردن از class های آماده رو یاد میگرفتیم که حالا یا built-in بودن یا با package بودن و Install شون کردیم. الان قراره خودمون بسازیم. هرچند بلدیم ولی خب باید ببینیم و یادآوری بشه.

-برای اینکه یه function یا class رو خالی بزاریم و همچنان بتونیم ادامه بدیم و بعدا پرش کنیم روش درست گذاشتن pass هست. خودمون یه استرینگ خالی میزاشتیم.

-یه چیزی قبلتر درمورد نوع نام گذاری گفتیم که تنها یچیزش اشتباه بود، اونم اینه که برای class ها از camelCase استفاده نمیکنن از PascalCase استفاده میکنن. که اولش بزرگه حتما . گفت از camelCase اصلا در پایتون استفاده نمیشه. (که خب این خودش جالبه میتونه برای متغیر های خاصی که فقط خودمون باید بدونیم استفاده بشه. بگذریم)

یادآوری: همونطور که گفتیم، یک class دارای attribute و method() هست. یه راه اضافه کردن attribute اینه که زارت بهش اشاره کنیم و مقدار بدیم، اون attribute ساخته میشه. فکر میکنم برای method هم همین باشه. ولی یه راه دیگش constructor عه.

یادآوری: برای ساخت یک class :

و داخلش هرچی بنویسیم جزو اون کلاس میشه.

برای ساخت constructor :



که این تابع همون constructor رو میسازه. دوتا underscore ش مهمه.

نکته ش هم اینه که هربار که class صدا زده بشه این تابع کال میشه.

Syntax برای ساختن attribute در class \_\_init\_\_() :



اینطوریه که self میشه همون object که داره ساخته میشه با این init ، خب اولیش که self عه هیچی . هرچی parameter اضافه کنیم در داخل function متصلش میکنیم به همین object درحال ساخت به صورتی که نوشته.

نکته: وقتی یه class کال میشه حتما اولش constructor ش کال میشه و بسته به اینکه چندتا arg میخواد باید حتما بهش ارسال بشه، اگر نشه ارور میگیریم و object ساخته نمیشه، پس میتونیم این طوری در نظر بگیریم که وقتی constructor مینویسیم داریم یچیز دائمی مینویسیم که باید حتما اون Object برای ساخته شدن داشته باشه و یجورایی اون اطلاعات پیش نیازش هستن. و زمانی که نمیخوایم حتما اون اطلاعات رو در ابتدا به صورت arg ارسال کنیم ، ولی نیازه که باشه از default value استفاده میکنیم و دیگه توی اون فرمت به عنوان parameter اضافه نمیکنیم، فقط self.myValue = defaultValue .

##عاها، یه سوالی هم که اینجا مطرحه اینه که فرض بگیریم یه class ساختیم و ازش استفاده کردیم و چندین object ازش ساختیم و داریم استفاده میکنیم ، بعد میخوایم به class مون یسری attribute و البته function اضافه کنیم، اونموقع چطور میشه؟ (در نگاه اول که میشه ، قطعا روش های خاصی هم وجود باید داشته باشه که این کارو راحت کنه، ولی راه حلی که خودم الان به ذهنم میرسه اینه که یه Loop بنویسیم که اون مقادیر رو برای Object های قبلی پیش فرض یه مقداری قرار بده یا با فرایندی پرشون کنه .)

برای ساختِ method() هم basic یکسانه، از اونجایی که ما یه self داریم باید همه چی رو به اون attach کنیم ، حالا یا با arg یا بصورت default. درکل باید با self کار کنیم.

برای مثال از ساخت و استفاده از class, object, method, attribute در عکس پایین فرایند فالو و فالوینگ اینستا رو تا حدی شبیه سازی کرده، خیلی جالبه:



که خب در basic ای که برای یک class مینویسیم خیلی مهمه که اینا دقیق نوشته بشه چون همه Object ها ازش تبعیت میکنن.

حالا که این هارو که فهمیدیم باید بریم سراغ پروژه امروز .

تا ابتدای day017 006

ادامه

-یه نکته ای که درمورد list وجود داره اینه که میشه بدون ساخت یک variable یک object رو بهش فرستاد، چراکه هر Index خودش یک variable عه . درصورتی که کاری که خودش کرد این بود که اول ریختش توی متغیر.

-todo های این پروژه بعد از اینکه data set رو درست کردیم:



نکته: چیزی که مهمه اینه که باید برای مرور یدور دیگه سعی کنیم این پروژه رو از صفر خودمون بزنیم. یا اصن یه پروژه دیگه مثل این رو.

ta Day017 009 03:35

ادامه

-یه سوال دیگه ای هم که هست اینه که کدوم متغیر رو self در نظر بگیریم (یعنی بچسبونیمش به object و بتونیم بهش ارجاع بدیم و return بگیریم، و کدوم رو نکنیم، وقتی self میزاریم هم میشه ولی یسری جاها توی دوره self نزاشتیم، این رو میدونیم که قطعا باید توی \_\_init\_\_() از self استفاده کنیم. ولی بقیه جاها چی؟

همونطور که خودم فکر میکردم:



-در ادامه رسیدیم به اینکه گفت توی یه همچین بازی هایی یدونه data set منطقی نیست و بهتره یدونه دیگه داشته باشیم، یه سایت معرفی کرد opentdb فکر میکنم بود تو دوره لینکش هست از اونجا میشه سوال های verify شده دریافت کرد که البته json تحویل میده که همون dictionary در پایتونه و ممکنه key:value های متفاوتی داشته باشه که ما question و answer رو نیاز داریم. دو راه داریم یکی اینکه data set رو تغیر بدیم به چیزی که باید. یا کد برنامه رو تغیر بدیم به چیزی که باید . (خب از جایی که data set ها امکان داره شامل تعداد بسیار زیادی از اطلاعات باشن پس تغیر دادن کل اونا منطقی نخواهد بود. منطقی، همینه که کد رو طوری تغیر بدیم که بسته به نیاز ما بتونه با اون data set کار کنه. )

-در IDE ها یه قابلیت وجود داره به نام reformat code که Indent هارو مرتب میکنه، کد رو خوانا تر میکنه و استفاده ما ازش در این قسمت این بود که data set ای که گرفتیم رو مرتب میکرد تا ازش سر دربیاریم که به عنوان یه dictionary داریمش یا list آیا یه List از dict هاست یا یه dict از List ها و کلی ترکیب دیگه. که در vscode باید یچیزی براش دانلود کنی. که با shift+alt+F کار میکنه، use that! (گویا، توی vscode قضیه یه مقدار سخت تره، بعدا باید پیگیرش باشیم چیز مهمیه.)

