بسمه تعالی

کدهای نوت بوک فصل آشنایی با پایتون:

https://github.com/babaktei/datamining-python

دوره کامل داده کاوی با پایتون در آکادمی همراه اول:

https://hamrah.academy/course/۱۵۰

babaktei@gmail.com

https://www.linkedin.com/in/babak-teimourpour-۷۸۷۷۴۸۲b/

داده‌کاوی با پایتون
به همراه تحلیل شبکه‌های اجتماعی

دکتر بابک تیمورپور
(عضو هیات علمی دانشگاه تربیت مدرس)
وحید یادگاری
محمد فاتحی پیکانی

داده‌کاوی با پایتون به همراه تحلیل شبکه‌های اجتماعی
تیمورپور - یادگاری - فاتحی

Data Mining in Python with Social Networks Analysis

Babak Teimourpour
Vahid Yadegari
Mohammad Fatehi Peykani

ISBN:978-600-205-169-1

علوم رایانه
بابل، خیابان شریعتی، مجتمع میلاد، واحد ۱۷
تلفن: ۳۲۳۶۰۷۷۲-۰۱۱ فکس: ۳۲۳۲۰۶۶۷-۰۱۱
شابک: ۱-۱۶۹-۲۰۵-۶۰۰-۹۷۸
www.nourayandeh.net

داده‌کاوی با پایتون

به همراه تحلیل شبکه‌های اجتماعی

نویسندگان :

بابک تیمورپور

استادیار مهندسی فناوری اطلاعات، دانشگاه تربیت مدرس

وحید یادگاری

محمد فاتحی

مرداد ۱۳۹۹

فهرست

فصل دوم: مقدمه ای بر پایتون

۱-۱ مقدمات و توابع ابتدایی

پایتون[۱]، زبانی با یادگیری آسان محسوب می‌شود و از همین رو بسیاری از برنامه‌نویس‌های تازه‌کار آن را به عنوان اولین زبان برنامه‌نویسی خود برمی‌گزینند، زیرا پایتون به عنوان یک زبان همه‌منظوره[۲] ساخته و توسعه داده شده و محدود به توسعه نوع خاصی از نرم‌افزارها نیست. به بیان دیگر، می‌توان از آن برای هر کاری، از تحلیل داده[۳] گرفته تا ساخت بازی‌های کامپیوتری استفاده کرد.

همچنین، پایتون در میان جوامع علمی از محبوبیت فوق‌العاده‌ای برخوردار است، زیرا از آن برای محاسبه معادلات پیچیده و تحلیل‌های داده استفاده می‌شود.

پایتون یک زبان برنامه‌نویسی شی‌گرا[۴] و سطح بالا[۵] با معناشناسی[۶] پویای یکپارچه شده برای وب و ساخت و توسعه نرم‌افزارهای کاربردی[۷] است.

---

۱. Python
۲. General-Purpose Language
۳. Data Analysis
۴. Object Oriented
۵. High-Level
۶. Semantic
۷. Application software

خواندن و ترجمه کدهای نوشته شده به زبان برنامه‌نویسی پایتون نسبت به دیگر زبان‌ها برای توسعه‌دهندگان ساده‌تر محسوب می‌شود. این موضوع به نوبه خود هزینه‌های نگهداری و توسعه برنامه‌های نوشته شده به این زبان را کاهش می‌دهد زیرا امکان همکاری تیم‌ها بدون مواجهه با موانع زبانی و وجود تجربیات کاری متفاوت در میان اعضای تیم را به دست می‌دهد.

علاوه بر این، پایتون از ماژول‌ها[۱] و بسته‌ها[۲] استفاده می‌کند، بدین معنا که برنامه‌های این زبان قابل طراحی به سبک ماژولار[۳] هستند و کدهای نوشته شده در یک پروژه در پروژه‌های گوناگون دیگر نیز قابل استفاده مجدد محسوب می‌شوند. هنگامی که کاربری ماژول یا بسته مورد نیاز خود را توسعه داد، خودش یا دیگر علاقمندان (در صورتی که کد در اختیار عموم قرار بگیرد) می‌توانند آن را برای استفاده در دیگر پروژه‌ها گسترش دهند. ایمپورت[۴] و اکسپورت[۵] کردن این ماژول‌ها نیز کار آسانی است.

یکی از قابل توجه‌ترین مزایای زبان برنامه‌نویسی پایتون آن است که کتابخانه[۶] و مفسر استاندارد[۷] آن، هم به صورت دودویی[۸] و هم منبع[۹] به رایگان در دسترس همگان قرار دارند. در پایتون هیچ انحصاری وجود ندارد، زیرا همه ابزارهای لازم برای آن در کلیه پلتفرم‌های اصلی[۱۰] موجود هستند. بنابراین، پایتون برای توسعه‌دهندگانی که نمی‌خواهند دغدغه هزینه‌های بالای توسعه را داشته باشند گزینه‌ای جذاب به شمار می‌آید.

---

۱. modules
۲. packages
۳. modular
۴. Import
۵. Export
۶. Library
۷. Standard Interpreter
۸. binary
۹. source
۱۰. Main Platforms

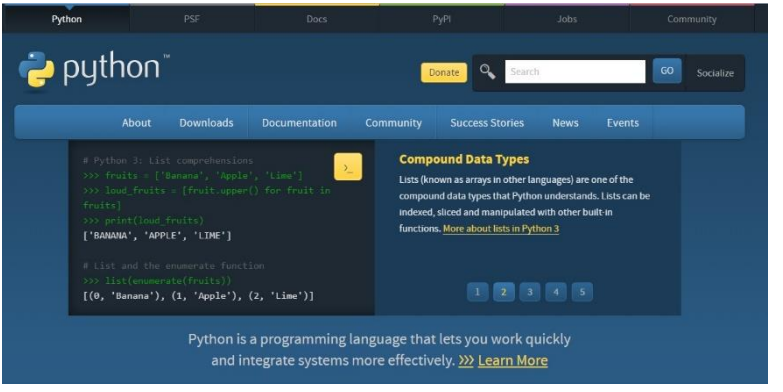
۲-۱ مزایای منحصربه‌فرد پایتون

یک نرم‌افزار متن‌باز[۱] و رایگان است. این زبان دارای ویژگی‌های زیادی است که در ادامه به برخی از آنها اشاره می‌شود:

قابلیت‌های فنی :

* کاربری و نگهداری داده‌ها به‌صورت مفید و مؤثر
* دارا بودن بسیاری از عملگرهای لازم برای محاسبات ماتریسی و آرایه‌ای
* دارا بودن مجموعه کاملی از ابزار تجزیه و تحلیل داده‌ها
* امکانات گرافیکی منحصربه‌فرد برای تحلیل داده‌ها و نمایش آن‌ها
* زبان برنامه‌نویسی ساده با قابلیت‌های بروز رسانی بالا

برای دانلود پایتون می‌توان به سایت آن مراجعه نمود. (https://www.python.org)



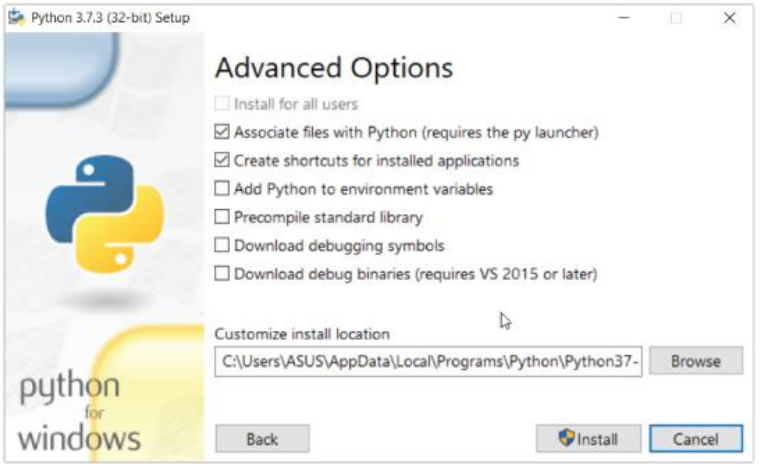
شکل ۱-۲-۲- وب‌سایت پایتون

با اجرای فایل نصاب[۲] و دنبال کردن گام‌ها این برنامه قابل نصب است. در طول فرایند نصب، باید گزینه «Add Python to environment variables»

۱. Open Source
۲. exe

را انتخاب کرد. این کار، پایتون را به متغیرهای محیطی اضافه می‌کند و کاربر را قادر می‌سازد تا زبان برنامه‌نویسی پایتون را از هر بخشی از کامپیوتر اجرا کند.

همچنین، می‌توان مسیری که پایتون در آن نصب می‌شود را انتخاب کرد.



