دانشکده‌ی فیزیک
دانشگاه صنعتی شریف

نقشه‌ی راه پایتون برای درس شبیه‌سازی

پاییز ۱۴۰۰

---

۱ مقدمه

همان‌طور که دکتر اجتهادی نیز اشاره فرمودند برای این درس هیچ زبان برنامه‌نویسی به خصوصی مدنظر نیست و حتی مهم‌تر از آن، هدف این درس هم به هیچ‌وجه آموزش برنامه‌نویسی نخواهد بود. ولی از آن‌جایی در میان شما بی‌شک افرادی هستند که قصد دارند برنامه‌نویسی را با این درس جلو ببرند و یاد بگیرند و هم‌چنین افرادی که قبلا با پایتون کار کرده و آشنایی مناسبی دارند ولی در مورد اینکه در این درس به چه مهارت‌هایی بیش‌تر نیاز خواهند داشت و از چه ابزارهایی باید استفاده کنند سردرگم هستند، در تیم دست‌یاران آموزشی تصمیم بر آن شد که یک نقشه‌ی راه مختصر برای زبان پایتون و مهارت‌های لازم در آن برای این درس بر اساس منابع موجود در وب آماده شود تا قدری به سوالات و نگرانی‌های شما در این زمینه پاسخ دهد؛ امیدواریم که راه‌گشا و مفید نیز واقع شود!

مرجع اصلی این نقشه‌ی راه، بر اساس لکچرنوت‌های Getting started with Python for science می‌باشد که در آن به آشنایی با اکوسیستم پایتون، مفاهیم پایه‌ای پایتون، پکیج‌های Numpy، Matplotlib و Scipy می‌پردازد. به جز پکیج Scipy که در این درس نیاز چندانی به آن نخواهیم داشت، بقیه‌ی مطالب ذکر شده بسیار پرکاربرد و مهم هستند و در این لکچرنوت‌ها به خوبی و به طور کامل پوشش داده شده اند. هر چند که در بعضی از بخش‌ها به موضوعاتی پرداخته شده که چندان مهم و اساسی نیستند و یا به بخشی از مفاهیم و مطالب به صورت جزئی اشاره شده است؛ به همین دلیل در این نوشته تلاش شده که مفاهیم مهم و یا غیر مهم موجود در هر بخش ذکر و هم‌چنین برای مفاهیمی که به خوبی پوشش داده نشده اند، مراجع بیش‌تری معرفی شود. در ادامه به معرفی بخش‌های مختلف این لکچرنوت‌ها می‌پردازیم.

۲ آشنایی با اکوسیستم پایتون (بخش ۱.۱.)

در این بخش ابتدا به معرفی اکوسیستم پایتون و ضعف‌ها و مزیت‌های آن پرداخته و مقایسه‌ی مختصری نیز با زبان‌های مطرح دیگر در این حوزه انجام شده است. شاید دانستن این موضوعات اهمیت چشم‌گیری در شروع کار نداشته باشد ولی دانستن ویژگی‌های مثبت و منفی یک زبان به ما کمک می‌کند تا پیش از انجام یک کار و پروژه بدانیم که آیا این زبان پاسخ‌گوی نیازهای ما خواهد بود یا نه و چه زمان مشکلی که با آن روبرو می‌شویم ناشی از محدودیت‌های آن زبان است و نه عدم توانایی خود ما.

در ادامه به سراغ نصب پایتون و محیط کاری مناسب می‌رود. در این‌جا نیز همانند انتخاب زبان، آزادی عمل کامل خواهید داشت ولی توصیه‌ی ما به استفاده از Jupyter Notebook است. هم‌چنین برای آن که از قابلیت‌های مدیریت فایل و پروژه، debuging و autocomplete کد بهره ببرید، ادیتور Visual Studio Code را به شدت توصیه می‌کنیم که از notebooks نیز پشتیبانی می‌کند. برای آشنایی مقدماتی با Jupyter Notebook می‌توانید از مقاله‌ی خوب How to Use … و برای مباحث پیشرفته‌تر از … Tutorial: Advanced استفاده کنید.

۳ آشنایی با زبان پایتون (بخش ۲.۱.)

در این بخش به مفاهیم پایه و اساسی پایتون و ابزارها و توانمندی‌های آن پرداخته شده است. لازم به ذکر است که در این بخش گفته شده که قطعه کدها را در Ipython shell اجرا نمایید، ولیکن شما با استفاده از مهارت کسب شده در قسمت قبل، از Jupyter Notebook استفاده کنید. هم‌چنین توجه نمایید که تفاوت‌های ذکر شده‌ی مربوط به دو نسخه‌ی Python 2 و Python 3 کاملا بی‌اهمیت هستند و شما تنها کافی است دستورات مربوط به Python 3 را فرا بگیرید؛ زیرا نسخه‌ی پیشین پایتون به طور کامل منسوخ شده است و دیگر کاربردی نخواهد داشت. در ادامه به نکات مهم هر یک از زیربخش‌ها خواهیم پرداخت.