-نکته: اینجا دقیقا یه برنامه OOP داریم، که میتونیم به عنوان الگو بهش نگاه کنیم برای نوشتن برنامه های دیگه:

1. اینکه class ها و data set ها و برنامه اصلی بهتره جدا باشن، اینطوری واسه فهمیدن راحت ترن، واسه کامپیوتر فرقی نداره عملا.

2. در main.py بهتره طوری بنویسیم و در class ها طوری نام گذاری کنیم که در نگاه اول بدون اینکه یادمون باشه کدش چطوری نوشته شده بشه فهمید فرایند برنامه چطوری انجام میشه.

پایان day017

Day018

Turtle graphic, tuples and more importing modules

یه module داشتیم به نام turtle خاطرمون باشه و میتونستیم ازش برای استفاده در محیط گرافیکی که خودش ایجاد میکرد استفاده کنیم، نکته یادآوری اینه که برای اینکه اون صفحه رو بالا نگه داریم و با کلیک از دید ما خارج بشه و برنامه بسته بشه باید یسری کارای دیگه بکنیم. Screen() کلاس و استفاده از exitonmyclick() .

گفته شد که برای این رنگ ها از TK استفاده میکنه که اون هم از TK interface استفاده میکنه که یکی از راه های درست کردن یک GUI (Graphical User Interface) هست هی داره جالب تر میشه. البته فعلا که فکر میکنم فقط با همون turtle کار داریم، ولی جالب بود که بیان شد ، احتمال زیاد جلوتر باهاش کار خواهیم داشت.

-در قسمت اول گفته شد که از این چالش های turtle (فکر میکنم حداقل در این قسمت) زیاد داریم، به دو دلیل یکیش به چالش کشیدن مهارت، یکیش هم که مهمتره، **دنبالِ Documentation گشتن**  که یاد بگیریم باید فصلی باشیم، یعنی به صورت کلی خودمون بتونیم هرکاری خواستیم بکنیم به کمک document ها . میدونستیم خیلی مهمه ، واقعا هم خیلی مهمه، اینجا هم میگن خیلی مهمه.

-درمورد راه های مختلف Import کردن گفت: 3 راه داره، با Import کل فایل رو بیاریم، که باید برای دسترسی در هر خط به همون فایل اشاره کنیم و بعد به چیزی که توشه، From..import.. هم برای اضافه کردن یچیز مشخص از یک فایل و from..import \* برای اضافه کردن همه چیز از اون فایل. که گفت سومی هیچوقت استفاده نکنید باعث گیج شدن میشه (اینکه نمیدونیم از کجا اومده باعث گیجی میشه) درمورد دومی بهتره وقتی استفاده کنیم که بیشتر از 3 تا لاین با اون بخش کار داریم و درمورد اولی هم وقتی که 1 بار بیشتر باهاش کار نداریم. (برای اینکه سر ضرب میتونیم تشخیص بدیم این بخش از کجا اومده و مربوط به کدوم Module عه) و همچنین میتونیم از

استفاده کنیم ، چراکه در پروژه ها module ها میتونن اسم های خیلی طولانی ای داشته باشن، میتونیم با این روش کد نویسی رو راحت تر کنیم.

تا day018 004 08:50

ادامه

-یه نکته جالبی گفت درمورد اینکه چرا Pycharm ؟ گفت که تغیر ورژن از python 2 به 3 خیلی تغیرات داره و پروژه هایی که اونجا نوشته شدن توی python 3 نمیتونن ران بشن، برای اینکار از یه روشی که فکر میکنم فکر میکنم این باشه که معادل module ها و package ها در هر python رو تشخیص میده و نصب میکنه درمحل درست استفاده میشه که داخل pycharm اتفاق میوفته ، و دلیل اینه فکر میکنم. چراکه خودش گفت شاید بعضیاتون از مسیر pip رفته باشید گفت ولی این به این دلیل میتونه مشکل ساز بشه و ... (از همین جهت شاید بریم روی pycharm. نمیدونم) (و این هم گفته شد که از راه pycharm وقتی نصب کنیم package ها و module هارو ، میره توی یه folder بنام .venv یا یه همچین چیزی میشینه که این به معنیه که فقط module مربوطه در فایل های همین پروژه که روش نصب شد قابل استفاده است و هرجا بخوایم استفاده کنیم باید اضافه ش کنیم. )

تا انتهای day018 006

ادامه

خب ما یسری challenge داشتیم امروز که همشون کار با turtle بود که انجام شد رسید به مرحله آخر که پروژه پایانی روز 18 باشه. برای درست کردن یه Color pallet مثل چیزی که در دوره بود از یه package میخواد استفاده کنیم به نام colorgram.py که بهت بیشترین رنگ های استفاده شده از یه عکس رو میده.

تا

Day 018 010 07:30

ادامه

نکته مربوط به جلسه قبل: هرجا که میبینیم میخوایم یه Path\to\my\file بدیم به برنامه ها و شناسایی نمیکنه در حالی که از توی خودِ explorer کپی کردیم، باید از raw string استفاده کنیم. تنها کاری هم که باید بکنیم اینه که یه r بزاریم پشتِ استرینگ. اینطوری متوجه میشه که back-slash \ ها معنی دیگه ای بجز چیزی که نوشته شده ندارن.