شکل ۲-۲-۲- فرایند نصب پایتون

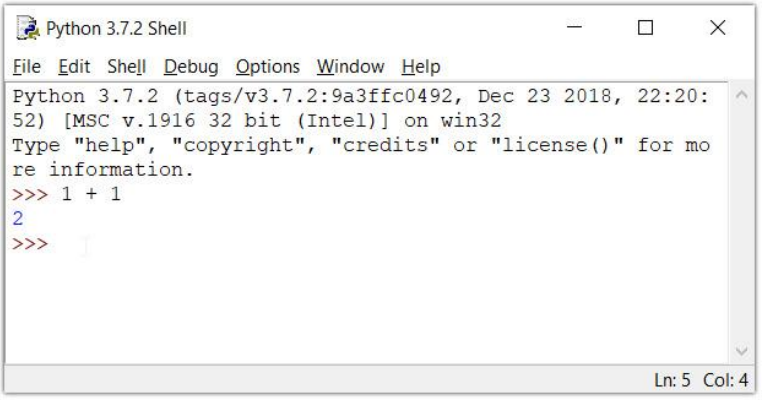
پس از پایان یافتن نصب پایتون، می‌توان آن را اجرا کرد. با نوشتن عبارت python در خط فرمان، مفسر پایتون فورا فراخوانی می‌شود. برای برنامه‌نویسی به زبان پایتون، می‌توان کد را مستقیما در Python تایپ کرد و «دکمه ورود» (کلید انتر|Enter Key) را برای دریافت خروجی فشرد. با نوشتن ۱+۱ و فشردن دکمه ورود، خروجی ۲ نمایش داده می‌شود. این دستور را می‌توان به عنوان ماشین حساب استفاده کرد. برای خروج از این حالت، باید دستور quit() را نوشت و دکمه ورود را فشرد.

شکل ۳-۲-۲- اجرای پایتون در خط فرمان

اجرای پایتون در محیط توسعه یکپارچه

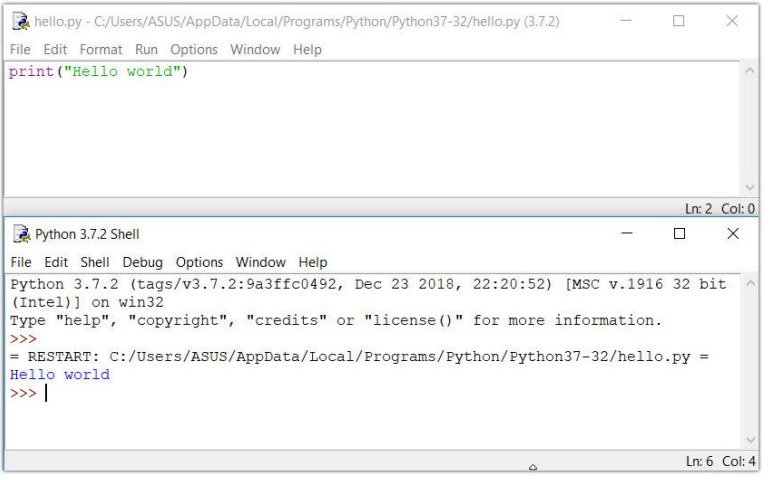
می‌توان از هر نرم‌افزار ویرایش متنی برای نوشتن فایل اسکریپت پایتون استفاده کرد. در این راستا، تنها نیاز به ذخیره کردن کد نوشته شده در ویرایش‌گر با پسوند py. است. اما، استفاده از یک IDE می‌تواند کار برنامه‌نویسی را تا حد زیادی آسان‌تر کند. IDE نرم‌افزاری است که ویژگی‌های مفیدی مانند تکمیل کد خودکار[۱]، برجسته‌سازی و بررسی نحو، جستجو در فایل و دیگر موارد را برای برنامه‌نویسان جهت ساخت و توسعه نرم‌افزار فراهم می‌کند. به هر حال، هنگام نصب پایتون در ویندوز و مک، یک IDE به نام آی‌دی‌ال‌ای[۲] که خود نیز به زبان پایتون نوشته شده است، نصب می‌شود. می‌توان از این محیط توسعه یکپارچه برای اجرای پایتون روی کامپیوتر استفاده کرد. این IDE برای افراد مبتدی بسیار عالی محسوب می‌شود. البته، این IDE همراه بسته‌های لینوکس عرضه نمی‌شود. هنگام باز کردن IDLE، یک شل پایتون تعاملی باز می‌شود.

۱. Code Completion
۲. IDLE



شکل ۴-۲-۲- نمایی از IDLE

اکنون می‌توان یک فایل جدید ساخت و آن را با پسوند .py ذخیره کرد. برای مثال، می‌توان فایل hello.py را ساخت. اکنون، ابتدا باید کد پایتون را در فایل نوشت. سپس، آن را ذخیره کرد. برای اجرای فایل باید به مسیر Run > Run Module رفت و یا دکمه F۵ را زد.



شکل ۵-۲-۲- اجرای برنامه در محیط IDLE

پایتون نیز مانند هر زبان مهم برنامه‌نویسی دیگری از کتابخانه‌ها و فریمورک‌های شخص ثالث پشتیبانی می‌کند. این کتابخانه‌ها در یک مخزن[۱] به نام ایندکس بسته پایتون (PyPI) قابل دسترسی هستند.

از سوی دیگر دریافت، نصب و مدیریت این بسته‌ها به صورت دستی، کاری دشوار و زمان‌بر است و به همین دلیل بسیاری از توسعه‌دهندگان پایتون، معمولاً از یک ابزار دیگر به نام PIP برای پایتون استفاده می‌کنند تا همه کارها را آسان‌تر و سریع‌تر انجام دهند.

PIP اختصار بازگشتی برای عبارت‌های «PIP بسته‌ها را نصب می‌کند» یا «برنامه نصب ترجیحی» در نظر گرفته می‌شود. PIP در واقع یک ابزار خط فرمان است که بسته‌های PyPI را نصب، حذف و یا نصب مجدد می‌کند.

نگاهی به توزیع Anaconda

آناکوندا یک توزیع آزاد و متن‌باز از زبان‌های برنامه‌نویسی پایتون و R برای انجام محاسبات علمی علم داده و یادگیری ماشین است. هدف توزیع پایتون آناکوندا ساده کردن مدیریت و استقرار بسته است. نسخه‌های بسته به وسیله سیستم مدیریت بسته کوندا[۲] مدیریت می‌شوند. توزیع آناکوندا توسط بیش از ۱۵ میلیون کاربر مورد استفاده قرار می‌گیرد و شامل بیش از ۱۵۰۰ بسته علم داده محبوب می‌شود.

توزیع پایتون آناکوندا با بیش از ۱،۵۰۰ بسته شامل بسته کوندا و مدیر محیط مجازی[۳] ارائه می‌شود. این توزیع دارای رابط کاربری گرافیکی[۴] به نام راهنمای آناکوندا[۵] است که به عنوان جایگزینی برای رابط خط فرمان[۶] محسوب می‌شود.

---

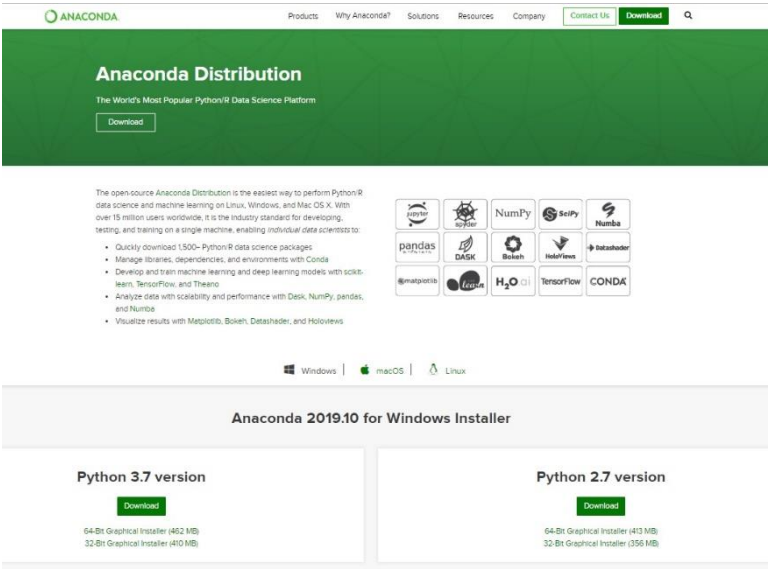
۱. Repository
۲. Conda
۳. Virtual Environment Manager
۴. Graphical User Interface
۵. Anaconda Navigator
۶. Command Line Interface

تفاوت اصلی بین مدیر بسته کوندا و pip در چگونگی مدیریت وابستگی‌های بسته‌ها است که چالشی قابل توجه برای علم داده در پایتون محسوب می‌شود و به همین دلیل، کوندا ایجاد شده است.

pip همه وابستگی‌های مورد نیاز بسته پایتون را صرف‌نظر از اینکه با دیگر بسته‌هایی که پیش از این نصب شده‌اند سازگار هستند یا نه، نصب می‌کند. بنابراین، برای مثال نصب فعال تنسورفلو[۱] ممکن است ناگهان وقتی کاربر با pip بسته متفاوت دیگری را از کتابخانه NumPy نصب می‌کند متوقف شود. بدتر آنکه ممکن است این چنین به نظر برسد که همه چیز همچنان به خوبی کار می‌کند، اما نتایج متفاوتی در فعالیت‌های علم داده و خروجی‌ها حاصل شود یا کاربر قادر به بازتولید نتایج مشابه در جای دیگری نباشد زیرا pip install به ترتیب مشابهی انجام نشده است.