- در زیربخش ۲.۲.۱. به معرفی تایپ‌ها و ساختارهای‌داده اولیه‌ی موجود در پایتون پرداخته شده است. ساختارهای داده زیربنای اصلی هر زبان برنامه‌نویسی هستند. در این بخش به پشتیبانی پایتون از اعداد گنگ، کار با آرایه‌ها، تفاوت میان شی‌های(objects) قابل‌تغییر (immutable) و غیرقابل‌تغییر (mutable)، تفاوت‌های میان list و tuple چه از لحاظ کارکرد و چه از لحاظ سرعت و مفهوم keys to values map در dictionary توجه بسیار کنید.

- در زیر بخش ۳.۲.۱. به آموزش روش‌های کنترل جریان کد پرداخته شده است. پس از تایپ‌ها و ساختارهای داده، این دسته اصلی‌ترین زیربنای هر زبان هستند و به صورت تئوری تنها با فراگرفتن این دو بخش، توانایی نوشتن هر کدی در آن زبان را خواهید داشت، ولی به احتمال زیاد با سختی فراوان، پس چندان عجله نکنید! در این قسمت به زیربخش ۵.۳.۲.۱. توجه بیش‌تری کنید. روش‌های ذکر شده در آن علاوه بر راحت‌تر کردن کار خودتان، کدهایتان را هم بسیار خواناتر می‌کند.

- نوشتن یک کد تمیز و قابل توسعه که به راحتی بتوانید حتی در آینده هم با آن کار کنید، بدون استفاده از توابع امکان‌پذیر نیست. پس زیربخش ۴.۲.۱. را با دقت بخوانید و یادتان باشد که هر قسمت از کدتان که یک کار و وظیفه‌ی مشخص را انجام می‌دهد و امکان این که در قسمت‌های دیگر نیز مورد استفاده قرار بگیرد را دارد، باید تبدیل به یک تابع شود. با گذشت زمان و تمرین، به مرور خودتان پیش از شروع به نوشتن کد می‌توانید تشخیص دهید که چه بخش‌هایی باید به صورت تابع نوشته شوند. در بخش ۴.۴.۲.۱. به تفاوت میان پاس دادن آرایه‌ها و متغیرهای ساده به توابع دقت کنید. بخش بزرگی از مشکلات ابتدایی افراد در استفاده از توابع به این موضوع برمی‌گردد. بخش ۵.۴.۲.۱. را بخوانید ولی هیچ‌وقتِ هیچ‌وقت از متغیرهای global استفاده نکنید! به شدت کد را کثیف و ناخوانا می‌کنند. بخش ۶.۴.۲.۱. نیز می‌تواند بسیار کارآمد باشد؛ به خصوص در مواقعی که می‌خواهید یک سری متغیرهای زیادی را به عنوان تنظیمات به یک و یا چند تابع و کلاس پاس بدهید. هم‌چنین برای نوشتن Function Wrappers هم نیاز به آن‌ها خواهید داشت. برای آشنایی با این مفهوم می‌توانید از آموزش ... Function Wrappers استفاده کنید.

- همان‌طور که آن قسمت‌هایی از کد که وظیفه‌ای مشخص و قابلیت استفاده‌ی دوباره دارند را در قالب توابع می‌نویسیم، مجموعه‌ای از توابع، کلاس‌ها و متغیرهایی که کارکردی در راستای یک‌دیگر دارند و می‌توان به آن‌ها به عنوان یک مجموعه‌ی واحد نگاه کرد را نیز در قالب module می‌نویسیم. در زیر بخش ۵.۲.۱. به این موضوع پرداخته شده است. شاید برای کدهای ابتدایی به نظر برسد که استفاده از ماژول‌ها کاربرد چندانی ندارد، ولی در ادامه و برای تمرین‌های پیچیده‌تر انتهایی، بسیار به تمیز و خوانا شدن کدهایتان کمک خواهد کرد و باعث می‌شود که حتی بتوانید از برنامه‌هایی که برای تمرین‌های قبل نوشته‌اید، به راحتی دوباره استفاده کنید. در بخش ۲.۵.۲.۱. به کادر قرمز نوشته شده توجه بسیار کنید و هیچ‌گاه تمام توابع و دیگر چیزهای موجود در یک ماژول را با استفاده از دستور **from** [module name] **import** * وارد کد خود نکنید. و در نهایت بخش ۷.۵.۲.۱. که به چند شیوه‌ی مهم برای تمیز و خوانا نوشتن کد پرداخته که خواندن و عمل کردن به آن‌ها به شدت توصیه می‌شود.