-

ادامه

ابزار کار کردن با rgb راحت و سریع ، w3schools / rgb

- نکته ای که در حال استفاده از colorgram.py باید در نظر داشته باشیم اینه که از اونجایی که از روی یک تصویر رنگ هارو استخراج میکنه، پس رنگ پس زمینه (در این بحث سفید) رو هم استخراج میکنه، و باید خودمون مقادیر رو یبار چک کنیم قبل از استفاده.

-وقتی که میخواستیم tuple(r,g,b) رو در turtle استفاده کنیم ارور میداد که bad color number و ... سرچ که زدیم تو stack overflow راه حلش این بود که :

مثل اینکه این باعث میشه رنگ ها معنی بدن. نمیدونم. (بعد از اینکه جوابِ course رو نگاه کردم گفت که در قسمت های قبلی این رو دیدیم، ولی من چیزی یادم نمیومد، به هر ترتیب اگر ...)

پایان day018

ادامه

Day019

More turtle graphics, event listeners, state and multiple instances

پروژه پایانی این روز هم ساخت یه turtle race هست که کاربر پیشبینی کنه کدوم یکی برنده میشه.

-یه نکته بسیار مهم، وقتی میخوایم function ای رو به عنوان argument به function یا method دیگه ای ارسال کنیم باید حواسمون باشه که پرانتز نزاریم انتهاش، حالا چرا؟ چونکه در صورت گذاشتن ، function call اتفاق میوفته و کدش اجرا میشه، باید بدون پرانتز ارسال بشه.

-به هرکدوم از حالت های ساخته شده از یک class() میگن instance و به حالتی که هر instance داره و فرق هایی که داره نسبت به یه instance دیگه میگن state

ta day019 005 10:38

ادامه

بازی ساخته شد . نکته خاصی نداشت بجز اینکه وقتی میخوایم چندین تا از یک Object بسازیم منطقی تره که for بنویسیم و از حلقه استفاده کنیم نه اینکه تک تکشون رو کد بنویسیم ، این چیزی بود که وسط راه تغیر مسیر دادیم و باعث این شد که از قابلیت branch در گیت استفاده کنیم و مسیر دیگری رو develop کنیم . (چیزی که تو سرم بود این بود که چطوری میشه با For داخل یک dictionary اینکارو کرد که Key ها هم خودمون تعیین کنیم. )

End of Day019

Day020

خب در این روز قرار نیست مثل اینکه چیزی یاد بگیریم ، قراره از چیزایی که یاد گرفتیم استفاده کنیم تااااااااااااااااااااااااااااااااااااا ، بازی Snakes رو بسازیم. ها ها



اینا چیزاییه که باید مرحله به مرحله انجام بشن تا قضیه تکمیل بشه (Break the problem) :



که قراره این کار رو توی 2 روز در دوره انجام بده، روز اول سمت چپ و روز دوم سمت راست. فکر میکنم خودم تلاش کنم در ابتدا جالب خواهد بود.

پس ابتدا خودم میرم سراغش هرجا کم اووردم ، میرم دوره رو میبینم.

تا انتهای Day020 001

ادامه

خب بریم سراغ اینکه خودمون بسازیم، هرجا دیگه جدا کم اووردیم میریم و وقت رو تلف نمیکنیم، از هرچیزی هم که داریم و بلدیم و میدونیم هم مجازیم استفاده کنیم.

-خب الان انتهای تایم هستیم، خودمون تلاش کردیم بسازیمش ، مرحله یک که ساخت shape اولیه بود رو خوب رفتیم، مرحله دو موندیم، ما چیکار کردیم، گفتیم که نسبت به اولی میچینیم و اینکه همه از اولی تبعیت کنن، که البته درست بود ولی خب شکل درستی به خودش نمیگرفت.(از چه جهت؟ از این جهت که وقتی که تغیر جهت میداد آخریا میومدن جلو رفتگی اولی رو پر میکردن ) به این فکر افتادیم که یکاری کنیم که همه به سمت جلو حرکت کنن 10تا ، بعد زمانی که مثلا میگیم بره بالا ، coordinate اون لحظه ذخیره بشه و ثبت بشه که اگر اونجا رسیدن همه همون کار رو بکنن، تا Len(turtles) ، و زمانی که تموم شد هم که دیگه تموم شده دیگه. اینو باید جلسه بعد پیاده سازی کنیم، من کم نمیارم.

ادامه

-خب الان تونستیم درستش کنیم (snake food ) ، با یه call back function یا همون تابع برگشتی که البته هنوز نگفته کاری کردیم هرکدوم از Position قبلیِ مربوط به turtle بعدی تبعیت کنن که قبل از اینکه به جلو حرکت کنه داشته. الان level 2 انجام شده. بریم ببینیم چیکار باید بکنیم برای مرحله بعد.

-level 3 میشه snake food . که باید یسری circle آبی باشن که به صورت رندوم داخل صفحه سبز میشن و اگر بخوریمشون یه turtle باید ساخته بشه و به انتهای turtles اضافه بشه.

-که البته بین راهی یه نکته ای باید بگم: ما یه فکری هم زد به سرمون ، اونم این بود که method بنویسیم و method رو اضافه کنیم به object turtle ها. که همون عملیات تبعیت از بعدی رو انجام بده. راهش هم دراوردیم توی perplexity ولی دیگه انجامش ندادیم، حس میکنم راه حل درست تر اون بود، ولی خب.

-الان رسیدیم به جایی که غذارو به صورت رندوم یجایی قرار میده اما مشکل اینجاست که وقتی غذارو میخوره و score میگیره، غذای قبلی پاک نمیشه از روی صفحه، اینطوری هم نیست که دیگه score بده ، یجورایی بی مصرفه و فقط نشون داده میشه، جلسه بعدی باید رفعش کنیم و ادامه بدیم.

(خودمونیم، تنها خوب اومدیم جلو ها ، ایول)

ادامه

تا اینجای کار خودمون تونستیم body شو بسازیم، حرکتش رو فیکس کنیم، غذا رو رندوم اضافه کنیم و وقتی غذا میخوره بزرگ میشه و به score اضافه میشه. از حالت call back خارج شدیم و به صورت عادی پیاده سازیش کردیم تا سریع تر بشه چون وقتی score میگیره از سرعتش میوفته، الانم همینطوریه البته ولی کمتر. (البته این برای قبل از دیدن دوره س. خودِ دوره راه گشا بود.)

