

Hossein Seilani

Designer, Developer, SysAdmin

Complete Reference Books

2025

<https://predator-store.ir/>

Book 2\100



1001 Python Practice Examples ^{1st} Edition

**Complete
Python
Examples
Reference**

Introduction

This book is a complete, practical, and well-structured guide to learning the Python programming language. Whether you're a beginner taking your first steps or an experienced developer looking to strengthen your Python skills, this book is designed to walk you through every essential concept — from the basics to more advanced topics — in a step-by-step manner. Rather than simply listing commands and syntax, the book focuses on real understanding through clear explanations, practical examples, helpful tips, and hands-on exercises. It begins by exploring the history of Python, the reasons behind its creation, and the unique features that make Python one of the most popular languages in the world today.

ہم تعالیٰ

۱۰۰۱ مثال تمرینی پایتون

1001 Python Practice Examples

یادگیری پایتون با مثال

نویسندگان

حسین سیلانی

شهریور ۱۴۰۴

تقیم به تو
یگانه فرزانه دلم

پیشگفتار مولفان

پایتون نمونه ای از یک زبان همه کاره است. این زبان با امتیاز برنامه نویسان در حال حاضر زبان چهارم در جهان است. نکته جالبتر در مورد پایتون این است که به راحتی می‌توانید یک برنامه را در یک پلت‌فرم بنویسید و آن را در هر پلت‌فرم دیگر استفاده کنید. پایتون امکان استقلال را در سایر پلت‌فرم‌ها برای برنامه نویسان فراهم کرده است. قدرت پایتون علاوه بر قابلیت خواندن کد و نحو مختصر، این اجازه را به برنامه نویس می‌دهد که برنامه‌های کاربردی را با استفاده از خطوط کمتری بنویسند. بعضی افراد پایتون را به عنوان یک زبان اسکریپت می‌شناسند، درحالی که پایتون چیزی فراتر از آن است. این کتاب در سطح مقدماتی و پیش نیاز سایر مباحث کاربردی پایتون نوشته شده است. تا حد امکان از مثال‌های فراوان برای آموزش بهتر مطالب استفاده شده است.

شهریور ۱۴۰۴

پیشنهادهای و انتقادات خود را از طریق ایمیل زیر با مولف در میان بگذارید:

hosseinseilany@gmail.com





سرشناسه	سیلانی، حسین، ۱۳۶۰.
عنوان و نام پدیدآور	برنامه نویسی
مشخصات نشر	حسین سیلانی؛ ویراستاری حسین سیلانی.
مشخصات ظاهری	تهران: کیان دانش، ۱۴۰۴.
شابک	۲۵۰ ص.
وضعیت فهرست نویسی	۹۷۸، ۶۲۲، ۳۰۱، ۰۴۵، ۶
موضوع	فیپا
موضوع	برنامه نویسی پایتون
شناسه افزوده	برنامه نویسی (کامپیوتر)
رده بندی کنگره	سیلانی، حسین، ۱۳۶۰.
رده بندی دیویی	۷۶/۷۶QA
شماره کتابشناسی ملی	۴۳۲/۰۰۵
اطلاعات رکورد کتابشناسی	۹۶۷۳۵۹۳
	فیپا

نشر کیان دانش: ناشر کتاب‌های دانشگاهی



عنوان: ۱۰۰۱ مثال تمرینی پایتون

تالیف: حسین سیلانی

صفحه آرای و ویراستاری: حسین سیلانی

چاپ و صحافی: موسسه مهرداد

طراحی جلد: حسین سیلانی

چاپ اول: ۱۴۰۴

شمارگان: ۲۰۰ جلد

شابک: ۶، ۰۵۵، ۴۰۱، ۶۲۲، ۷۸۶

حق چاپ محفوظ است.

خیابان ۱۲ فروردین، خیابان شهدای ژاندارمری، شماره ۱۲۶، طبقه چهارم

فهرست مطالب

پیشگفتار مولفان	۵
فصل اول : معرفی زبان پایتون	۱۱
فصل اول : معرفی زبان پایتون	۱۱
تاریخچه زبان	۱۲
علت اصلی ایجاد پایتون	۱۲
ویژگی‌های پایتون	۱۲
سایرو ویژگی‌ها :	۱۲
انواع داده	۱۴
چه کسانی از پایتون استفاده می‌کنند؟	۱۴
قابلیت‌ها	۱۶
پشتیبانی از کتابخانه‌ها	۱۶
قابلیت حمل برنامه	۱۷
فصل دوم : نصب و تنظیمات زبان پایتون	۱۸
ویژگی‌های جدید در پایتون ۳	۱۹
نصب برنامه پایتون	۱۹
نصب در یونیکس و لینوکس	۱۹
نصب پایتون در ویندوز	۲۰
اجرای برنامه پایتون در ویندوز	۲۲
اجرای برنامه پایتون از طریق Ms-Dos	۲۳
شروع برنامه نویسی	۲۳
بازکردن برنامه IDLE	۲۴
اجرای مفسر	۲۵
اجرای فایل اسکریپتی پایتون در ویندوز	۲۶
اجرای فایل اسکریپتی پایتون در لینوکس	۲۸
نکته‌های آغازین	۲۹
مفسر پایتون	۲۹

۲۹.....	ساختار زبانی.....
۳۰.....	ابزارها گرافیکی و چارچوب ها.....
۳۰.....	ابزار Python/tkinter GUIs.....
۳۰.....	ابزار wxPython GUI API.....
۳۱.....	سایر محیط های توسعه پایتون.....
۳۴.....	برنامه های طراحی و توسعه پایتون.....
۳۴.....	ابزارهای گرافیکی.....
۳۶.....	فصل سوم : برنامه نویسی در IDLE
۳۷.....	شروع برنامه نویسی.....
۳۸.....	روش مقدار دهی متغیرها.....
۳۹.....	به توان بردن یک عدد.....
۴۰.....	عملوندها.....
۴۰.....	عملوندهای ریاضی.....
۴۱.....	عملوندهای مقایسه ای.....
۴۳.....	عملوندهای منطقی.....
۴۴.....	عملوند بیتی.....
۴۴.....	عملوندهای انتصابی.....
۴۵.....	عملوندهای تشخیص.....
۴۶.....	عملوندهای عضویت.....
۴۷.....	حالت متقابل.....
۴۷.....	کلید میانبرهای برنامه.....
۴۸.....	ویژگی های IDLE
۴۹.....	درج توضیحات.....
۴۹.....	روش اول : توضیحات.....
۴۹.....	توضیحات یک خطی.....
۴۹.....	توضیحات چندخطی.....
۵۰.....	روش دوم : رشته.....
۵۰.....	برنامه نویسی در vscode