کوندا محیط کنونی فرد شامل هر آنچه نصب کرده و هر محدودیت نسخه‌ای که تعیین شده (برای مثال کاربر فقط تنسورفلو نسخه ۲,۰ به بالا را می‌خواهد) را تحلیل می‌کند و تشخیص می‌دهد که چگونه وابستگی‌های ناسازگار را نصب کند و یا به کاربر می‌گوید که کار مد نظر او قابل انجام نیست. این در حالی است که pip صرفا چیزی که کاربر خواسته و هر وابستگی را نصب می‌کند، حتی اگر دیگر چیزها دچار مشکل شوند.

۱. Tensorflow

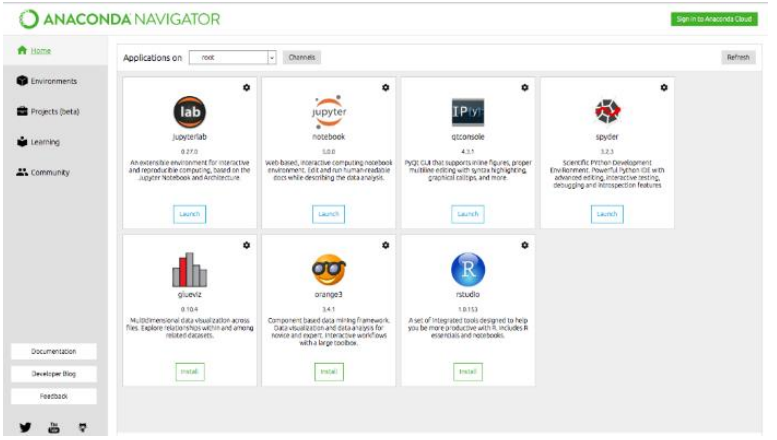


شکل ۶-۲-۲- نمایی از وب سایت آناکوندا

(https://www.anaconda.com/distribution/**)**

آناکوندا نویگیتور[۱] رابط کاربری گرافیکی دسکتاپ قرار گرفته در توزیع پایتون آناکوندا است که به کاربر امکان راه‌اندازی برنامه‌های کاربردی و مدیریت بسته‌ها، محیط‌ها و کانال‌های کوندا را بدون استفاده از دستورات خط فرمان می‌دهد. نویگیتور می‌تواند به دنبال بسته‌های جدید روی آناکوندا کلود یا مخزن محلی آناکوندا بگردد، آن‌ها را در یک محیط نصب، اجرا و به روز رسانی کند. آناکوندا نویگیتور برای ویندوز، مک‌اواس و لینوکس در دسترس است.

۱. Anaconda Navigator



شکل ۷-۲-۲- نمایی از آناکوندا نویگیتور

با استفاده از ژوپیتر نوت‌بوک[۱] به پیاده سازی کدها پرداخته شده است. ژوپیتر نوت‌بوک، امکان استفاده مشارکتی، گسترده، مقیاس‌پذیر و قابل بازتولید را فراهم می‌کند.

۳-۱ عملگرها در پایتون

عملگرها[۲] سمبل‌های خاصی در پایتون هستند که پردازش‌های حسابی و منطقی را انجام می‌دهند.

۱-۳-۱ عملگرهای حسابی

عملگرهای حسابی[۳] برای انجام پردازش‌های ریاضی مانند جمع، تفریق، ضرب و دیگر موارد استفاده می‌شود. در جدول زیر، کلیه عملگرهای حسابی موجود در پایتون ارائه و عملکرد آن‌ها همراه با مثالی شرح داده شده است.

۱. Jupyter Notebook
۲. Operators
۳. Arithmetic Operators

جدول ۱-۱-۳-۲- عملگرهای حسابی در پایتون

عملگر	شرح عملکرد
+	جمع کردن دو عمل‌وند یا عمل یگانی مثبت
-	تفریق عمل‌وند سمت راست از سمت چپی یا عمل یگانی منفی
*	ضرب دو عمل‌وند
/	تقسیم کردن عملوند سمت چپ بر سمت راستی
٪	عملیات پیمانه‌ای (محاسبه باقی‌مانده تقسیم عملوند سمت چپ بر سمت راستی)
//	خارج قسمت صحیح (این تقسیم، خارج قسمت صحیح را در خروجی ارائه می‌کند)
**	به توان y رساندن متغیر x

قطعه کد زیر را در نظر بگیرید:

```
۱ x = ۱۵
۲ y = ٤
۳ print('x + y =',x+y)
```

به دو متغیر x و y مقدار ۱۵ و ۴ تخصیص داده می‌شود. در خط سوم خروجی برنامه نمایش به صورت زیر نمایش داده می‌شود:

```
x + y = ۱۹
```

به همین ترتیب داریم:

```
۱ print('x - y =', x-y)
```

```
print('x * y =', x*y)
print('x / y =', x/y)
print('x // y =', x//y)
print('x ** y =', x**y)
```

خروجی عبارتست از:

```
x - y = ۱۱
x * y = ۶۰
x / y = ۳,۷۵
x // y = ۳
x ** y = ۵۰۶۲۵
```

۲-۳-۱ عملگرهای مقایسه

عملگرهای مقایسه[۱] برای انجام مقایسه مقادیر مورد استفاده قرار می‌گیرند. متناسب با شرط، خروجی مقایسه برابر با True یا False خواهد بود. در جدول زیر، عملگرهای مقایسه در پایتون ارائه شده‌اند.

جدول ۱-۲-۳-۲- عملگرهای مقایسه در پایتون

عملگر	شرح عملکرد
>	بزرگ‌تر است از (زمانی درست است که عمل‌وند سمت چپ بزرگ‌تر از سمت راستی باشد.)
<	کوچک‌تر است از (زمانی درست است که عمل‌وند سمت چپ کوچک‌تر از سمت راستی باشد.)
==	برابر است با (زمانی درست است که هر دو عمل‌وند برابر باشند.)
=!	نامساوی (زمانی درست است که هر دو عمل‌وند برابر نباشند.)
>=	بزرگ‌تر یا مساوی (زمانی درست است که عمل‌وند سمت چپ،

۱. Comparison Operators

	بزرگ‌تر یا مساوی سمت راستی باشد.)
<=	کوچک‌تر یا مساوی (زمانی درست است که عمل‌وند سمت چپ، کوچک‌تر یا مساوی سمت راستی باشد.)

قطعه کد زیر را در نظر بگیرید:

```
۱ x = ۵
۲ y = ۷
۳ print('x > y is',x>y)
```

خروجی قطعه کد بالا عبارتست از:

```
x > y is False
```

به همین ترتیب داریم:

```
۱ print('x < y is',x<y)
۲ print('x == y is',x==y)
۳ print('x != y is',x!=y)
۴ print('x >= y is',x>=y)
۵ print('x <= y is',x<=y)
```

خروجی عبارتست از:

```
x < y is True
x == y is False
x != y is True
x >= y is False
```

```
x <= y is True
```

۳-۳-۱ عملگرهای منطقی

عملگرهای منطقی[۱] در واقع or، and و not هستند.

جدول ۱-۳-۳-۲- عملگرهای منطقی در پایتون

عمگر	شرح عملکرد
and	در صورتی که هر دو عمل‌وند درست باشند، درست است.
or	در صورتی درست است که یکی از عمل‌وندها درست باشد.
not	در صورتی درست است که عمل‌وند غلط باشد.

قطعه کد زیر را در نظر بگیرید:

```
۱ x = True
۲ y = False
۳ print('x and y is',x and y)
```

خروجی عبارتست از:

```
x and y is False
```

به همین ترتیب داریم:

```
۱ print('x or y is',x or y)
```

۱. Logical Operators

```
۲ print('not x is',not x)
```

خروجی عبارتست از:

```
x or y is True
not x is False
```

۱-۳-۴ عملگرهای تخصیص

عملگرهای تخصیص[۱] در پایتون برای تخصیص مقدار به یک متغیر مورد استفاده قرار می‌گیرند. x = ۵ یک عملگر تخصیص ساده است که مقدار ۵ را در سمت راست به متغیر x در سمت چپ تخصیص می‌دهد. عملگرهای ترکیبی متعددی مانند x+=۵ در پایتون وجود دارند که به متغیر اضافه می‌شوند و بعدتر به طور مشابه تخصیص پیدا می‌کند. آنچه پیش‌تر بیان شد، در واقع برابر با x = x + ۵ است.