- در بخش ۶.۲.۱. به کار کردن با فایل و خواندن و نوشتن آن‌ها پرداخته شده است. تا حدود اواسط کلاس به کار با فایل نیازی نخواهیم داشت، از این رو پیش‌نهاد می‌شود فعلا از این مبحث عبور کرده و در زمان مناسب به آن مراجعه کنید.

- بخش ۷.۲.۱. در مورد تعامل کردن با سیستم‌عامل از طریق کتاب‌خانه‌های موجود است که کاربردهای محدودی در بعضی مواقع خواهند داشت. از این رو پیش‌نهاد می‌شود از این مبحث نیز عبور کرده زیرا هرگاه در آینده به آن‌ها نیازی پیدا کنید، با یک جست‌وجوی ساده دستورات مربوط را خواهید یافت.

- در بخش ۸.۲.۱. به مدیریت Exceptions پرداخته شده است. در هنگام نوشتن برنامه‌هایی که در تعامل با عوامل دیگر از جمله سیستم عامل (در هنگام کار کردن با فایل‌ها)، کاربر (گرفتن پارامترهای ورودی) و شبکه (گرفتن و یا فرستادن اطلاعات) هستند، ممکن است خطاهایی پیش بیایند که از کنترل ما خارج باشند و نتوانیم جلوی آن‌ها را بگیریم، همانند قطع شدن ارتباط با سرور. از این رو برای آن‌که اجرای برنامه متوقف نشود، لازم است که بتوانیم این خطاها را به درستی مدیریت کنیم. به همین خاطر هم از لحاظ کارایی و هم خوانایی، این مبحث بسیار مهم است. ولی از آن‌جایی که در برنامه‌هایی که برای این درس خواهیم نوشت به احتمال خیلی زیاد با این موارد روبرو نخوایم شد، می توانید با خیال راحت از این مبحث نیز عبور کنید.

- برنامه‌نویسی شی‌گرا یکی از اصلی‌ترین و پایه‌ای‌ترین پارادایم‌های برنامه‌نویسی است که علاوه بر افزایش خوانایی و قابلیت استفاده‌ی مجدد کد، شیوه‌ی نگاه شما به مسئله‌ای که با آن‌ها روبرو شده‌اید و می‌خواهید آن را با استفاده از هر زبانی پیاده‌سازی کنید را نیز به شدت تغییر می‌دهد. شاید در ابتدا مفاهیم آن قدری گیج‌کننده و سخت به نظر بیاید ولی با گذشت زمان و تمرین، شیوه‌ی برنامه‌نویسی شما را دگرگون خواهد کرد. ولی متاسفانه در بخش ۹.۲.۱. به مفاهیم شی‌گرایی چندان پرداخته نشده و تنها به صورت خلاصه، شیوه‌ی پیاده‌سازی آن در پایتون ذکر شده است؛ از این رو چند منبع بیش‌تر در این زمینه معرفی می‌کنیم. اگر می‌خواهید تا حد ممکن سریع و با تکیه‌ی بیش‌تر بر مفاهیم، شی‌گرایی را بیاموزید، به سراغ مقاله‌ی ... Understand بروید. ولی اگر دوست دارید که به صورت عمیق‌تر این مبحث را فرابگیرید، برای کسانی که با آموزش‌های ویدئویی راحت‌تر هستند کورس ... Object-Oriented Programming و برای آن دسته که آموزش‌های نوشتاری را ترجیح می‌دهند مقاله‌ی (OOP) in Python 3 را پیش‌نهاد می‌کنیم.

در آخر به یاد داشته باشید که یک کد غیربهینه‌ی خوانا، در طولانی مدت، ارزش بالاتری از یک کد بهینه‌ی ناخوانا خواهد داشت! پس تا حد ممکن good practices و پارادایم‌های مناسب را رعایت کنید.

۴ آشنایی با تفاوت‌های میان Python 2 و Python 3 (بخش ۳.۱.)

همان‌طور که در بخش قبل نیز ذکر شد، نسخه‌ی قدیمی پایتون کاملا منسوخ شده است و دیگر استفاده‌ای نخواهد داشت؛ به همین خاطر از این بخش عبور کرده و به سراغ بخش جذاب بعدی بروید!