۵۰.....	برنامه نویسی پایتون در (VSCode) Visual Studio Code
۵۱.....	اجرای کد پایتون در وی اس کد:
۵۸.....	فصل چهارم : ساختار کدنویسی در پایتون
۵۸.....	فصل چهارم : ساختار کد نویسی در پایتون
۵۹.....	مقدمه
۵۹.....	ساختار زبان پایتون
۵۹.....	ارتباط اجزا
۶۱.....	نوع داده‌ها در پایتون
۶۳.....	انواع عددی
۶۴.....	عملگرهای ریاضی
۶۶.....	فصل پنجم : نوع داده عددی و رشته ای در زبان پایتون
۶۷.....	۱) نوع عددی
۶۷.....	مقدار دهی از نوع عددی
۶۸.....	مقایسه مقادیر عددی
۶۹.....	تقسیم کردن
۷۰.....	تفاوت floor , truncate
۷۱.....	دقت اعداد صحیح
۷۲.....	اعداد مختلط
۷۵.....	تبدیل مبنایها
۷۶.....	کار با توابع درونی
۷۸.....	توابع مثلثاتی
۷۹.....	توابع هذلولی
۷۹.....	عدد ثابت
۸۳.....	ساخت عدد راندام
۸۵.....	۲) نوع داده رشته ای
۸۶.....	مقدار دهی رشته‌ها
۸۷.....	رشته های ممیزی
۸۹.....	نوع داده رشته ای

تعریف رشته	۹۰
دسترسی به محتوا	۹۰
متدها درنوع داده string	۹۳
از این متد برای موقعیت دهی رشته به سمت چپ و راست استفاده می‌شود.	۹۷
ماژول textwrap	۹۸
عملگرهای رشته ای	۹۹
جمع دو رشته	۹۹
ضرب و توالی رشته	۹۹
دسترسی به کاراکترهای رشته های	۱۰۰
دسترسی با عملوند [] :	۱۰۰
تبدیل رشته ها	۱۰۱
تبدیل رشته به عدد بامتد int()	۱۰۱
تبدیل عدد به رشته با متد str()	۱۰۲
تبدیل رشته به عدد اعشاری	۱۰۲
تبدیل رشته به کد اسکی و برعکس	۱۰۲
متدهای کار بر روی رشته ها	۱۰۳
پیدا کردن رشته در جمله	۱۰۴
تقسیم کردن یک رشته به حروف	۱۰۴
فصل ششم : نوع داده لیست، تاپل، دیکشنری ومجموعه در پایتون	۱۰۶
لیست ها	۱۰۷
کار با لیست ها	۱۰۷
لیست های تودرتو	۱۰۸
دنباله‌ها	۱۰۹
جمع دو لیست	۱۰۹
تکرار کردن یک لیست	۱۱۰
کار با شاخص ها در لیست ها	۱۱۰
تغییر و جایگزینی در لیست ها	۱۱۰
جاگزین کردن همه لیست	۱۱۱

۱۱۱.....	اضافه کردن ابتدای لیست
۱۱۲.....	اضافه کردن به لیست
۱۱۲.....	حذف کردن از لیست
۱۱۳.....	حذف تمامی عناصر لیست
۱۱۴.....	حذف کردن یک عنصر
۱۱۵.....	حذف کردن از لیست با متد <code>del()</code>
۱۱۵.....	حذف همه عناصر لیست
۱۱۵.....	مرتب کردن لیست
۱۱۵.....	متد <code>count()</code>
۱۱۶.....	بسط دادن یک لیست
۱۱۶.....	معکوس کردن عناصر لیست
۱۱۷.....	دیکشنری
۱۱۷.....	ایجاد دیکشنری خالی
۱۱۷.....	ایجاد دیکشنری با مقادیر تعریف شده
۱۱۸.....	ایجاد دیکشنری تو در تو
۱۱۸.....	دسترسی به عناصر
۱۱۸.....	تغییر مقادیر دیکشنری با کلید
۱۱۸.....	اضافه کردن به دیکشنری
۱۱۹.....	حذف یک عنصر
۱۱۹.....	حذف یک عنصر از دیکشنری
۱۱۹.....	حذف کامل دیکشنری
۱۱۹.....	مشاهده نام عناصر
۱۲۰.....	متد <code>item()</code>
۱۲۰.....	دیکشنری های تو در تو
۱۲۱.....	بررسی وجود عنصر
۱۲۱.....	نوع داده تاپل
۱۲۱.....	تعریف تاپل
۱۲۱.....	ایجاد تاپل با عنصر