۱-۴ انواع داده در پایتون

هر مقدار[۲] در پایتون دارای یک نوع است. با توجه به اینکه در زبان برنامه‌نویسی پایتون همه چیز شی محسوب می‌شود، انواع داده در واقع کلاس هستند و متغیرها نمونه‌های (شی‌های) این کلاس محسوب می‌شوند. انواع داده مختلفی در پایتون وجود دارد. برخی از مهم‌ترین انواع داده‌ها در پایتون، در ادامه بیان شده‌اند.

۱. Assignment Operators
۲. Value

۱-۴-۱ اعداد در پایتون

اعداد صحیح[۱]، ممیز شناور[۲] و مختلط[۳] در دسته انواع عددی پایتون قرار می‌گیرند. از تابع type() برای دانستن اینکه یک متغیر یا مقدار به کدام کلاس تعلق دارد (چه نوع داده‌ای دارد)، استفاده کرد. تابع isinstance() برای بررسی این است که آیا یک شی به یک کلاس خاص تعلق دارد یا خیر.

```
۱ x = ۵
۲ print(x, "is of type", type(x))
```

خروجی قطعه کد بالا عبارتست از:

```
۵ is of type <class 'int'>
```

قطعه کد زیر را در نظر بگیرید:

```
۱ x = ۳,۱۲
۲ print(x, "is of type", type(x))
```

خروجی عبارتست از:

```
۳,۱۲ is of type <class 'float'>
```

قطعه کد زیر را در نظر بگیرید:

```
۱ x = ۱ + ۲j
۲ print(x, "is complex number?",
  isinstance(۱+۲j,complex))
```

۱. Integer
۲. Float
۳. Complex

خروجی عبارتست از:

```
(۱+۲j) is complex number? True
```

اعداد صحیح می‌توانند طول‌های مختلفی داشته باشند، این مورد تنها بر اساس میزان حافظه موجود محدود شده است. یک عدد ممیز شناور تا ۱۵ رقم اعشار صحیح است. بخش صحیح و اعشاری یک عدد ممیز شناور با نقطه ممیز از یکدیگر جدا می‌شوند. اعداد مختلط به شکل x + yj نوشته می‌شوند، که در آن x بخش صحیح و y بخش موهومی است. در ادامه، مثال‌هایی در این رابطه ارائه شده است.

```
۱ x = ۱۲۳٤٥٦۷۸۹
۲ x
```

خروجی عبارتست از:

```
۱۲۳٤٥٦۷۸۹
```

```
۱ y = ۰,۱۲۳٤٥٦۷۸۹۱۲۳٤٥٦۷۸۹
۲ y
```

خروجی عبارتست از:

```
۰,۱۲۳٤٥٦۷۸۹۱۲۳٤٥٦۷۸
```

متغیر y که از نوع float است، بریده[۱] شده است.

```
۱ z = ۱ + ۲j
```

۱. Truncated

```
۲ z
```

خروجی عبارتست از:

```
(۱ + ۲j)
```

۲-۴-۱ لیست‌ها در پایتون

لیست[۱] یک توالی دارای ترتیب از عناصر است. لیست یکی از انواع داده پرکاربرد در زبان برنامه‌نویسی پایتون است و انعطاف‌پذیری بالایی دارد. نیازی نیست که همه عناصر موجود در لیست از یک نوع باشند. اعلان یک لیست کار ساده‌ای است. عناصر لیست با استفاده از کاما از یکدیگر جدا می‌شوند و با استفاده از براکت محصور شده‌اند (در براکت قرار گرفته‌اند).

```
۱ x = [۱, ۲,۲, 'python']
```

می‌توان از عملگر برش زدن [ ] برای استخراج یک عنصر یا طیفی از عناصر از یک لیست استفاده کرد. اندیس‌ها در پایتون از ۰ آغاز می‌شوند.

```
۱ a = [۵,۱۰,۱۵,۲۰,۲۵,۳۰,۳۵,۴۰]
۲ print("a[۲] = ", a[۲])
```

خروجی عبارتست از:

```
a[۲] =  ۱۵
```

همچنین داریم:

۱. List

```
print("a[۰:۳] = ", a[۰:۳])
```

خروجی عبارتست از:

```
a[۰:۳] =  [۵, ۱۰, ۱۵]
```

همچنین داریم:

```
print("a[۵:] = ", a[۵:])
```

خروجی عبارتست از:

```
a[۵:] =  [۳۰, ۳۵, ۴۰]
```

لیست‌ها تغییر پذیر هستند. بدین معنا که مقدار عناصر یک لیست قابل جایگزینی است.

۳-۴-۱ تاپل‌ها در پایتون

تاپل[۱] یک توالی دارای ترتیب از عناصر مانند لیست است. تنها تفاوت تاپل و لیست در این است که تاپل‌ها غیر قابل تغییر هستند. تاپل‌ها پس از آنکه ساخته شدند، قابل ویرایش نیستند. تاپل‌ها برای نوشتن داده‌های محافظت شده در مقابل نوشتن و غیر قابل تغییر، مورد استفاده قرار می‌گیرند و معمولا سریع‌تر از لیست‌ها هستند چون به صورت پویا تغییر نمی‌کنند. تاپل‌ها با استفاده از پرانتز تعریف می‌شوند و عناصر آن‌ها به وسیله کاما از یکدیگر جدا می‌شوند.

```
t = (۵,'program', ۱+۳j)
```

۱. Tuple

می‌توان از عملگر برش زدن [] برای استخراج عناصر تاپل استفاده کرد، اما نمی‌توان مقادیر آن را تغییر داد.

```
۱ t = (۵,'program', ۱+۳j)
۲ print("t[۱] = ", t[۱])
```

خروجی عبارتست از:

```
t[۱] =  program
```

همچنین داریم:

```
۱ print("t[۰:۳] = ", t[۰:۳])
```

خروجی عبارتست از:

```
t[۰:۳] =  (۵, 'program', (۱+۳j))
```

۱-۴-۴ رشته‌ها در پایتون

رشته[۱] یک توالی از کاراکترهای یونیکد است. می‌توان از تک علامت نقل قول انگلیسی (سینگل کوتیشن | Single Quotation) یا (دابل کوتیشن | Double Quotation) برای نمایش رشته‌ها استفاده کرد. رشته‌های چند خطی با استفاده از سه کوتیشن ’’’ یا """ قابل اعلان شدن هستند.

```
۱ s = "This is a string"
```

۱. String

```
s = '''a multiline
```

مانند لیست و تاپل، عملگر برش‌زنی [ ] برای رشته‌ها نیز قابل استفاده است. رشته‌ها غیر قابل تغییر هستند.

```
s = 'Hello world!'
print("s[۴] = ", s[۴])
```

خروجی عبارتست از:

```
s[۴] =  o
```

همچنین داریم:

```
print("s[۶:۱۱] = ", s[۶:۱۱])
```

خروجی عبارتست از:

```
s[۶:۱۱] =  world
```

۵-۴-۱ مجموعه‌ها در پایتون

مجموعه[۱]، گروهی از عناصر فاقد ترتیب یکتا هستند. مجموعه به وسیله مقادیر درون کروشه {} که با کاما از یکدیگر جدا می‌شوند، تعریف می‌شود. عناصر مجموعه فاقد ترتیب هستند.

```
a = {۵,۲,۳,۱,۴}
print("a = ", a)
```

۱. Set

خروجی عبارتست از:

```
a ={۱, ۲, ۳, ٤, ٥}
```

همچنین داریم:

```
۱ print(type(a))
```

خروجی عبارتست از:

```
<class 'set'>
```

می‌توان عملیات مجموعه‌ها مانند اتحاد و اشتراک را روی مجموعه‌ها در پایتون اجرا کرد. مجموعه دارای مقادیر یکتا است. مقادیر تکراری از مجموعه حذف می‌شوند.

```
۱ a = {۱,۲,۲,۳,۳,۳}
۲ a
```

خروجی عبارتست از:

```
{۱, ۲, ۳}
```

از آنجا که عناصر مجموعه‌ها فاقد ترتیب هستند، اندیس‌گذاری هیچ معنایی ندارد. بنابراین، عملگر [] روی مجموعه‌ها کار نمی‌کند.