چیزایی که مونده اینه که وقتی به خودش میخوره باید بازی تموم بشه، یه دیوار باید داشته باشه، وقتی به دیوار میخوره باید بازی تموم بشه، اون بالا باید score نمایش داده بشه.

میریم دوره رو میبینیم و مقایسه میکنیم.

نکته جالبی که استفاده کرد از screen.tracer(0) بود که باعث میشه animation ها در turtle graphic نشون داده نشن و .... و از time استفاده کرد برای time.sleep مثل چیزی که در کد خودون توی ptk داشتیم.(

-در استفاده از این Library ها و package ها دوراه وجود داره، یا از ai کمک بگیریم و بپرسیم که آیا فلان کار رو میشه کرد یا خیر ، یا اینکه به مرور مهم هارو document بخونیم و یاد بگیریم تا یه چکیده ای دستمون بیاد که قضیه از چه قراره

) و باعث شد مشکل سرعت در ازای score رو که کند میشد رو رفع کنه. روشای جالبی میره. مثلا یکی دیگش اینه که فانکشن range() عملا مالِ خودِ python نیست و مالِ C عه پس نمیشه از keyword ها در call کردنش استفاده کرد ، نکته جالبیه، و اینکه در استفاده از range() ما میتونیم از start, stop, step استفاده کنیم، که یعنی میتونیم مقادیر منفی با استفاده از توابع خطی و غیر خطی هم داشته باشیم. جالبه.

Ta entehaye day020 003

ادامه

عنوانِ قسمت چهارم اینه، create a snake class & move to OOP که یعنی بهتر بوده یه Class میساختیم برای اینکار، منطقی هم هست دیگه باید کم کم یاد بگیریم از حالت procedural در بیایم و بریم توی oop . یعنی همه چیز رو class دار و object در نظر بگیریم و پیش بریم. (که این به این معنیه که هرچیزی در OOP یک شی هست که یسری attribute داره و یسری Method و یعنی به ترتیب ، یسری مشخصه داره یسری اطلاعات داره (که حالا یا ساخته میشن یا آماده حاضر اونجان) یا یسری عملکرد داره که روی اون اطلاعات انجام میشن .)

-الان challenge داریم که چیزی که نوشتیم رو ببریم روی OOp بعد الان یه اشتباه فاحش کردیم که دیگه نباید تکرار شه، در حینی که میخواستیم کد هارو ببریم توی class Snake اسم متغیر هارو تغیر دادیم. که این اصلا کار درستی نیست هرچی میره جلوتر بیشتر مشکل ایجاد میکنه، باید در ابتدا Class رو با name های قبلی بسازیم، ساخته که شد درست که عمل کرد اون زمان اسمارو میتونیم عوض کنیم . پس الان باید برنچ رو پاک کنیم مجدد برنچ بزنیم.

-خب رفتیم تا انتهای day020 و تمامی کد ها رو هم برسی کردیم و به موازات دوره تغیر دادیم. الان میمونه باقی موارد. از نظرم باید در زمان باقی مانده برم سراغ تغیر کامل به oop توی کدی که زدم، و اون رو کامنت کنم برای جلسه بعدی.

pishravi ta entehaye Day020

ادامه

-بخش مربوط به food و growth به صورت OOP پیاده سازی شد .

-خب تا اینجا هرچیزی که قبل از دیدن دوره کد زده بودیم رو با OOP پیاده سازی کردیم، مثلِ food که دارای .is\_ate() هست و snake که دارای snake.growth() هست. الان باید بریم نگاه کنیم ببینیم چه چیز دیگه ای باید به این پروژه اضافه بشه.

محض رضایِ راحتی:



باید اون بالا یه scoreboard بیاریم و در طی score گرفتن اونم آپدیت بشه.

Scoreboard هم طبق OOP ساختیم و میمونه wall و tail که واسه جلسه بعد

ادامه

-خب در حال حاضر باید بریم سراغ اینکه چطور کاری کنیم که وقتی به دیوار میخوره Lose کنه.

اول باید بریم سراغ document خودِ turtle ببینیم چیزی هست که بشه این رو براش پیاده کرد و اگر میشه از همون استفاده کنیم، باید توی Snake class هم یه method برای Lose درست کنیم. اگر هم خودِ turtle چیزی در این رابطه نداره، باید یطوری خودمون قضیه رو جفت و جور کنیم و یکی از چیزایی که همین الان تو سرم دارم اینه که Position هارو چک کنیم اگر از یه حدی بیشتر شد یعنی خورده به دیوار که این تو هر مرحله از اون while باید چک بشه. فعلا document

-چیزی مبنی بر اینی که میخوایم پیدا نکردیم، باید خودمون بسازیم، ابتدا باید بریم سراغ اینکه یه Box بسازیم سفید زیرِ score که معلوم بشه اونه بعد کاری کنیم که اگر خورد بهم که یعنی Position ش باهاش برابر شد بازی تموم شه.

-Box رو اول پیاده سازی کردیم، کاری کردیم وقتی میخوره بهش game over بشه ، بعد رفتیم class box رو ساختیم. جلسه بعدی باید در ابتدا به فکر پیاده سازی tail collision بشیم و بعد هرچی میتونیم oop ش کنیم.

ادامه

خب در حال حاضر اتفاقی که افتاده اینه: ما در تلاش بودیم برای نصب لینوکس روی لپتاپ ، همزمان هم فایل docx رو از حالت tracked حذف کرده بودیم ، چیزی که اتفاق افتاد این بود که فایل های روی لپتاپ به دلیل وجود bit locker به صورت کلی نابود شد و الان چیزایی که مربوط به pong game بوده رو کد هاشو داریم ولی نکته هاشو نداریم، الان هم وسطای day023 هستیم. میمونه فقط کد بزنیم و ادامه بدیم. این هم نمیدونیم که الان libreOffice Writer که روی لینوکس داریم استفاده میکنیم آیا مشکل ساز خواهد شد یا خیر. به هرحال ادامه میدیم، من کلا قصد move به linux دارم.