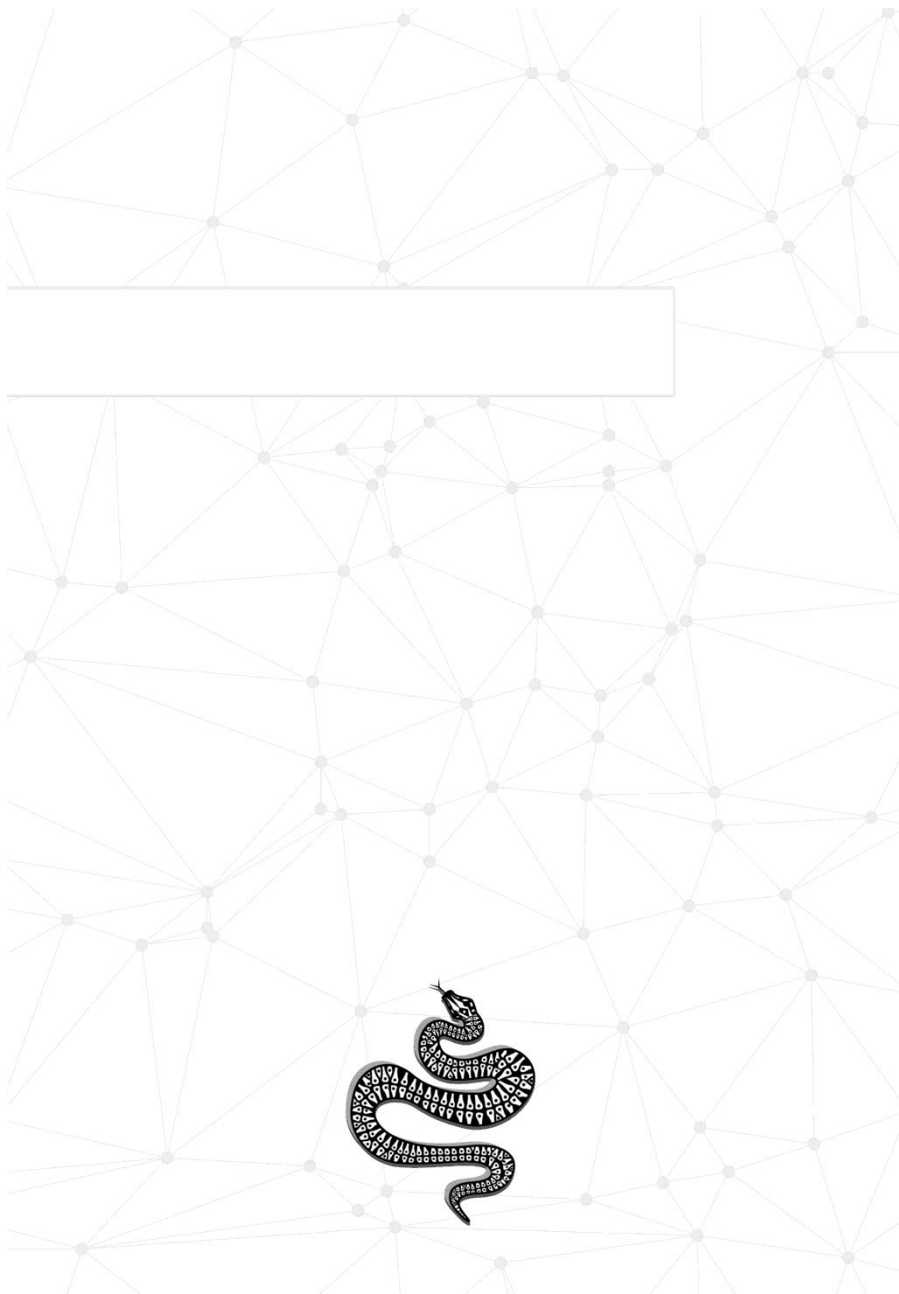
۱۲۱.....	تاپل تو در تو
۱۲۲.....	جمع دو تاپل
۱۲۲.....	تکرار تاپل
۱۲۲.....	دسترسی به عناصر تاپل
۱۲۲.....	دسترسی با دادن آدرس عناصر
۱۲۲.....	دسترسی از آخر تاپل
۱۲۳.....	مرتب کردن تاپل
۱۲۳.....	دسترسی به عناصر یک تاپل
۱۲۳.....	شمارش مقدار در تاپل
۱۲۳.....	بررسی وجود عنصر در تاپل
۱۲۳.....	شمردن عناصر در تاپل
۱۲۴.....	مرتب سازی تاپل
۱۲۵.....	فصل هفتم : ساختارهای کنترلی در زبان پایتون
۱۲۶.....	دستورات کنترلی
۱۲۶.....	ساختار if
۱۲۶.....	نوشتن شرط با دوتقطه
۱۲۷.....	مفهوم بلوک بندی
۱۲۸.....	فرمت نوشتن if
۱۲۹.....	بررسی نبود برقراری شرط
۱۲۹.....	شرط های تودرتو با if , elif
۱۳۰.....	بررسی شرط با choice
۱۳۰.....	بررسی شرط با مقادیر بولین
۱۳۱.....	ساختار while
۱۳۱.....	ساختار حلقه
۱۳۱.....	کنترل حلقه های while
۱۳۲.....	کنترل گر break
۱۳۲.....	کنترل گر else loop
۱۳۳.....	استفاده از for

۱۳۵.....	حلقه های for تو در تو
۱۳۸.....	فصل هشتم : توابع در زبان پایتون
۱۳۹.....	توابع
۱۳۹.....	انواع توابع
۱۴۰.....	شروع کار با توابع
۱۴۵.....	استفاده از def
۱۴۶.....	تعریف تابع
۱۴۶.....	فراخوانی تابع
۱۴۹.....	مقادیر پیش فرض توابع
۱۵۰.....	لیستی از آرگومان ها
۱۵۰.....	توابع داخلی پایتون
۱۶۳.....	فصل نهم : ماژول ها در زبان پایتون
۱۶۴.....	ماژول ها
۱۶۶.....	ماژول کار زمان
۱۶۷.....	متد ctime()
۱۶۷.....	سایر متدهای زمان
۱۶۷.....	دریافت اجزای تاریخ و زمان
۱۶۸.....	کار با ماژول date
۱۶۹.....	جایگزین کردن مقدار تاریخ
۱۷۱.....	ماژول logging
۱۷۱.....	نوع Log ها
۱۷۳.....	ماژول پسورد getpass
۱۷۳.....	ماژول platform
۱۷۵.....	مشاهده معماری برنامه
۱۷۵.....	صفحات وب
۱۷۵.....	بازکردن یک صفحه جداگانه
۱۷۶.....	تعیین مرورگر اختصاصی
۱۷۹.....	فصل دهم : کار با فایل ها در زبان پایتون

۱۸۰.....	نوع داده فایل
۱۸۰.....	متدهای کار با فایل
۱۸۰.....	حالت باز کردن فایل
۱۸۱.....	متد close
۱۸۱.....	کار با فایل ها
۱۸۱.....	بازکردن فایل و نوشتن در آن
۱۸۱.....	خواندن محتوای فایل با حالت r
۱۸۳.....	خواندن مقدار دلخواه از فایل
۱۸۴.....	مشاهده نام فایل
۱۸۴.....	مشاهده حالت فایل
۱۸۴.....	مشاهده یونی کد فایل
۱۸۴.....	بررسی بسته بودن فایل
۱۸۵.....	تعریف یونی کد زمان خواندن متون
۱۸۵.....	مشاهده جزئیات فایل باز شده
۱۸۵.....	فایل های باینری و متنی
۱۸۶.....	عدم وجود فایل
۱۸۷.....	خواندن فایل های فشرده
۱۸۷.....	ایجاد فایل موقت
۱۸۸.....	ساخت فایل موقت
۱۹۰.....	کار با فایل های CSV
۱۹۱.....	مآزول zip
۱۹۱.....	دریافت اندازه فایل ها
۱۹۲.....	کار با فایل فشرده gzip
۱۹۳.....	چاپ خروجی با یک اسلش
۱۹۳.....	مشاهده فایل و فهرست ها
۱۹۴.....	مشاهده فهرست یک درایو
۱۹۴.....	تغییر نام یک فهرست
۱۹۵.....	حذف یک فهرست