۶-۴-۱ دیکشنری در پایتون

دیکشنری[۱] مجموعه‌ای فاقد ترتیب از جفت‌های کلید-مقدار است. به طور کلی، از مجموعه‌ها زمانی استفاده می‌شود که حجم زیادی از داده‌ها وجود داشته باشد. دیکشنری‌ها برای بازیابی داده‌ها بهینه شده‌اند. برای بازیابی یک مقدار از دیکشنری، باید کلید آن را دانست. در پایتون، دیکشنری‌ها با {} تعریف می‌شوند و هر عنصر در آن به شکل key:value است. کلیدها و مقادیر می‌توانند از هر نوعی باشند.

```
۱ d = {۱:'value','key':۲}
۲ type(d)
```

خروجی عبارتست از:

```
<class 'dict'>
```

از کلیدها برای بازیابی مقادیر متناظر آن‌ها استفاده می‌شود. اما راه دیگری برای بازیابی مقادیر دیکشنری‌ها وجود ندارد.

```
۱ d = {۱:'value','key':۲}
۲ type(d)
۳ print("d[۱] = ", d[۱])
۴ print("d['key'] = ", d['key'])
```

خروجی عبارتست از:

```
<class 'dict'>
d[۱] =  value
```

۱. Dictionary

```
d['key'] =  ۲
```

۱-۴-۷ تبدیل انواع داده در پایتون

در پایتون، می‌توان انواع داده را به یکدیگر تبدیل کرد. به این کار، تبدیل نوع[۱] گفته می‌شود. می‌توان تبدیل بین انواع مختلف داده‌ها را با استفاده از توابع گوناگون تبدیل نوع، مانند int() ، float() و str() انجام داد.

```
۱ float(۵)
```

خروجی عبارتست از:

```
۵,۰
```

تبدیل نوع از float به int موجب بریدن مقدار می‌شود (آن را به صفر نزدیک‌تر می‌کند).

```
۱ int(۱۰,۶)
```

خروجی عبارتست از:

```
۱۰
```

```
۱ int(-۱۰,۶)
```

خروجی عبارتست از:

```
-۱۰
```

۱. Type Conversion

۵-۱ تابع در پایتون

تابع در پایتون گروهی از عبارت‌های مرتبط است که یک کار مشخص را انجام می‌دهند. توابع کمک می‌کنند تا برنامه به بخش‌های کوچک‌تر و دانه‌بندی شده‌ای ماژولار شکسته شود. هرچه برنامه بزرگ و بزرگ‌تر شود، تابع‌ها به سازمان‌یافته‌تر و قابل مدیریت شدن آن کمک می‌کنند. علاوه بر این، توابع مانع از تکرار برنامه‌نویسی برای یک کار واحد می‌شوند و کد را قابل استفاده مجدد می‌کنند.

تعریف یک تابع که شامل مولفه‌های زیر می‌شود:

یک. کلیدواژه def علامت آغاز سرآیند[۱] تابع است.

دو. نام تابع (function_name) که به صورت یکتا، تابع را مشخص کند.

سه .پارامترها (آرگومان‌ها) که به وسیله آن‌ها، مقادیر به تابع پاس داده می‌شوند. آرگومان‌ها اختیاری هستند.

چهار. یک علامت نقل قول یا دو نقطه (:) برای تعیین پایان هدر.

پنج. یک یا تعداد بیشتری دستور پایتون که بدنه تابع را تشکیل می‌دهند. دستورها باید دارای سطح دندانه‌گذاری یکسانی باشند (معمولا ۴ فاصله).

شش. یک دستور return اختیاری برای بازگرداندن یک مقدار از تابع.

برای مثال تابع زیر را در نظر بگیرید:

```
۱  def absolute_value(num):
۲      if num >= ۰:
۳     return num
```

۱. Header

```
  else:
    return -num
```

به همین ترتیب داریم:

```
print(absolute_value(۲))
```

خروجی عبارتست از:

```
۲
```

به همین ترتیب داریم:

```
print(absolute_value(-٤))
```

خروجی عبارتست از:

```
٤
```

توجه داریم که پایتون نسبت به سطح دندانه‌گذاری یا تورفتگی‌ها حساس است. از سوی دیگر پایتون یک زبان برنامه‌نویسی حساس به بزرگی و کوچکی حروف[۱] است. این یعنی، Variable و variable مشابه نیستند.

۶-۱ حلقه for در پایتون

حلقه for در پایتون برای تکرار کردن کاری در یک توالی (لیست، تاپل و رشته) یا دیگر اشیای قابل تکرار، مورد استفاده قرار می‌گیرد.

۱. Case Sensitive

در قطعه کد زیر، مثالی از یک حلقه for در پایتون را مشاهده می‌کنید. این حلقه مجموع اعداد موجود در لیست numbers را محاسبه می‌کند.

```
numbers = [٦, ٥, ٣, ٨, ٤, ٢, ٥, ٤, ١١]
sum = ٠
for val in numbers:
  sum = sum+val
print("The sum is", sum)
```

خروجی عبارتست از:

```
The sum is ٤٨
```

۱-۷ تابع ()range در پایتون

می‌توان یک توالی از اعداد را با استفاده از تابع ()range تولید کرد. range(۱۰) اعداد از ۰ تا ۹ را تولید می‌کند (ده عدد). همچنین، می‌توان سایز شروع، پایان و گام را به عنوان range(start,stop,step size) تعریف کرد. سایز گام به طور پیش‌فرض و در صورتی که مقدار دهی نشده باشد، برابر با یک خواهد بود. این تابع، همه مقادیر را در حافظه ذخیره نمی‌کند زیرا موجب عدم کارایی می‌شود. این در حالی است که نقطه شروع، توقف و سایز گام را به خاطر دارد و عدد بعدی را ضمن تکرار می‌سازد. برای مجبور کردن این تابع به خروجی دادن همه عناصر، می‌توان از تابع ()list استفاده کرد. مثال زیر، این موضوع را شفاف خواهد کرد.

```
print(range(١٠))
```

خروجی عبارتست از:

```
range(۰, ۱۰)
```

قطعه کد زیر را در نظر بگیرید:

```
۱ print(list(range(۱۰)))
```

خروجی عبارتست از:

```
[۰, ۱, ۲, ۳, ٤, ٥, ٦, ۷, ۸, ۹]
```

قطعه کد زیر را در نظر بگیرید:

```
۱ print(list(range(۲, ۸)))
```

خروجی عبارتست از:

```
[۲, ۳, ٤, ٥, ٦, ۷]
```

قطعه کد زیر را در نظر بگیرید:

```
۱ print(list(range(۲, ۲۰, ۳)))
```

خروجی عبارتست از:

```
[۲, ٥, ۸, ۱۱, ۱٤, ۱۷]
```

در کد بالا اعداد از ۲ شروع شده و هر ۳ گام چاپ می‌شوند.

می‌توان از تابع range() برای حلقه‌ها به منظور تکرار کردن یک توالی از اعداد استفاده کرد. این تابع را می‌توان با تابع len() برای تکرار کردن کاری در یک توالی با استفاده از اندیس‌دهی، ترکیب کرد. در ادامه، مثالی در همین رابطه ارائه شده است.

```
۱ colors = ['Red', 'Blue', 'Green']
۲ for i in range(len(colors)):
۳     print("This colir is", colors[i])
```

خروجی عبارتست از:

```
This colir is Red
This colir is Blue
This colir is Green
```

۸-۱ حلقه for با else

یک حلقه for، می‌تواند یک بلوک else انتخابی نیز داشته باشد. بخش else، در صورتی اجرا می‌شود که عناصر توالی مورد استفاده از حلقه for به پایان برسند. از عبارت break می‌توان برای متوقف کردن حلقه for نیز استفاده کرد. در چنین شرایطی، بخش else نادیده انگاشته می‌شود. بنابراین، قسمت else حلقه for، در صورتی که هیچ خطایی وجود نداشته باشد اجرا می‌شود. در ادامه، مثالی برای حلقه for همراه با else ارائه شده است.

```
۱ digits = [۰, ۱, ۵]
۲ for i in digits:
۳   print(i)
۴ else:
۵   print("No items left.")
```

خروجی عبارتست از:

```
۰
۱
۵
No items left.
```

حلقه for، عناصر لیست را تا هنگامی که حلقه متوقف شود، چاپ می‌کند. هنگامی که حلقه for متوقف شد، بلوک کد موجود در else اجرا و چاپ می‌شود.