-این برامون میشه تجربه که دیگه فایل هارو از tracked خارج نکنیم.

-باز خوبه از git استفاده میکردیم وگرنه که الان کاملا فایل هامون به فنا رفته بود.

ادامه

-میریم برای ادامه ، اول باید یه تستی بکنیم ببینیم اوضاع چطوریه تا کجا پیش رفتیم و اینکه بریم از تو ویدیو ها break downرو دربیاریم.



اینم از این. حالاا میریم برای چک کردن اینکه تا کجا کد زدیم، میتونیم از git log استفاده کنیم.

تا جایی که خاطرم هست ما کلِ ستونِ چپ رو رفتیم و انجام دادیم. الان مونده ستونِ راست.



خب بازم وقتمون رفت برای یاد گیری لینوکس ولی یسری چیزا رو فیکس کردیم، فردا سر ضرب میشینیم پاش.

ادامه

خب میریم که داشته باشیم

-بهتره که بعدا برای اینکه درجه reflect چند باشه یه فانکشن جداکار بنویسیم ولی فعلا بیخیال. (منظورم اینه که از اونجایی که کد رو خیلی کثیف نوشتیم بهتره که بریم و فانکشن براش بنویسیم. )

-خب برگشتیم به بازی. عرضم به حضور شما که reflect ها همه درست دارن کار میکنن، الان عملا بازی رو ساختیم، فقط میمونه حل کردن موقعیت وقتی که paddle miss اتفاق میوفته، و اینکه چطور score بدیم به بازیکنا. جلسه بعد ایشالا.

ادامه

خب الان باید بریم سراغ miss\_paddle

-نکته: داریم متوجه میشیم دیگه هرچی کد میزنیم، باید main.py اینطوری باشه که بخونیش و متوجه بشی منطق برنامه چطوریه، باید بیشتر خوندنی باشه، و اینکه حالا منطق کد نویسی چی هست رو داخل فایل مربوط به اون class یا module بخصوص پیدا کنی.

Khb dar hale hazer nmidonm chera keyboard farsim kar nmikne. Vali number baraye scoreboard sakhte shd, jalase badi bayad berim soragh chapi va inke add knm scoring ro va update score ro. Va halati ke beshe game ro edame dd

number برای scoreboard ساخته شد، جلسه بعدی باید بریم سراغ چپی و اینکه اد کنیم scoring و فرایندش رو و بعد از اینا همه باید کاری کنیم بازی repeatable باشه به صورت خودکار هر دست بره از اول. مثلا وقتی enter رو میزنیم بره از اول.

ادامه

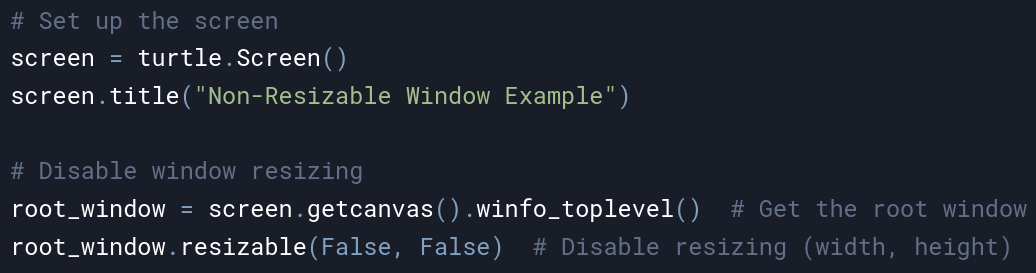
-خب امروز scoreboard رو کامل کردیم، بعد رفتیم سراغ اینکه بازی رو repeatable کنیم. تا به اینجای کار از نظر من چیزی نمیمونه، همه کاراش انجام شده. فقط اینکه جلسه آینده بهتره یه notification طور درست کنیم برای وقت هایی که منتظره تا enter یا escape بزنیم که player متوجه بشه که باید چکار کنه. و همچنین خوبه که ~~یه کامنت طوری یه گوشه ای از صفحه باشه که player ها با paddle هاشون بدونن چطوری باید کار کنن.~~ که به نظرم بهتره که زیرِ white\_box اد بشن، و اینکه بگردیم ببینیم میتونیم راهی پیدا کنیم که ~~اجازۀ full screen کردن رو از کاربر بگیریم~~ یا نه. که در همون اندازه ای که ما گفتیم بمونه . اینا دونه دونه برای جلسه بعد هستن، بعدش باید بریم جواب دوره رو ببینیم و یادداشت برداری کنیم.

ادامه

خب بریم برای فیکس کردن چیزایی که دیروز بهشون پی بردیم و بریم سراغ دیدن ویدیو ها.

-یادآوری: turtle graphic که داریم باهاش کار میکنیم با library ای به نامِ tkinter ساخته شده. حالا این چیه؟ toolkit آماده ای هست برای ساخت graphic UI با استفاده از python . (یعنی چی؟ یعنی اینکه اگر داریم با python کد میزنیم و میخوایم یه user interface گرافیکی درست کنیم برای برنامه ای که نوشتیم ، باید از این استفاده کنیم. آیا چیز دیگه ای هم هست برای این استفاده؟ الان یه سرچ زدم و تعدادشون زیاده، از فریمورک های مختلف گرفته تا استفاده های بخصوص نمونه داره ، ولی از deepseek که پرسیدم کدوم از همه محبوب تره گفت pyQt/pySide)

-حالا با استفاده از همین والد turtle میتونیم کاری کنیم که window بازی ای که نوشتیم غیر قابل تغیر از جهت اندازه باشه از طرف کاربر. که نیاز به توضیح این داره که در tkinter پنجره اصلی root window نامیده میشه و هر چیزی داخلش که بلاک بلاک تعریف میشه widget نام دارند. ما برای این کار باید ابتدا root ویندو رو مشخص کنیم و بعد بهش دستور بدیم که resize = false :



همینطور که میبینید اینجا یه متغیر داریم برای مشخص کردن root\_window و بعدش بهش دستور root\_window.resizable(False, False) رو میدیم. بریم تست.