۱۹۵.....	متد remove()
۱۹۵.....	حذف فهرست های خالی
۱۹۵.....	متد rmdir()
۱۹۵.....	حذف فهرست های پر
۱۹۶.....	مقایسه فایل ها و فهرست ها
۱۹۶.....	تغییرمد یک فایل
۱۹۹.....	لیست فهرست ها
۱۹۹.....	بررسی وجود مسیر
۲۰۰.....	بررسی نوع ورودی
۲۰۰.....	صحت فایل بودن
۲۰۰.....	صحت فهرست بودن
۲۰۰.....	صحت لینک بودن
۲۰۱.....	سایر متد ها
۲۰۱.....	مشاهده مسیر مطلق فایل
۲۰۱.....	مشاهده نام فایل در مسیر داده شده
۲۰۱.....	مشاهده مسیر فهرست فایل
۲۰۱.....	برگرداندن مسیر فایل در حالت عادی
۲۰۲.....	بررسی مشابه بودن دو مسیر
۲۰۲.....	بدست آوردن اندازه یک فایل
۲۰۲.....	بدست آوردن
۲۰۳.....	زمان ایجاد شدن فایل
۲۰۳.....	برگرداندن نام کاربر جاری سیستم
۲۰۳.....	برگرداندن شماره ID کاربر جاری
۲۰۳.....	کار با دستورات ترمینال
۲۰۳.....	اجرای دستور و گرفتن خروجی با متد check_output
۲۰۳.....	نمایش خطوط فایل
۲۰۴.....	پردازش فایل های تصویری
۲۰۴.....	نمایش تصویر

۲۰۵.....	شارپ کردن تصویر
۲۰۵.....	بلر کردن تصویر
۲۰۶.....	روشن و تاریک کردن تصویر
۲۰۶.....	افزایش کنتراست تصویر
۲۰۷.....	دریافت اندازه تصویر
۲۰۷.....	تغییر اندازه تصویر
۲۰۷.....	چرخاندن تصویر
۲۰۸.....	قرینه کردن تصویر
۲۱۰.....	فصل یازدهم : عبارات بی قاعده در زبان پایتون
۲۱۲.....	ساختار نوشتن الگو عبارات باقاعده:
۲۱۷.....	فصل دوازدهم : کار با بانک اطلاعاتی در زبان پایتون
۲۱۸.....	بانک اطلاعاتی
۲۱۸.....	مفهوم و ساختار بانک اطلاعاتی
۲۱۸.....	بانک های sqlite
۲۱۸.....	ویژگی های بانک:
۲۲۰.....	نا محدود بودن
۲۲۰.....	اندازه ها
۲۲۱.....	داتلود و نصب
۲۲۱.....	روش نصب در لینوکس
۲۲۱.....	مفاهیم بانک اطلاعاتی
۲۲۲.....	دستورات استاندارد:
۲۲۲.....	زبان تعریف داده:
۲۲۳.....	پیوست ۱: مثال ها



تاریخچه زبان

زبان پایتون سال ۱۹۸۵-۱۹۹۰ در موسسه تحقیقاتی بین المللی هلند به نام CWI توسط جیودو ون راسوم^۱ طراحی و نام آن را از نام یک مار به نام پایتون قرار داد. زبان پایتون ترکیبی از زبان های :

ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell

همچنین پایتون تحت مجوز GPL (GNU General Public License) است.

علت اصلی ایجاد پایتون

طراح زبان پایتون در ابتدا یک زبان اسکریپتی را برای سیستم Amoeba ایجاد کرده بود. پایتون از ابتدا برای استفاده ساده و یادگیری سریع ایجاد شد. به راحتی قابل خواندن، نوشتن و فهمیدن برای یک زبان برنامه نویسی است. همچنین این زبان به صورت رایگان و متن باز^۲ ارائه شد.

ویژگی‌های پایتون

- 🔗 کاربرد عمومی^۳
- 🔗 ارتباط متقابل^۴
- 🔗 شی‌گرا^۵
- 🔗 قدرتمند برای برنامه نویسی پیشرفته
- 🔗 نگارش و تایپ ساده تر
- 🔗 پیچیدگی بسیار کم و تعداد خط کد کمتر
- 🔗 دارای کتابخانه کد بسیار وسیع
- 🔗 رفع عیب راحت تر و سریع تر
- 🔗 نزدیک به زبان محاوره سطح بالا^۶
- 🔗 زبانی قابل حمل و قابل اجرا در همه جا^۷
- 🔗 زبان کاربرپسند^۸

سایر ویژگی‌ها :

1 Guido Van Rossum

2 Open Source

3 general-purpose

4 interactive

5 object-oriented

6 high-level language

7 portable and native

8 user-friendly

قابلیت‌ها

همچنین اکثر تهدیدات علیه سازمان‌ها از طریق زبان پایتون صورت می‌گیرد. پایتون تمرکز خوبی بر روی خوانایی، کیفیت نرم افزار، انسجام نسبت به سایر زبان‌های برنامه نویسی دارد. کد پایتون برای قابلیت خواندن طراحی شده است و از این رو قابل استفاده مجدد و نگهداری بهتر نسبت به خیلی از زبان‌های برنامه نویسی سنتی را دارا است. یکنواختی کد پایتون آن را برای درک، آسان می‌کند، زبانی که نیاز به اشکال یابی debug و نگهداری کمتر است و به سرعت و مستقیم اجرا می‌شوند.

پشتیبانی از کتابخانه‌ها

پایتون با یک مجموعه بزرگی از توابع و کتابخانه‌های کد همراه است و هم چنین کتابخانه‌های استاندارد آن شناخته شده است. امروزه پایتون می‌توان آن را با کتابخانه‌های C و جاوا و ++C و دات نت ادغام کرد.

تعداد بیشماری از کتابخانه‌های کد پایتون برای دانلود وجود دارد که برنامه نویس می‌تواند برنامه‌هایی را برای:

🔧 خواندن فایل‌ها

🔧 پردازش داده‌ها

🔧 ارتباطات از طریق وب و شبکه

🔧 ساخت وب سرورها

🔧 طراحی و توسعه وب

🔧 برنامه نویسی شبکه

🔧 برنامه‌های گرافیکی

🔧 برنامه نویسی اسکریپتی برای سیستم عامل

🔧 ایجاد بازی

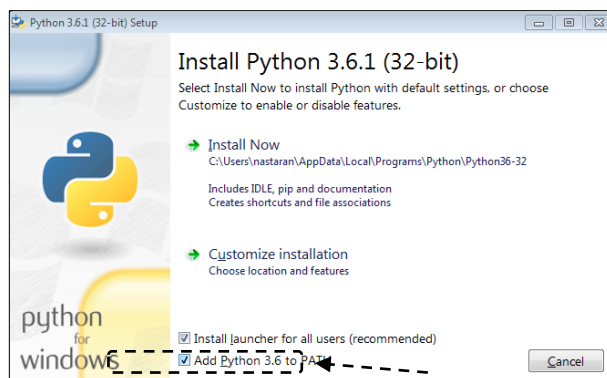
🔧 ساخت برنامه‌های تجاری

🔧 ساخت نمودارهای آماری

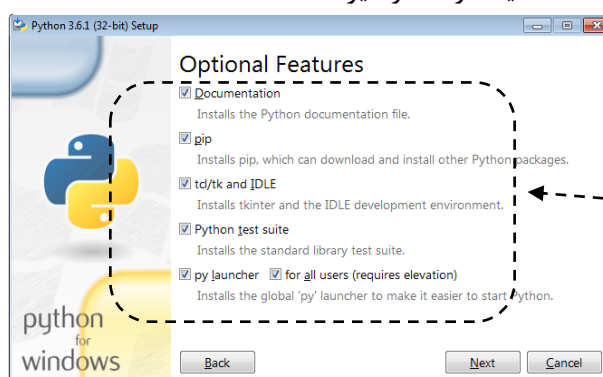
و همچنین می‌تواند با اجزای دات نت، می‌تواند مثل ارتباط با COM و رابط با دستگاه‌ها با پورت‌های سریال در شبکه‌های با رابط‌های تعامل مثل:

SOAP ، XML-RPC ، و CORBA

در پایین پنجره گزینه `add python to path` را انتخاب نمایید تا خود برنامه متغیرهای محلی را برای سیستم به صورت پیش فرض نصب و تنظیم نماید.

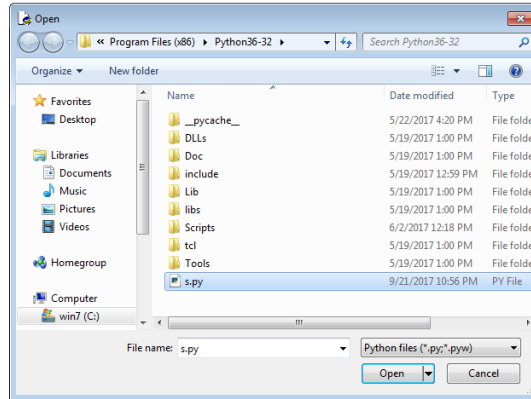


بعد انتخاب گزینه‌ها بر روی گزینه `customize installation` کلیک نمایید در پنجره باز شده بعدی گزینه‌هایی وجود دارد که می‌توان تعیین نمایید چه پارامترهایی همراه برنامه نصب شود. از قبیل: راهنما، سندها، محیط توسعه و غیره

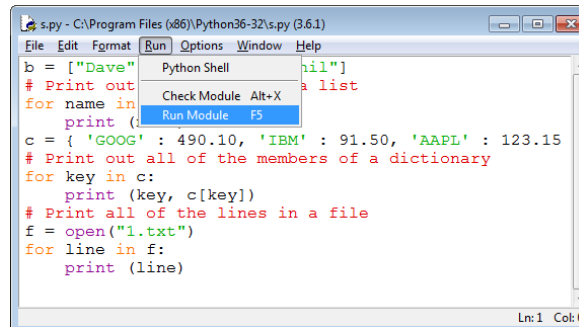


در ادامه بر روی گزینه `next` کلیک نمایید.

در پنجره باز شده بعدی گزینه‌های از قبیل نصب برای همه کاربران، اجرا شدن همه فایل‌های وابسته با پایتون با برنامه پایتون، ایجاد شدن میانبر برنامه بر روی میزکار، اضافه کردن متغیرهای محلی، کمپایل شدن کتابخانه‌های استاندارد و غیره وجود دارد که می‌توان آن‌ها را انتخاب نمایید.



پس از باز کردن آن می‌توان از طریق منوی **run** یا با فشار دادن کلید **F5** آن را اجرا کرد.



پنجره اسکرپتی دارای سیستم رنگ بندی برای قسمت‌های کد می‌باشد و به خوبی می‌توان خوانایی برنامه را مشاهده کرد.

رنگ های پیش فرض کدهای برنامه مفسر IDLE

رنگ	کاربرد	مثال
مشکی	داده و متغیر ها	ali ۶,۳
سبز	رشته ها	"test"
صورتی	توابع	Len
نارنجی	دستورات	If for else
آبی	توابع کاربر	Get_age
قرمز تیره	توضیحات	this is comments#
قرمز روشن	پیام های خطا	Syntax

محل تنظیم و تغییر رنگ دلخواه IDLE

شروع برنامه نویسی

همانگونه که می دانید یکی از اجزای زبان برنامه نویسی، متغیرها و ثوابت در آن هستند. متغیر مقادیری هستند که در طول اجرای برنامه تغییر می کنند و ثوابت در طول اجرای برنامه بدون تغییر هستند. همچنین متغیرها و ثوابت در خانه های حافظه موقت سیستم ذخیره می شوند. هر متغیر اشاره به یک مقدار در خانه های حافظه را دارد.



به کمک متغیرها می توان مقادیر خود را در خانه های حافظه برای مدت زمان دلخواه نگهداری کرد. در جدول زیر نحوه تعریف متغیر هادر سایر زبان ها و زبان پایتون نشان داده شده است. همان طور که مشاهده می کنید تعریف متغیر به سادگی هرچه تمام رخ داده است. در زبان پایتون نیاز نیست نوع متغیر ها^۱ را بیان کنید.

¹ loosely typed

مثال از عملوندهای ریاضی:

```
a = 10
b = 6
```

```
print('a + b =', a+b)
# Output: a + b = 16
```

```
print('a - b =', a-b)
# Output: a - b = 5
```

```
print('a * b =', a*b)
# Output: a * b = 50
```

```
print('a / b =', a/b)
# Output: a / b = 1.666
```

```
print('a // b =', a//b)
# Output: a // b = 1
```

```
print('a ** b =', a**b)
# Output: a ** b = 1000000
```

عملوندهای مقایسه ای

عملوندهای مقایسه ای در پایتون		
مثال	توضیح	عملوند
$x > y$	بزرگ تر از (عملگر سمت چپ اگر بزرگ باشد مقدار true برمی گرداند)	>
$x < y$	کوچک تر از (عملگر سمت چپ اگر کوچک باشد مقدار true برمی گرداند)	<
$x == y$	مساوی با (مقدار true بر می گردد، اگر هردو برابر باشند)	==
$x != y$	مساوی نیست با (مقدار true بر می گرداند، اگر مساوی نباشند)	!=
$x >= y$	بزرگ تر مساوی با (عملگر سمت چپ اگر بزرگ باشد مقدار true برمی گرداند)	>=
$x <= y$	کوچک تر مساوی با (عملگر سمت چپ اگر کوچک باشد مقدار true برمی گرداند)	<=

```
# XOR بیتی
bitwise_xor = a ^ b
print(bitwise_xor) # Output: 1011 (Binary)

# NOT بیتی
bitwise_not = ~a
print(bitwise_not) # Output: 0101 (Binary)

# Left shift
left_shift = a << 2
print(left_shift) # Output: 40 (Binary)

# Right shift
right_shift = a >> 2
print(right_shift) # Output: 2 (Binary)
```