۹-۱ تابع zip

یکی از جالب‌ترین ترفندهای برنامه نویسی در پایتون، استفاده از تابع zip() است. تابع zip، یک تابع تعبیه‌شده در زبان برنامه نویسی پایتون است. این تابع، تعدادی شیء قابل تکرار[۱] را به عنوان ورودی دریافت و لیستی از چندتایی‌ها را به عنوان خروجی تولید می‌کند. هر چندتایی، عناصر یک شیء ورودی را بر اساس شاخص مکانی آن‌ها گروه‌بندی می‌کند.

```
۱ keys = ['a', 'b', 'c']
۲ vals = [۱, ۲, ۳,]
۳ zipped = dict(zip(keys, vals))
۴ zipped
```

خروجی عبارتست از:

```
{'a': ۱, 'b': ۲, 'c': ۳}
```

۱. Iterable Objects

۱-۱۰ تابع بی نام در پایتون

در پایتون، تابع بی نام Lambda، تابعی است که بدون نام تعریف می‌شود. در حالی که توابع معمولی با استفاده از کلیدواژه def تعریف می‌شوند، تابع‌های بی نام در پایتون با استفاده از کلیدواژه lambda تعریف می‌شوند. بنابراین، به تابع بی نام در پایتون، تابع lambda نیز گفته می‌شود.

به مثال زیر توجه کنید:

```
۱ double = lambda x: x * ۲
۲ print(double(۵))
```

خروجی عبارتست از:

```
۱۰
```

تابع بالا را می‌توان به صورت زیر نوشت:

```
۱ def double(x):
۲   return x * ۲
```

۱-۱۱ بسته های پایتون

کاربران معمولا همه فایل‌های خود را در کامپیوتر در یک محل ذخیره نمی‌کنند؛ بلکه از یک سلسله مراتب برای ذخیره‌سازی آن‌ها بهره می‌برند تا دسترسی به فایل‌ها برایشان ساده‌تر باشد. در واقع، فایل‌های مشابه در پوشه[۱] مشابهی قرار می‌گیرند. برای مثال، همه آهنگ‌ها در پوشه Music نگه‌داری می‌شوند. به طور

۱. Directory

مشابه، پایتون دارای بسته‌هایی[۱] برای دایرکتوری‌ها و ماژول‌هایی برای فایل‌ها است. هر چه برنامه نوشته شده توسط کاربر بزرگ و بزرگ‌تر شود و تعداد ماژول‌های آن افزایش پیدا کند، ماژول‌های مشابه در یک بسته و ماژول‌های متفاوت در بسته‌های متفاوتی قرار می‌گیرند.

ماژول۲ به فایلی گفته می‌شود که حاوی دستورات و تعاریف پایتون است. یک فایل حاوی کدهای پایتون، برای مثال example.py، یک ماژول نامیده می‌شود؛ برای مثال بیان شده، نام ماژول «example» است. از ماژول‌ها برای شکستن برنامه‌ها به فایل‌های کوچک و قابل مدیریت استفاده می‌شود. علاوه بر آن، ماژول‌ها قابلیت استفاده مجدد از کد را فراهم می‌کنند. کاربران می‌توانند به جای کپی کردن تعاریف در برنامه‌های گوناگون، توابع پر استفاده خود را تعریف و وارد کنند (ایمپورت).

۱-۱۲ آمار توصیفی در پایتون با کتابخانه Numpy

برای استفاده از این کتابخانه ابتدا نیاز به نصب است. با استفاده از pip می‌توان آن را به صورت زیر نصب کرد:

```
pip install numpy
```

قطعه کد زیر را در نظر بگیرید:

```
import numpy as np
X = [۳۲,۳۲, ٥٦,۹۸, ۲۱,٥۲, ٤٤,۳۲, ٥٥,۶۳]
X.sort()
print(X)
```

۱. Packages
۲. Module

در قطعه کد بالا ابتدا کتابخانه numpy فراخوانی شده است. در ادامه لیست X شامل ۵ عدد تعریف شده است. با استفاده از دستور ()sort مقادیر به صورت صعودی مرتب شده است. خروجی عبارتست از:

```
[۲۱,۵۲, ۳۲,۳۲, ۴۴,۳۲, ۵۵,۶۳, ۵۶,۹۸]
```

در ادامه داریم:

```
۵  mean = np.mean(X)
۶  median = np.median(X)
۷  sd = np.std(X)
۸  variance = np.var(X)
۹  maximum=np.max(X)
۱۰ minimum = np.min(X)
۱۱ # Printing the values
۱۲ print("Average:", mean)
۱۳ print("Median:", median)
۱۴ print("Standard Deviation:", sd)
۱۵ print("Variance:", variance)
۱۶ print("Maximum:",maximum)
۱۷ print("Minimum:",minimum)
```

با اجرای قطعه کد بالا مقادیر آماری میانگین، میانه، انحراف معیار، واریانس، بیشینه و کمینه محاسبه می شود:

```
Average: ۴۲,۱۵۳۹۹۹۹۹۹۹۹۹۶
Median: ۴۴,۳۲
Standard Deviation: ۱۳,۶۲۷۷۲۱۱۵۹۴۶۰۲۲۶
```

```
Variance: ۱۸۵,۷۱۴۷۸۳۹۹۹۹۹۹۸
Maximum: ۵۶,۹۸
Minimum: ۲۱,۵۲
```

۱-۱۳ ساختار قاب داده و کتابخانه Pandas

نوع دیگری از ساختار داده‌ها که بسیار پرکاربرد می‌باشد و شامل حالت‌های ترکیبی و مختلفی از داده ها می‌باشد (شامل داده‌های عددی، کاراکترها و ...) قاب داده[۱] است. جدول زیر یک مثال از قاب داده می‌باشد که دارای ستون‌های عددی و کاراکتری می‌باشد.

جدول ۱-۱۳-۲- قاب داده

name	age	des
a	۲۰	VP
b	۲۷	CEO
c	۳۵	CFO
d	۵۵	VP
e	۱۸	MD

۱. Data Frame

قاب داده چیزی به جز یک بازنمایی درون حافظه‌ای از یک برگه اکسل در زبان برنامه‌نویسی پایتون نیست. قاب داده نیز همانند اکسل کارکردهای مختلفی مانند تحلیل، تغییر و استخراج اطلاعات ارزشمند از مجموعه داده مفروض را در اختیار می‌گذارد.

۱-۱۳-۱ چگونه یک قاب داده **Pandas** ایجاد کنیم؟

در دنیای واقعی یک قاب داده Pandas از طریق بارگذاری مجموعه داده‌ها از حافظه‌ای دائمی برای مثال از فایل‌های اکسل، csv و یا پایگاه داده ایجاد می‌شود. با این حال برای این که این مفهوم را بهتر درک کنید، در این بخش از ساختمان داده پایتون (دایرکتوری و لیست) استفاده می‌کنیم.

همان طور که در برگه اکسل فوق مشخص است، اگر نام ستون‌ها را به عنوان کلید و لیست آیتم‌های تحت هر ستون را به عنوان مقدار در نظر بگیریم، می‌توانیم به راحتی از یک دیکشنری پایتون برای نمایش آن به صورت زیر استفاده کنیم:

```
۱ my_dict = {
      'name': ["a", "b", "c", "d",
"e"],
      'age': [۲۰, ۲۷, ۳۵, ۵۵, ۱۸],
      'des':
["VP","CEO","CFO","VP","MD"]
}
```

می‌توانیم از این دیکشنری یک قاب داده Pandas به صورت زیر بسازیم:

```
۱ import Pandas as pd
```

```
۲ df = pd.DataFrame(my_dict)
۳ df
```

قاب داده حاصل می‌تواند ظاهری شبکه همان برگه اکسل که دیدیم داشته باشد:

	name	age	des
0	a	20	VP
1	b	27	CEO
2	c	35	CFO
3	d	55	VP
4	e	18	MD

شکل ۱-۱-۱۳-۲- نمایش قاب داده

این احتمال وجود دارد که ستون‌ها به همان ترتیبی که در دیکشنری هستند نباشند، چون پایتون دیکشنری را به صورت hash پیاده‌سازی می‌کند و حفظ توالی را تضمین نمی‌کند.

۱-۱۳-۲ اندیس ردیف

از آنجا که ما هیچ اندیس ردیفی برای قاب داده ارائه نکرده‌ایم، به طور خودکار توالی (۰ تا ۴) را به عنوان اندیس ردیف تولید می‌کند.

برای این که خودمان اندیس ردیف ارائه کنیم، باید پارامتر index را در تابع DataFrame(…) به صورت زیر ارسال کنیم:

```
۱ df = pd.DataFrame(my_dict,
  index=[۱,۲,۳,٤,٥])
```

این اندیس می‌تواند به صورت رشته‌ای نیز باشد.