-تا اینجا هرچی نکته جلسه قبلی چیده بودیم که انجام بدیم انجام شده، مضاف بر اون module عه screen.py هم کامل oop کردیم، ولی ورژن بدون OOPش رو نگه داشتیم با اسمِ screen+.py .

جلسه بعد ، بعد از مرور ، سر ضرب میریم سراغ ویدیو ها و چک میکنیم و نکات رو مینویسیم و اگر نیاز بود یک branch میسازیم و اصلاح میکنیم.

ادامه

خب ، الان باید بریم سراغ ویدیو ها از قسمت 002: لازم به ذکره که تغیراتی که با دیدن course بهشون میرسیم رو توی branch = course\_way تغیر میدیم و در انتها push میکنیم.

-تا به اینجای کار اندازه صفحه ای که دادیم فرق داره، و اندازه paddle ای که دادیم هم فرق داره… خیلی فرق کمی داره و اصلاح شد ،

-سوالی که دارم اینه که آیا این event listener هارو میشه بزاریم تو کلاس یا نه؟ (خودش این کار رو نکرد و همون داخل main.py تعریف کرد)

-یه مشکلی که داشتیم و الان بهش دارم فکر میکنم دقیقا همین فکر کردن قبل از عمل کردنه. برای وقتی که میخواستیم collision with wall and ball رو بدست بیاریم عملا اضافه کاری کردیم ، به ما ربطی نداشت وقتی که توپ میخورد به ضلع های شرقی و غربی. ولی واسه اونا هم کد نوشتیم. کدی که عملا نیازی به نوشتنش نبود هم وقت گذاشتیم بهش فکر کردیم هم نوشتیمش

-خب ، یه فرق اساسی ای که داریم اینه که ما نشستیم دقیق زاویه ای حساب کردیم و برای reflect کردن یا به قول دوره bounce کردن از setheading() استفاده کردیم که خیلی باعث شد زمان بزاریم سر محاسباتش ، جدا اعصاب خورد کن شده بود، تتا بگیر و زاویه تابش پیدا کن و … اینجا از ابتدا برای حرکت دادنش از ()goto استفاده شد و تنها کاری که در جهت برگردوندن توپ انجام شد این بود که y رو معکوس کنه و یه منفی بندازه روش. همین. ما پیر خودمون رو درآوردیم یجورایی، کد اضافی هم نوشتیم واسه ضلع شرقی و غربی. ولی خب at the end of the day ما کار رو خودمون انجام دادیم، ملالی نیست، تغیرشم نمیدم، چون حس میکنم این چیزی که دوره داره مینویسه ، فقط چهار جهت داره، ولی چیزی که من نوشتم شامل تمامی زوایا به صورت دقیق میشه. (کنترلش کن ولی) (طوری که دوره رفته اینه که از x , y جدید ساختن و استفاده از اون توی ()goto استفاده میشه، که با منفی کردن مقدار y برای بازتابِ دیوار ها (بالا و پایین) و منفی کردن مقدار x برای paddleها خیلی راحت انجام شد. البته بازم قصد تغیرش رو ندارم.)

-گویا وقتی که یه miss\_paddle اتفاق میوفته، باید جهت حرکت توپ برعکس بشه، یعنی اگر راستیه باخته باید جهت حرکت توپ به سمت چپ باشه و بالعکس. که این رو ما انجام ندادیم. (انجام شد.)

-یه چیز دیگه ای هم که متفاوته scoreboard هست، ما برای هرکدوم از عددها یک turtle جدا تعریف کردیم و واسه هرکدوم یه object ساختیم، که حالا که میبینم نیازی نبوده، و یجورایی الکی شلوغش کردیم، ولی خب بازم داره کار میکنه.

-یه نکته ای که گفت اینه که میتونیم این رو به بازی اضافه کنیم که وقتی که توپ به هرکدوم از paddle ها برخورد میکنه، یه مقدار سرعت بازی بیشتر بشه، جالب بود.

-نکته بعدی اینکه گویا وقتی دکمه w رو نگه داری یبار بیشتر کار نمیکنه، ولی وقتی باقی دکمه هارو نگه داری تا زمانی که ولش نکنی به حرکت خودش ادامه میده. گفت این مشکل از turtle module هست که این اتفاق میوفته، (و گفت اگر دارید با یکی از دوستاتون بازی میکنید میتونید تقلب کنید و همیشه جهت راست بشینید :) )

-اینم از انتهای این بازی ، خیلی وقت گرفت ازم ولی ارزششو داشت.

گفت که ترس یچیزیه که میتونه و این قدرت رو داره که بین چیزی که میخواید و شما بایسته، بهش فکر کنید که چرا میترسید، به این فکر کنید که چی میتونه پیش بیاد اگر ترستون اتفاق بیوفته (احتمالا خیلی نخواهد بود) و این باعث میشه که با ثبات قدم بیشتری پیش برید. حرفش جالب بود، برای تمامی جهاتِ زندگی :)

انتهای Day022

Day023

خب وارد روز بعد میشیم.

بازی قراره بسازیم، بازی ای که یسری ماشین به صورت افقی رد میشن و یه لاکپشت باید در صورتی که میتونه فقط به جلو حرکت کنه و توسط کاربر کنترل میشه بدون برخورد به این ماشینا به بالاترین سطح برسه و به اصطلاح از خیابون رد بشه. و مرحله به مرحله این بازی پیش میره level by level و هر مرحله هم باید سخت تر بشه.