۶. عملوندهای انتصابی:

```
Python
x = 10

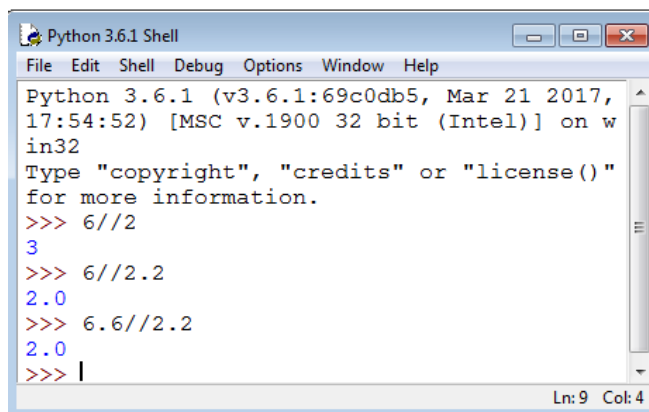
# انتساب ساده
x = 20
print(x) # Output: 20

# انتساب با جمع
x += 5
print(x) # Output: 25

# انتساب با تفریق
x -= 3
print(x) # Output: 22

# انتساب با ضرب
x *= 2
print(x) # Output: 44
```

در این نوع تقسیم خروجی به صورت تقسیم گرد شده^۱ نمایش داده می‌شود. به عبارتی آن را خلاصه و کوتاه^۲ می‌کند. البته خروجی برحسب نوع مقدار ورودی متفاوت است.



```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 6//2
3
>>> 6//2.2
2.0
>>> 6.6//2.2
2.0
>>> |
```

مثال:

```
>>> 10 / 4
2.5
>>> 10 / 4.0
2.5
>>> 10 // 4
2
>>> 10 // 4.0
2.0
```

تفاوت floor, truncate

این دو دستور برای گرد کردن خروجی اعداد به کار برده می‌شوند با این تفاوت که truncate خروجی را به مقدار پایین ترین مقدار خودگرد می‌کند که مقداری پایین تر از تابع floor است. مثلاً با تابع floor عدد منفی ۲.۵ به صورت منفی ۳ نشان داده می‌شود و با تابع truncate به صورت منفی ۲.

```
>>> import math
>>> math.floor(2.5)
2
>>> math.floor(-2.5)
-3
```

1 truncating division
2 truncate

```
import math # This will import math module

print "math.floor(-45.17) :", math.floor(-45.17)
print "math.floor(100.12) :", math.floor(100.12)
math.floor(-45.17) : -46.0
math.floor(100.12) : 100.0
```

متد **cmp()**

این متد مخفف **compare** است و برای مقایسه دو مقدار ورودی به کار می‌رود.

متد **exp**

برای محاسبه مقدار e^x از متد **exp** استفاده می‌نماییم.

```
>>>import math
>>>math.exp( x )
>>> import math
>>> math.exp(-45.17)
2.4150062132629406e-20
```

متد **fabs**

با متد **fabs()** مقدار مثبت یک عدد را بر می‌گردانیم.

```
>>>import math
>>>math.fabs( x )
>>>math.fabs(-45.17) : 45.17
>>>math.fabs(100.12) : 100.12
```

متد **log**

از این متد برای گرفتن مقدار لگاریتم یک عدد استفاده می‌شود.

```
>>>import math
>>>math.log( x )
>>> math.log(100)
4.605170185988092
>>> math.log(10)
2.302585092994046
```

توابع مثلثاتی

برای ساخت چند عدد راندوم از یک لیست ورودی، کافی است که تعداد عدد دلخواه خود را در انتهای متد وارد نمایید.

```
>>> random.sample(values, 2)
[6, 2]
>>> random.sample(values, 2)
[4, 3]
>>> random.sample(values, 3)
[4, 3, 1]
>>> random.sample(values, 3)
[5, 4, 1]
```

ایجاد لیست راندوم

گاهی یک لیست وردی داریم و قصد داریم این لیست را به صورت راندوم تغییر دهیم. این تغییر برای هربار استفاده متفاوت خواهد بود. برای این امر از متد `shuffle()` استفاده می‌کنیم.

```
>>> values = [1, 2, 3, 4, 5, 6]
>>> random.shuffle(values)
>>> values
[2, 4, 6, 5, 3, 1]
>>> random.shuffle(values)
>>> values
[3, 5, 2, 1, 6, 4]
```

متد `randint`

از این متد برای ایجاد عدد راندوم از نوع صحیح استفاده می‌شود. در این متد دامنه اعداد را تعریف می‌کنیم.

```
>>> random.randint(0,10)
2
>>> random.randint(0,10)
5
>>> random.randint(0,10)
0
>>> random.randint(0,10)
7
>>> random.randint(0,10)
10
```

متد `random`

```
youraddress= "iran"
```

تعریف رشته

برای تعریف متغیر رشته‌ای هم از تک و جفت گیومه می‌توان استفاده نمود. متغیرهای رشته‌ای در خانه‌های حافظه به صورت آدرس دهی شده قرار می‌گیرند. این خانه‌های حافظه با عدد صفر شروع می‌شوند. یعنی آدرس آفست^۱ آن از صفر است. علت این که متغیرهای رشته‌ای در خانه‌های حافظه با شماره ایندکس، شماره گذاری می‌شوند این است که بتوان به محتوای خانه‌های رشته با استفاده از شماره ایندکس خانه، دسترسی پیدا کرد.