با استفاده از دستور زیر می‌توان نوع داده‌های هر ستون در قاب داده را مشاهده کرد:

```
۱ df.dtypes
```

خروجی عبارتست از:

```
name    object
age      int٦٤
des     object
dtype: object
```

به همین ترتیب می‌توان برای هر ستون به صورت جداگانه نوع آن را مشخص کرد:

```
۱ df['age'].dtype
```

خروجی عبارتست از:

```
dtype('int٦٤')
```

ممکن است قاب داده شامل صدها و شاید هزاران ردیف باشد. برای مشاهده گزیده‌ای از ردیف‌ها می‌توانیم از تابع‌های head(...) و tail(...) استفاده کنیم که به طور پیش‌فرض ۵ ردیف اول یا آخر را ارائه می‌کند.

```
۱ df.head()
```

خروجی عبارتست از:

	name	age	des
1	a	20	VP
2	b	27	CEO
3	c	35	CFO
4	d	55	VP
5	e	18	MD

شکل ۱-۲-۱۳-۲- خروجی قاب داده

به همین ترتیب داریم:

```
۱ df.tail()
```

خروجی عبارتست از:

	name	age	des
1	a	20	VP
2	b	27	CEO
3	c	35	CFO
4	d	55	VP
5	e	18	MD

شکل ۲-۲-۱۳-۲- خروجی دستور df.tail()

برای مشاهده تعداد متفاوتی از ردیف‌ها می‌توان مقدار مورد نظر را به صورت زیر وارد کرد:

```
۱ df.head(۳)
```

خروجی عبارتست از:

	name	age	des
1	a	20	VP
2	b	27	CEO
3	c	35	CFO

شکل ۳-۲-۱۳-۲- خروجی دستور df.head(۳)

به همین ترتیب برای دستور tail(...) نیز می‌توان مقادیر متفاوت از ردیف‌ها را مشاهده کرد.

در دستورهای فوق تنها روش نمایش داده‌ها را دیدیم، حال اگر بخواهیم اندیس ردیف‌ها و نام ستون‌ها را ببینیم چه باید بکنیم؟ قاب داده تابع‌های خاصی برای مشاهده این موارد ارائه کرده است:

```
۱ df.index
```

خروجی عبارتست از:

```
Int٦٤Index([۱, ۲, ۳, ٤, ٥],
dtype='int٦٤')
```

به همین ترتیب داریم:

```
۱ df.columns
```

خروجی عبارتست از:

```
Index(['name', 'age', 'des'],
dtype='object')
```

ستون‌های قاب داده Pandas تابع‌های کمکی ستون‌های مختلف را که برای استخراج اطلاعات ارزشمند از ستون‌ها بسیار مفید هستند ارائه می‌کند.

تابع unique() عناصر منحصر به فرد یک ستون را با حذف موارد تکراری ارائه می‌کند.

```
۱ df.des.unique()
```

خروجی عبارتست از:

```
array(['VP', 'CEO', 'CFO', 'MD'],
dtype=object)
```

تابع mean() مقدار میانگین همه آیتم‌های موجود در یک ستون را نشان می‌دهد.

```
۱ df.age.mean()
```

خروجی عبارتست از:

```
۳۱,۰
```

برای حذف یک ستون به صورت زیر می‌توان عمل کرد:

```
۱ df۱ = df
۲ del df۱['name']
۳ df۱
```

همچنین با استفاده از تابع drop می‌توانیم ستون‌ها و ردیف‌ها را حذف کنیم. اعداد ۰ تا ۱۱ به صورت ۳ ردیف و ۴ صورت در قاب داده تعریف می‌شوند:

```
۱ df = pd.DataFrame
(np.arange(۱۲).reshape(۳,٤),

columns=['A','B','C','D'])
```

خروجی عبارتست از:

	A	B	C	D
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

شکل ۴-۲-۱۳-۲- قاب داده تولید شده

با دستور زیر دو ستون B و C حذف می‌شود:

	A	D
0	0	3
1	4	7
2	8	11

شکل ۵-۲-۱۳-۲- حذف ستون‌ها از قاب داده

دستور بالا را می‌توان به صورت df.drop(columns=[‘B’,’C’]) نیز نوشت. همچنین با استفاده از دستور زیر می‌توان یک ردیف را به کمک اندیس آن ردیف حذف کرد:

```
۱ df.drop([۰, ۱])
```

خروجی عبارتست از:

	A	B	C	D
2	8	9	10	11

شکل ۶-۲-۱۳-۲- حذف ردیف‌ها از یک قاب داده

۳-۱۳-۱ تابع describe()

با استفاده از تابع describe()، آمار توصیفی برای ستون‌های عددی ارائه می‌شود، اما ستون‌های حاوی کاراکتر توسط این تابع در نظر گرفته نمی‌شوند. در ادامه، ابتدا یک قاب داده ساخته می‌شود که در آن، اسامی دانش‌آموزان و رتبه آن‌ها در ریاضیات (Math) و هنر (Art) نمایش داده شده است.

```
۱ data = {
      'Name': ['Ali', 'Hossein',
'Karim', 'Javad'],
      'Art': [۱۸, ۲۰, ۱۶, ۱۵],
      'Maths': [۱۳, ۱۵, ۱۹, ۱۷]
}
۲ df = pd.DataFrame(data)
۳ df
```

خروجی عبارتست از:

	Name	Art	Maths
0	Ali	18	13
1	Hossein	20	15
2	Karim	16	19
3	Javad	15	17

شکل ۱-۳-۱۳-۲- قاب داده نمرات دانش‌آموزان

با فراخوانی تابع describe() روی قاب داده سنجه‌های گوناگون مانند میانگین، انحراف معیار، میانه، عنصر بیشینه، عنصر کمینه و دیگر موارد محاسبه می‌شود:

```
۱ df.describe()
```

خروجی عبارتست از:

	Art	Maths
count	4.000000	4.000000
mean	17.250000	16.000000
std	2.217356	2.581989
min	15.000000	13.000000
25%	15.750000	14.500000
50%	17.000000	16.000000
75%	18.500000	17.500000
max	20.000000	19.000000

شکل ۲-۳-۱۳-۲- خروجی تابع describe()

۴-۱۳-۱ ادغام دو قاب داده

کتابخانه پانداس این امکان را برای کاربر فراهم می‌کند که اشیای قاب داده را با تابع merge() به یکدیگر متصل کنند. در ادامه، دو قاب داده ساخته و روش ادغام کردن آن‌ها با یکدیگر نمایش داده شده است.

```
۱ d = {
      'subject_id': ['۱', '۲', '۳',
'٤', '٥'],
      'student_name': ['Ali',
'Hassan', 'Mahdi', 'Karim',
'Javad']
}
۲ df۱ = pd.DataFrame(d,
columns=['subject_id',
'student_name'])
۳ df۱
```

خروجی عبارتست از:

	subject_id	student_name
0	1	Ali
1	2	Hassan
2	3	Mahdi
3	4	Karim
4	5	Javad

شکل ۱-۴-۱۳-۲- خروجی قاب داده اول

حال قاب داده دوم را تشکیل می‌دهیم:

```
۱ data = {
      'subject_id': ['٤', '٥', '٦',
  '٧', '٨'],
      'student_name': ['Rahimi',
  'Naderi', 'Ahmadi', 'Naseri',
  'Heydari']
  }
۲ df۲ = pd.DataFrame(data,
  columns=['subject_id',
  'student_name'])
۳ df۲
```

خروجی عبارتست از:

	subject_id	student_name
0	4	Rahimi
1	5	Naderi
2	6	Ahmadi
3	7	Naseri
4	8	Heydari

شکل ۲-۴-۱۳-۲- خروجی قاب داده دوم

اکنون، نیاز به ادغام دو قاب داده در امتداد مقادیر subject_id است. در اینجا، به سادگی تابع merge() به صورتی که در زیر نشان داده شده فراخوانی می‌شود.

```
۱ pd.merge(df۱, df۲,
  on='subject_id')
```

خروجی عبارتست از:

	subject_id	student_name_x	student_name_y
0	4	Karim	Rahimi
1	5	Javad	Naderi

شکل ۳-۴-۱۳-۲- خروجی قاب داده حاصل از ادغام

کاری که merging انجام می‌دهد، آن است که سطرها از هر دو قاب داده را با مقادیر یکسان برای ستون‌هایی که کاربر برای ادغام استفاده می‌کند، باز می‌گرداند.