از اونجایی که گیم قبلی pong رو خیلی روش وقت گذاشتیم و اینجا هم عملا قراره همون کارارو انجام بدیم، به نظرم با دوره پیش بریم بهتره. هرجا هم گفت که میریم کدشو میزنیم باهاش. برو بریم

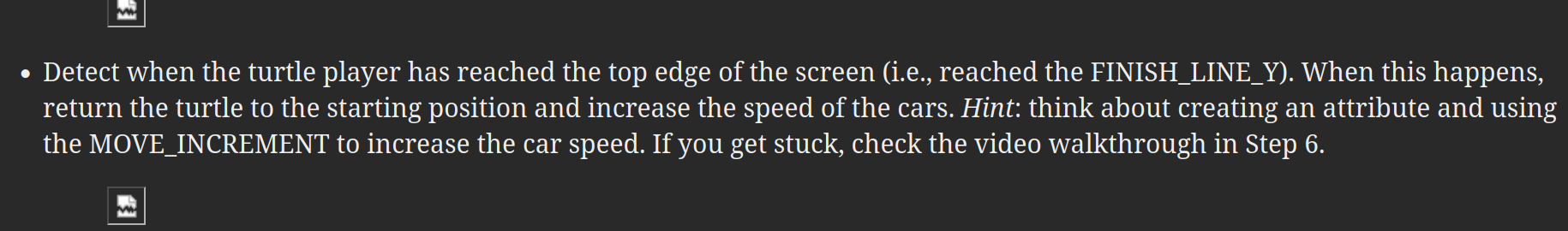
- خب توی فایل های دوره نوشته بود که یکی از سطوح سختی این قسمت رو انتخاب کنید که ما HARD رو انتخاب کردیم (EXPERT هم داشت ولی از اونجایی که نمیخواستیم همشو خودمون انجام بدیم رفتیم روی HARD)، در حال حاضر turtle رو ساختیم، نمونه اولیه car هم ساختیم. ادامه مراحل بمونه برای جلسه بعد.

ادامه

میریم برای ادامه نکته ای بود مینویسیم. الان وسطای day023 هستیم.

-تا الان مرحله یک از break down problem رو کامل رفتیم (درست کردن یه player که با Up به بالا حرکت میکنه در starter code و الان توی مرحله بعد هستیم که باید car رو کامل کنیم و به صورت رندوم از یه فضای مشخصی از صفحه اون هارو به سمتِ چپ و با یک سرعت مشخصی حرکت بدیم. تا الان نمونه اولیه رو درست کردیم.

-یادآوری: وقتی میخوایم در list ها در پایتون یه element رو بر اساس اینکه محتواش چیه پاک کنیم باید از متود ()remove استفاده کنیم.

-تا این مرحله انجام شد

مونده افزایش سرعت بعد از هربار که به خط پایان میرسه که جلسه بعد.

ادامه

خب بعد از یه وقفه طولانی میریم که داشته باشیم. فقط هرچه سریعتر بهتر.

-اضافه شدن سرعت رو deploy کردم اما یه مشکلی داریم، اونم اینه که چون reuse رو اصولی نساختیم بعد از افزایش سرعت مقدار ماشین هایی که دیده میشه تو محیط بازی هی کم و کمتر میشه (ولی خب سرعتشون بیشتره) فعلا کاریش نداریم تا شاید بعدا درستش کردیم.

-مشکل خالی بودن صفحه بازی وقتی که سرعت خیلی زیاد میشد رو با کمک کم کردن TIME\_SLEEP حل کردیم، اینطوری بازی سریع تر میشه جای اینکه فقط ماشینا سریع تر بیان. (البته با چک کردن کدهای خودِ دوره میشه دید که یه همچین کاری نکرده و همون سرعت forward های ماشین ها رو افزایش میده به یه مقدار مشخصی در هر مرحله.)

-بازی به صورت کامل ساخته شد ، جلسه آینده باید ویدیو هارو ببینیم.

ادامه

خب ، بازم بعد از یه وقفه خیلی طولانی اینجاییم.

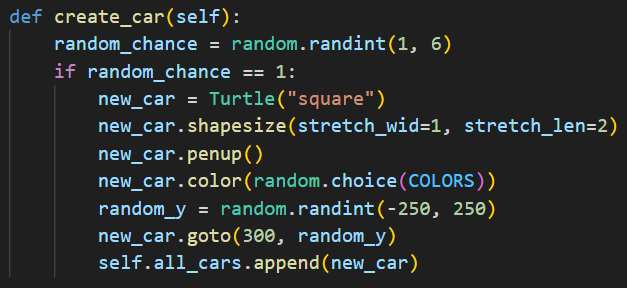
تا اینجا با Gemini فایل رو یه مرور کلی کردیم ، بازم باید مرور کنیم البته، برای مثال باید ابتدا یه 15 دقیقه مرور کنیم از ابتدا بیایم جلو، جلسه قبل هم مرور کنیم و بریم برای مطالب جدیدتر.

-یادآوری: که از debugger میشه خیلی خوب استفاده کرد و اصن در پروژه های بزرگ باید استفاده بشه قاعدتا. برای مثال ، توی کلِ پروژه یسری breakpoint داریم (##که مثل اینکه breakpoint های مختلفی داریم، بهتره یه بار درموردش یه تحقیقی بکنیم) اینارو توی debugger اون پایین در سطح همۀ پروژه برات لیست میکنه، و میتونی قشنگ بین همشون trace کنی.

-

ادامه

-ابتدا به سراغ ویدیو های پایانی Day023 رفتیم (وای که چقدر طول کشید این بیصاحاب) ، در انتها کدهای آماده ش وجود داشت. و این شد که رفتیم برای چک کردنش چراکه کدهای خودمون هم یدور چک کردیم . رسیدیم به کدِ زیر:



این راه حلیه که course در پیش گرفته، فرقش با مالِ ما اینه که ما هر 6 مرحله یکبار از مراحل while یک ماشین میساختیم (یا از resue\_list برمیداشتیم و استفاده میکردیم) ولی اینجا هربار که به create\_car میرسه، انگار یه تاس میریزه و اگر 1 اومد یه ماشین میسازه. خب این باعث چی میشه؟ باعث این میشه که بازی randomness بیشتری خواهد داشت از نظرم. کارِ جالبیه.

-همچنین حین تجربه بازی ای که خودِ دوره نوشته ، متوجه میشیم که هرمرحله ای که میگذره واقعا سرعت افزوده میشه و مثل چیزی که ما نوشتیم هیچوقت car های ساخته شده از دوباره استفاده نمیشن ، پس یعنی با اضافه شدن مراحل، تا یجایی میشه پیش رفت، یعنی یه حدی داره، چراکه ram از اون به بعد پر میشه و دیگه کشش نداره.