مثال:

```
>>> myname="hossein"
```

H	o	s	s	e	i	n
0	1	2	3	4	5	6
Index(0)	Index(1)	Index(2)	Index(3)	Index(4)	Index(5)	Index(6)

دسترسی به محتوا

فرض کنید می‌خواهیم به محتوای خانه شماره ۵ دسترسی پیدا نماییم:

```
>>> myname="hossein"
>>> myname[5]
>>> i
```

دسترسی به محتوای آخرین خانه

```
>>> myname[-1]
>>> n
```

برگرداندن چند مقدار از محتوای خانه

بدین منظور کافی است با علامت: دامنه آدرس‌های ایندکس را وارد نمایید:

```
>>> myname[1:3]
>>> 'os'
```

¹ offset

```
>>> print('Updated animal list: ', animal)
Updated animal list: ['cat', 'dog', 'guinea pig']
```

علاوه بر یک مقدار، این متد عناصر تکراری را نیز از لیست حذف می‌نماید.

```
# animal list
>>> animal = ['cat', 'dog', 'dog', 'guinea pig', 'dog']

# 'dog' element is removed
>>> animal.remove('dog')

# Updated Animal List
>>> print('Updated animal list: ', animal)
Updated animal list: ['cat', 'dog', 'guinea pig', 'dog']
```

فرض کنید بخواهید حذف یک عنصر از لیست هستیم که این چنین عنصری در لیست وجود نداشته باشد. پس از استفاده از متد، برنامه مفسر خطایی را مبنی بر نبوده عنصر بر می‌گرداند.

```
# animal list
>>> animal = ['cat', 'dog', 'rabbit', 'guinea pig']

# Deleting 'fish' element
>>> animal.remove('fish')

print('Updated animal list: ', animal)
Traceback (most recent call last):
  File "...", line 5, in <module>
    animal.remove('fish')
ValueError: list.remove(x): x not in list
>>> mycar = ['reno', 'bmw', 'benz']
>>> mycar.remove('mazda')
>>> mycar
['reno', 'bmw', 'benz']
```

حذف تمامی عناصر لیست

برای حذف همه عناصر یک لیست کافی است از متد `del` استفاده کنید. بعد از متد `del` باید نام لیست را وارد نمایید.

```
# Defining a list
>>> list = [{1, 2}, ('a'), ['1.1', '2.2']]
# clearing the list
```

```
>>>del list[:]
>>>print('List:', list)
```

علاوه بر متد `del` از متد `clear` نیز به همین صورت استفاده می‌نماییم.

The `clear()` method only empties the given list. It doesn't return any value.

Defining a list

```
>>>list = [{1, 2}, ('a'), ['1.1', '2.2']]
```

clearing the list

```
>>>list.clear()
```

```
>>>print('List:', list)
```

```
List: []
```

حذف کردن یک عنصر

با متد `pop()` و با بیان موقعیت عنصر می‌توان عنصر مورد نظر را حذف کرد. توجه داشته باشید در صورتی که محل عنصر در متد `pop` بیان نشود، مقدار پیش فرض ۱- را در نظر می‌گیرد که عدد منفی یک به آخرین عنصر اشاره دارد و آن را بر می‌گرداند.

programming language list

```
>>>language = ['Python', 'Java', 'C++', 'French', 'C']
```

Return value from `pop()`

When 3 is passed

```
>>>return_value = language.pop(3)
```

```
>>>print('Return Value: ', return_value)
```

Updated List

```
>>>print('Updated List: ', language)
```

```
Return Value: French
```

```
Updated List: ['Python', 'Java', 'C++', 'C']
```

```
>>> mycar = ['reno', 'bmw', 'benz']
```

```
>>> mycar.pop(3)
```

```
>>> mycar
```

```
['reno', 'bmw', 'benz']
```

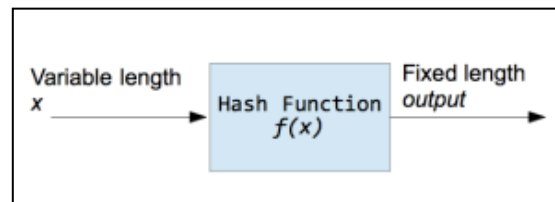
در صورتی که پارامتر وارد شده در متد `pop` در لیست وجود نداشته باشد با خطای زیر روبرو خواهید شد.

تغییر مقدار شمارشگر:

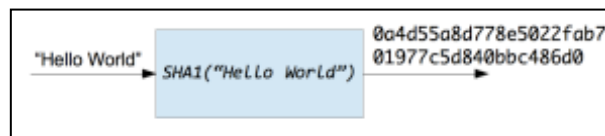
```
>>> enumerateGrocery = enumerate(grocery, 10)
>>> print(list(enumerateGrocery))
[(0, 'bread'), (1, 'milk'), (2, 'butter')]
[(10, 'bread'), (11, 'milk'), (12, 'butter')]
```

تابع `hash()`

از این تابع برای رمز کردن یک مقدار ورودی و گرفتن مقدار رمز شده در خروجی مورد استفاده قرار می‌گیرد.



به عنوان مثال :



تابع `hash` دارای یک آرگومان ورودی (شی) از نوع `integer`, `string`, `float` می‌باشد.

hash for integer

```
print('Hash for 181 is:', hash(181))
```

hash for decimal

```
print('Hash for 181.23 is:', hash(181.23))
```

hash for string

```
print('Hash for Python is:', hash('Python'))
```

Hash for 181 is: 181

Hash for 181.23 is: 530343892119126197

Hash for Python is: 2230730083538390373

```
vowels = ('a', 'e', 'i', 'o', 'u')
```

```
>>> print('The hash is:', hash(vowels))
```

The hash is: -695778075465126279

ماژول `getpass` پسورد

با کمک این ماژول می‌توان از کاربر درخواست رمز عبور کرد. درخواست رمز به گونه ای است که نیاز به داشتن نام کاربری نخواهد بود. این ماژول دارای دو متد است: متد اول در ساختار ماژول یک متن برای درخواست رمز عبور است. متد دوم در ساختار ماژول مربوط به کنترل ترمینال های مجازی `dev/tty/` در سیستم های لینوکسی می‌باشد.

```
getpass.getpass([prompt[, stream]])
```

مثال: دریافت رمز از ورودی

```
import getpass
p = getpass.getpass()
print(p)
```

مثال:

```
import getpass

p = getpass.getpass(prompt='What is your favorite color: ')
if p.lower() == 'blue':
    print('Right.')
else:
    print('bad !')
```

ماژول `platform`

با کمک این ماژول می‌توان دریافت اطلاعات سیستم را دریافت کرد. از آنجایی که پایتون یک زبان چند سکویی است و در زمان کار کردن با مفسر برنامه گاهی نیاز است که اطلاعاتی درباره برنامه پایتون داشته باشیم. این ماژول قابلیت دریافت اطلاعات در هر سیستم را داراست. این ماژول دارای ۴ تابع است:

`python_version()` و `python_version_tuple()`: این تابع برای بازیابی نسخه مفسر به کار

برده می‌شود.

`python_compiler()`: این تابع برای بازیابی کامپایلر که در مفسر به کار گرفته شده است.