۵-۱۳-۱ تابع **value_counts()**

یکی دیگر از توابع مهم و پرکاربرد در پانداس تابع value_counts() است. این تابع تعداد مقادیر یکتا را برای یک ستون خاص باز می‌گرداند.

```
۱ df = pd.DataFrame([(۱, 'Germany'),
                     (۲, 'Iran'),
                     (۳,
'Indonesia'),
                     (٤, 'Iran'),
                     (٥, 'Iran'),
                     (٦, 'Germany'),
                     (۷, 'UK'),
                     ],
                    columns=['id',
'country']
                    )
۲ df
```

خروجی عبارتست از:

	id	country
0	1	Germany
1	2	Iran
2	3	Indonesia
3	4	Iran
4	5	Iran
5	6	Germany
6	7	UK

شکل ۱-۵-۱۳-۲- خروجی قاب داده

با استفاده از تابع value_counts() داریم:

```
۱ df['country'].value_counts()
```

خروجی عبارتست از:

```
Iran          ۳
Germany       ۲
Indonesia     ۱
UK            ۱
Name: country, dtype: int۶۴
```

۱-۱۴ مدیریت مقادیر گمشده در Pandas

قاب داده زیر را در نظر بگیرید:

```
۱ import pandas as pd
۲ import numpy as np
۳ df =
  pd.DataFrame([[np.nan,۲,۱,np.nan],
                [۲,np.nan,۳,۴],
                [۴,np.nan,np.nan,۳],

  [np.nan,۲,۱,np.nan]],columns=list('AB
  CD'))
```

خروجی قاب داده را در شکل زیر مشاهده می کنید:

	A	B	C	D
0	NaN	2.0	1.0	NaN
1	2.0	NaN	3.0	4.0
2	4.0	NaN	NaN	3.0
3	NaN	2.0	1.0	NaN

شکل ۱-۱۴-۲- خروجی قاب داده

این قاب داده شامل مقادیر گمشده است. با استفاده از دستور زیر می‌توان تعداد مقادیر گمشده در هر ستون را محاسبه کرد:

```
۱ df.isnull().sum()
```

خروجی عبارتست از:

```
A     ۲
B     ۲
C     ۱
D     ۲
dtype: int۶۴
```

با دستور زیر می‌توان مقادیر گمشده را با مقدار ۰ جایگزین کرد:

```
۱ df.fillna(۰,۰)
```

خروجی عبارتست از:

	A	B	C	D
0	0.0	2.0	1.0	0.0
1	2.0	0.0	3.0	4.0
2	4.0	0.0	0.0	3.0
3	0.0	2.0	1.0	0.0

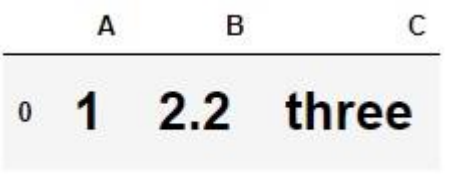
شکل ۲-۱۴-۲- جایگزینی مقادیر گمشده در قاب داده

دستور پرکاربرد بعدی در کتابخانه پانداس، دستور select_dtypes() است.

قاب داده زیر را در نظر بگیرید:

```
۱ df = pd.DataFrame([[۱, ۲,۲,
'three']], columns=['A', 'B',
'C'])
```

خروجی عبارتست از:



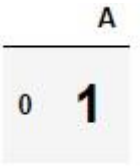
	A	B	C
0	1	2.2	three

شکل ۳-۱۴-۲- خروجی قاب داده

حال با استفاده از دستور بالا می‌توان ستون با نوع خاص داده را انتخاب کرد:

```
۱ df.select_dtypes(include=['int٦٤'])
```

داریم:



	A
0	1

شکل ۴-۱۴-۲- خروجی قاب داده با استفاده از select_dtypes()

۱۵-۱ کار با داده و فایل های اکسل در پایتون

برای بارگذاری مجموعه داده ای می توانید از تابع read_csv() در Pandas استفاده کنید. این تابع یک DataFrame بر می گرداند که می توانید بلافاصله شروع به خلاصه سازی و رسم کنید.

```
۱ import pandas as pd
۲ df =
pd.read_csv('../input/spam.csv')
```

ده سطر ابتدای داده ها را با دستور زیر مشاهده می نماییم:

```
١ df.head(n=١٠)
```

خروجی عبارتست از:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN

شکل ۱-۱۵-۲- نمایش ده سطر ابتدایی مجموعه داده

۱-۱۶ مصورسازی داده ها در زبان پایتون

زبان برنامه نویسی پایتون قدرت بسیار بالایی در پردازش داده ها دارد و همین امر باعث شده یکی از ابزارهای اصلی کابران و متخصصین حوزه علم داده به شمار رود. بسته Matplotlib اولین بسته مصورسازی داده‌ها بوسیله زبان پایتون می‌باشد که بسیار رایج و پرکاربرد است.

با کمک مصورسازی داده می‌توان الگوهای موجود در داده ها را شناسایی کرد و مصور سازی این داده‌ها راهی بهینه و مناسب برای کمک به درک و شناسایی الگوهای موجود در داده‌ها و انتقال مفاهیم است و همچنین فرآیند تحلیل و تصمیم‌گیری مبتنی بر اطلاعات را راحت‌تر می‌کند.

۱-۱۶-۱ نمودار میله ای

زمانی که بخواهیم نحوه تغییرات یک یا چند کمیت در طول زمان یا ... به نمایش بکشیم می‌توان از این نوع نمودار استفاده کرد. برای رسم این نمودار فقط کافی است داده های محور x و محور y را مشخص کنیم.

مثال زیر را در نظر بگیرید. ابتدا کتابخانه های زیر را بارگزاری می کنیم:

```
۱ import matplotlib.pyplot as plt
۲ import pandas as pd
```

در این مرحله داده ها را فراخوانی می نماییم:

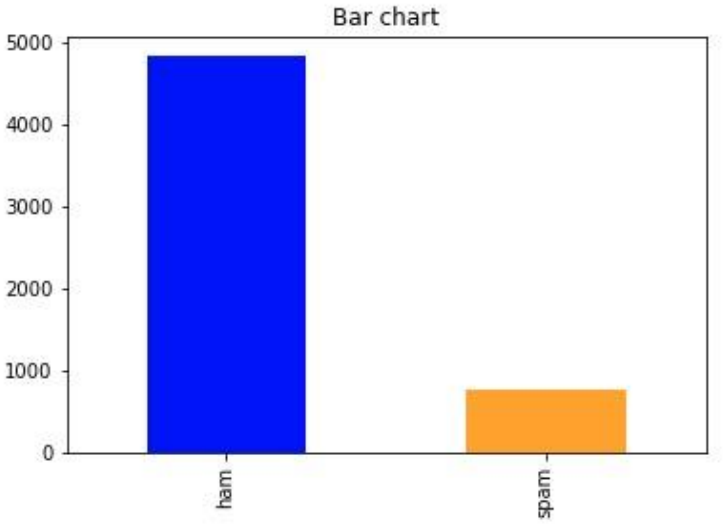
```
۱ data =
  pd.read_csv('../input/spam.csv',
  encoding='latin-۱')
```

در مجموعه داده توزیع نامه های spam و non-spam را مشاهده می کنیم:

```
۱ count_Class=pd.value_counts(data["v۱"
  ], sort= True)
۲ count_Class.plot(kind= 'bar', color=
  ["blue", "orange"])
۳ plt.title('Bar chart')
۴ plt.show()
```

نوع نمودار را bar معرفی می کنیم و مقادیر spam و non-spam یا ham را با رنگ های آبی و نارنجی مشخص می کنیم.

خروجی عبارتست از:

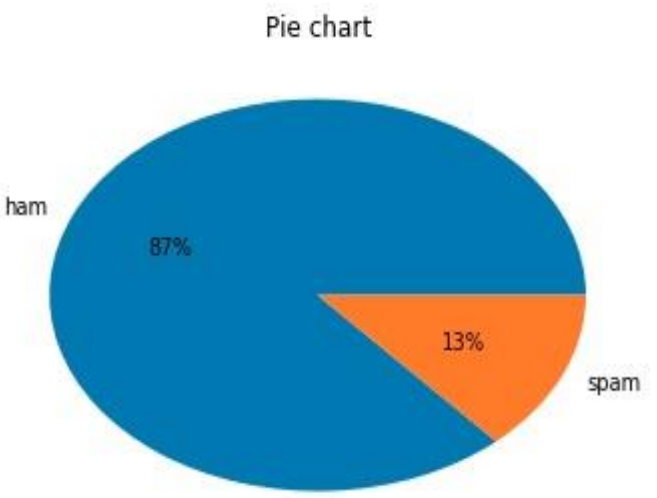


نمودار ۱-۱۶-۲- نمودار میله‌ای

نمودار دایره ای هم به صورت زیر قابل مشاهده است:

```
۱ count_Class.plot(kind = 'pie',
  autopct='%۱,۰f%%')
۲ plt.title('Pie chart')
۳ plt.ylabel('')
۴ plt.show()
```

نوع نمودار را دایره ای (pie) معرفی می کنیم.



نمودار ۲-۱۶-۲- نمودار دایره ای

۱-۱۷ کتابخانه Scikit-learn

کتابخانه Scikit-learn برای کاربردهای یادگیری ماشین از جمله خوشه بندی، رگرسیون و دسته بندی ابزار قدرتمند و در عین حال ساده ای است. یادگیری استفاده از آن بسیار سریع و آسان بوده و انعطاف آن باعث می شود تا بتوان آن را در مسائل متنوعی مورد استفاده قرار داد. در فصل‌های پیش رو الگوریتم های بسیاری از این کتابخانه مثال زده شده است.