-البته الان که چک میکنم یه فرق دیگه هم میبینم، ما TIME\_SLEEP رو ضرب در 0.9 میکردم و باعث میشد به دلیل کمتر منتظر موندن برنامه بازی سریعتر بشه، اما اینجا سرعت هر ماشین رو زیاد میکنه جدا. یادمه یه مشکلی وجود داشت در این مورد.

پایانِ Day023

Day024

Working with local file system

-با ذخیره کردن اطلاعات از برنامۀ ران شده (مثل بازی snake ای که ساخته بودیم) ، مثلا میتونیم highest score رو نگه داریم دقیقا مثل خودِ بازی که نگه میداشت. و در ادامه توضیحاتش گفت که میتونیم با این کار خیلی از کارهایی که میخوایم انجام بدیم رو به سرعت با python اتوماتیک کنیم. مثالش هم یه نامه دعوت هست که قراره یسری جاهاش اسمِ گیرنده بخوره ولی باقیش یکسانه، خب میشه این رو کد کرد که به صورت اتوماتیک انجام بشه و به قول خودش فیلم ببینیم در حین انجامش یا بازی کنیم.

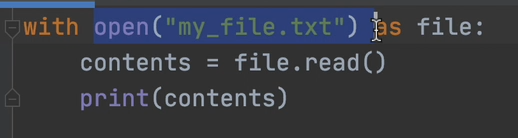
-که این نگه داری اطلاعات ، برنامه نویسی رو وارد فاز جدیدی میکنه که به قول مهندس نجاتی "از حالتِ دانشجویی خارج میشه" . و میتونه متصل بشه به database ها و اطلاعات ذخیره کنه و کوئری بزنه و ... که خیلی کاربردی ترش میکنه .

-open files, read, modify, write, close با python

-برای باز کردن یک فایل در کدهامون از open()استفاده میکنیم که به عنوان argمیشه خیلی چیزا بهش فرستاد، اولیش هم اسمِ فایلی هستکه میخوایم بازش کنیم.

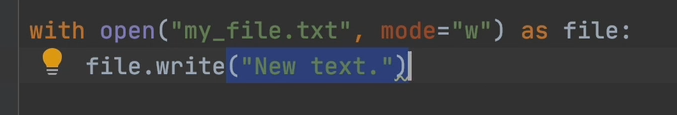
-و البته که کاری برات نمیکنه وقتی که باز شد. باید از یه functionدیگه به نام read() استفاده کنی که محتوای اون فایل رو بخونه، و بعد داخل یه متغیری ذخیره کنی و بعدش میتونی ازش استفاده کنی. (دقت بشه که الان فقط داری درمورد read صحبت میکنیم و نه write ) . بعدشم وقتی که کارمون تموم شد باید قاعدتا اونی که open() کرده بودیم رو close() کنیم که دقیقا با close() اینکارو انجام میدیم. (یادآوری: هرچیزی در pythonباید داخل یک variable ذخیره بشه و این فایل ها هم از این قاعده مستثنی نیستن. ) چراکه اگر close()نکنیم resource های سیستم به اون فایلی که عملا کارمون باهاش تموم شده درگیر میمونه و نمیشه دیگه ازش استفاده کرد و از یه جایی به بعد امکان bluescreen گرفتن پیش میاد، چون ram مثلا پر شده کامل.

-حالاکه این راه رو یاد گرفتیم (و قابلیت تست داره میشه باهاش محتوای یه فایل رو خوند و با پایتون پرینت کرد) بعدش، یه راه دیگه وجود داره که باید از اون استفاده کنیم ، چراکه ممکنه یادمون بره که fileای که باز کردیم رو close() کنیم، میتونیم به جای اینکار از عملگر with استفاده کنیم :



که به indent ها دقت بشه دیگه باید داخل باشن. این with به صورت خودکار وقتی متوجه میشه که ما از file ای که open() کردیم استفاده ای نداریم، خودش عملیات close() رو انجام میده. پ

-نکته درمورد open() : وقتی که یه فایل رو open()میکنیم به صورت پیشفرض با استفاده از mode=’r’ باز میشه که معنی Read-only بودن هست، و وقتیکه بخوایم چیزی رو داخلش تغیر بدیم بهمون error میده. باید وقتی میخوایم چیزی رو تغیر بدیم در حالت write بازش کنیم .که میشه mode=’w’ . البته با اینکار وقتی که چیزی رو file.write() کنیم ، همۀ محتوای قبلی پاک میشه و این چیزی که write کردیم جایگزین میشه:



اما اگه میخوایم این اتفاق نیوفته باید mode=’a’ بزاریم که به معنای appendهست و به انتهای فایل اضافه میشه. (##سوال پیش میاد که چطوری فایل رو از وسطاش تغیر بدیم که فکر میکنم جزو موارد advanced باشه. یه سرچ کوچیک بزن)

- نکته درمورد mode=’w’ : اگر یه فایلی رو با mode=’a’ باز کنیم که وجود نداره؛ و مثلا چیزی هم توش بنویسیم، پایتون اون فایل رو برامون ایجاد میکنه و اون رو توش مینویسه.

-open(mode=’’) ارگومان mode حالت های مختلفی داره، الان هم داشتیم تست میکردیم متوجه شدیم که اگر به اشتباه، در حالت w باشیم و read() انجام بدیم باعث میشه فایل خالی بشه. برای اینا احتمال زیاد نکاتی وجود داره. اما اگه نبود بعدا برو دنبالش .

-نکته درمورد git: ما زمانی mergeمیکنیم که کارهای branchمون تموم شده و میخوایم به main اضافه کنیم. پس بهتره که به جای استفاده از این دستور : git merge <featured-branch> از دستور git merge <featured-branch> -m <a-message-for-merge> استفاده کنیم تا یه پیامی از دلیل merge بمونه. مثل کامیت ها که همشون مشخص هستن که چه اتفاقی در اونها میوفته.

-

ادامه