`python_build()`: برای بازیابی رشته نسخه مربوط به مفسر پایتون است.

```
print("write hello word to file")
else:
    print('File not exists!')
```

خواندن فایل‌های فشرده

به منظور خواندن فایل‌های فشرده باید نام مازول همان فایل را وارد نمایید و سپس با متد `open` و متد `read` آنها را بخوانیم.

```
import gzip
with gzip.open('myfile.gz', 'rt') as f:
    text = f.read()
    print(text)
```

نوشتن در فایل

```
import gzip
with gzip.open('myfile.gz', 'wt') as f:
    f.write(text)
```

مثال:

```
import bz2
with bz2.open('myfile.bz2', 'rt') as f:
    text = f.read()
```

```
import bz2
with bz2.open('somefile.bz2', 'wt') as f:
    f.write(text)
```

```
import gzip
f = open('somefile.gz', 'rb')
with gzip.open(f, 'rt') as g:
    text = g.read()
```

ایجاد فایل موقت

منظور از موقت همان `temporary` است. فایل و فهرستی برای استفاده کوتاه مدت در زمان اجرای برنامه دلخواه ایجاد می‌شود. ابتدا مازول مربوط به ایجاد فایل موقت را فراخوانی کرده سپس یک فایل موقت ایجاد می‌نماییم. در ادامه مقادیری را در این فایل موقت قرار می‌دهیم.

```
from PIL import Image, ImageEnhance

myimg = Image.open("original-image.png")
enhancer = ImageEnhance.Sharpness(myimg)

count = 0.05
new_myimg = enhancer.enhance(count)
new_myimg.save('new-image-1.png')
```

روشن و تاریک کردن تصویر

برای روشن کردن تصویر از متد `ImageEnhance.Brightness` استفاده می‌نماییم.

```
from PIL import Image, ImageEnhance

im = Image.open("sample-image.png")
enhancer = ImageEnhance.Brightness(im)
```

تیره تر شدن تصویر

```
count = 0.5
im_output = enhancer.enhance(count)
im_output.save('darkened-image.png')
```

روشن تر شدن تصویر

```
count = 2
im_output = enhancer.enhance(count)
im_output.save('brightened-image.png')
```

افزایش کنتراست تصویر

با استفاده از متد `Contrast` می‌توان کنتراست (وضوح تصویر) را افزایش یا کاهش داد.

```
from PIL import Image, ImageEnhance

im = Image.open("sample-image.png")
enhancer = ImageEnhance.Brightness(im)
```

کاهش کنتراست

```
count = 0.5
```

```
myregex = r"({2})"
```

بین دو تا پنج مورد را مورد تطبیق قرار می‌دهد.

```
myregex = r"({2,5})"
```

دو یا موارد بیشتر مورد تطبیق قرار می‌دهد.

```
myregex = r"({2,})"
```

بیشتر از ۵ مورد تطبیق قرار می‌دهد.

```
myregex = r"({5})"
```

یکی از موارد a, b, c, d را مورد تطبیق قرار می‌دهد.

```
myregex = r"([ab-d])"
```

همه موارد به جز a, b, c, d را مورد تطبیق قرار می‌دهد.

```
myregex = r"([^\ab-d])"
```

همه موارد با a و A شروع شود تطبیق می‌دهد.

```
myregex = r"([aA] li)"
```

همه کلمات که با ali یا aly تطبیق پذیر باشد نشان می‌دهد.

```
myregex = r"(al[iy])"
```

همه اعداد بین ۰ تا ۹ را تطبیق می‌دهد.

```
myregex = r"([0-9])"
```

همه اعداد بین a تا z را تطبیق می‌دهد.

```
myregex = r"([a-z])"
```

همه اعداد بین a تا z و ۰ تا ۹ را تطبیق می‌دهد.

```
myregex = r"([a-zA-Z0-9])"
```

همه موارد را به غیر اعداد ۰ تا ۹ را تطبیق می‌دهد.

```
myregex = r"([^\0-9])"
```

تطبیق به صورت گروهی صورت می‌دهد. تطبیق بر روی al و fr انجام می‌دهد.

```
myregex = r"(al..)(fr..)"
```

یک یا موارد بیشتر مورد تطبیق قرار می‌دهد.

```
myregex = r"al +"
```

```

a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
a.sort()
print("Largest element is:",a[n-1])

```

مثال ۲۹: پیدا کردن دومین عدد بزرگ از لیست ورودی

```

a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
a.sort()
print("Second largest element is:",a[n-2])

```

مثال ۳۰: مرتب کردن یک لیست ورودی:

```

a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=input("Enter element:")
    a.append(b)
a.sort(key=len)
print(a)

```

مثال ۳۱: جایگزین کردن حرف دلخواه در یک رشته

```

string=input("Enter string:")
string=string.replace('a','$')
string=string.replace('A','$')
print("Modified string:",string)
print(string)

```

مثال ۳۲: بررسی مشابه بودن دو رشته ورودی:

```

s1=input("Enter first string:")
s2=input("Enter second string:")
if(sorted(s1)==sorted(s2)):
    print("The strings are similar.")
else:

```

برنامه مدیریت IP سیستم

نمایش IP محلی (Local IP)

نمایش IP عمومی (Public IP)

نمایش نام سیستم (Hostname)

نمایش همه‌ی آدرس‌های آی‌پی کارت‌های شبکه

```
import socket
import requests
import psutil

def get_hostname():
    return socket.gethostname()

def get_local_ip():
    return socket.gethostbyname(socket.gethostname())

def get_public_ip():
    try:
        return requests.get("https://api.ipify.org").text
    except:
        return "Unable to fetch public IP"

def get_all_ips():
    addrs = psutil.net_if_addrs()
    ips = []
    for iface, addr_list in addrs.items():
        for addr in addr_list:
            if addr.family == socket.AF_INET:
                ips.append((iface, addr.address))
    return ips

while True:
    print("\n\033[95m--- IP Management Menu ---\033[0m")
    print("1. Show Hostname")
    print("2. Show Local IP")
    print("3. Show Public IP")
    print("4. Show All Network Interfaces")
    print("5. Exit")

    choice = input("\033[93mChoose an option (1-5): \033[0m")

    if choice == "1":
```

MY PROJECTS

seilany.ir
learninghive.ir
artystone-os.ir
emperor-os.ir
predator-os.ir
little-psycho.ir

CONTACT ME

telegram: @seilany



HosseinSeilani

Designer, Developer and Linux sysadmin



I am Hossein Seilani, a Master of Science in Computer Science, and the founder and developer of Emperor-OS, Little-Psycho, and Predator-OS Linux, with extensive experience and certifications in Linux/Windows System Administration, UX/UI Design, Front-End Web Development, SEO, Graphic Design, Data Science, and Machine Learning.



Emperor-OS



Predator-OS