עודכן לאחרונה: November 1, 2026

🗺️ מפת דרכים לארבע שנים · שנה 1, אביב

על הקורס

לקורס החינמי הזה יש 9 יחידות ו-4 מעבדות. אלה המספרים האמיתיים, לא תוכנית ארוכה יותר שקוצרה.

בדיקה היא מספר או תוצאת קוד בעמוד קורס שסקריפט מחשב מחדש לפני הפרסום. המספר המודפס והמספר שחושב מחדש חייבים להתאים.

זהו מתווה של קורס חינמי ללימוד עצמי שנכתב על ידי Saving Planets. הוא אינו קורס אוניברסיטאי, אינו מעניק נקודות זכות אקדמיות, ואף מוסד לימודים אינו מעורב בו.

היקפו תואם קורסי תכנות של שנה ראשונה כמו MIT 6.100L (לשעבר 6.0001), CS50 של Harvard, CS106A של Stanford ו-CS 1371 של Georgia Tech למהנדסים. אנו מקשרים לחומרים חינמיים לאורך כל הדרך. אף אחד מהארגונים האלה אינו תומך בדף זה.

מה אנחנו מוסיפים: כל דוגמה היא קוד פייתון אמיתי שסקריפט הבנייה שלנו מריץ לפני שהדף עולה, והפלט המודפס שלה חייב להתאים בדיוק למה שאתם רואים כאן. הנתונים מגיעים ממקורות ציבוריים אמיתיים: רשומת ה-CO_2 של מאונה לואה מ-NOAA, ניתוח הטמפרטורה GISTEMP של NASA ומדי נהרות של USGS.

אין צורך להתקין דבר: השתמשו ב-Google Colab או ב-JupyterLite, שמריץ פייתון בתוך הדפדפן בעזרת Pyodide. ההתקדמות שלכם נשמרת רק בדפדפן הזה.

איך הקורס הזה עוזר להציל את כדור הארץ

מדעי האקלים והסביבה פועלים על קוד. עקומת ה-CO_2, רשומות הטמפרטורה העולמית ותחזיות הנהרות כולן נבנות על ידי תוכניות שקוראות נתונים ציבוריים, ואתם יכולים לקרוא את אותם קבצים.

כל יחידה מסתיימת בחלק קצר עם מקורות שמראה איך הרעיונות שלה עוזרים לאנשים ולכדור הארץ.

למורים

אפשר ללמד את הקורס הזה, או לתת לתלמיד לעבור אותו לבד. הוא חינם. Saving Planets לא נותנת ציון ולא נקודות זכות.

- העמוד הזה הוא כל הקורס: 9 יחידות ו-4 מעבדות (ENGR 130).
- תכננו 15 שבועות, בערך 6 עד 9 שעות בשבוע, כולל שבועות המעבדה.
- עקבו אחרי לוח 15 השבועות. קראו את היחידה, פתרו על הנייר, ואז פתחו את התשובה. בשבוע מעבדה עשו את המעבדה או את הגרסה הביתית.

- בדיקה היא מספר או תוצאת קוד בעמוד שסקריפט מחשב מחדש לפני הפרסום. המספר שרואים והמספר שחושב מחדש חייבים להתאים.
- השתמשו בהדפסת העמוד, או הורידו את מדריך הלימוד ב-PDF. דף הנוסחאות נמצא לקראת הסוף.
- שאלו: איך היחידה הזו עוזרת לאנשים ולכוכב? בכל יחידה יש תיבה ירוקה עם עובדה ומקור.
- העתיקו את תבנית המדריך למורה וכתבו את שם הכיתה, הזמן שיש לכם, והשאלות שתשאלו. פתיחת תבנית המדריך למורה

לפני שמתחילים

ידע מוקדם

- אלגברה ברמת תיכון.
- לא נדרש ניסיון בתכנות.
- פיזיקה I וחשבון אינפיניטסימלי I עוזרים ביחידת הסימולציה אבל אינם חובה.

איך ללמוד

- הקלידו כל דוגמה בעצמכם במקום להדביק; האצבעות לומדות את התחביר.
- נחשו את הפלט לפני שאתם מריצים את הקוד.
- כשנתקעים, קראו את הודעת השגיאה מלמטה למעלה.
- שנו דבר אחד בכל פעם והריצו שוב.
- פתרו כל תרגיל לפני שאתם פותחים את התשובה.
- עשו שוב את הבחנים כעבור כמה ימים. תרגול מרווח עדיף על מפגש ארוך אחד.

מה צריך

- מחשב או טאבלט עם דפדפן מודרני.
- חשבון Google חינמי ל-Colab, או JupyterLite בלי חשבון בכלל.
- אופציונלי: Python 3 מותקן מקומית עם NumPy ו-Matplotlib.

לוח זמנים מוצע ל־15 שבועות

כ־6 עד 9 שעות בשבוע: קריאה, תרגילים, ובשבועות המעבדה גם המעבדה.

שבוע	יחידה	נושאים	קריאה
1	צעדים ראשונים: משתנים, טיפוסים וביטויים	מחברות, משתנים, טיפוסים, ביטויים	מדריך פייתון §3
2	בקרת זרימה: תנאים ולולאות	תנאים ולולאות	מדריך פייתון §4
3	פונקציות	פונקציות	MIT 6.100L
4	רשימות, מילונים ומחרוזות	רשימות, מילונים, מחרוזות	מדריך פייתון §5
5	קבצים ונתוני CSV	קבצים ו-CSV	תיעוד המודול csv

שבוע	יחידה	נושאים	קריאה
6	קבצים ונתוני CSV	נתונים אמיתיים עם שגיאות; מעבדה A	NOAA GML
7	NumPy ו-Matplotlib	מערכי NumPy; תרגול לבחינת אמצע	יסודות NumPy
8	NumPy ו-Matplotlib	Matplotlib; מעבדה B	Matplotlib pyplot
9	ניפוי שגיאות ובדיקות	ניפוי שגיאות ובדיקות; מעבדה C	תיעוד pytest
10	אלגוריתמים וסיבוכיות	חיפוש וסיבוכיות	MIT 6.100L
11	אלגוריתמים וסיבוכיות	חציה ושיטות נומריות	MIT 6.100L
12	סימולציה נומרית	סימולציית קליע	פיזיקה I
13	סימולציה נומרית	מסלולים ואוכלוסיות	חשבון אינפיניטסימלי 2
14	סימולציה נומרית	גרר והתכנסות; מעבדה D	מבוא להנדסת אווירונאוטיקה וחלל
15	חזרה	חזרה וסדרת תרגול מסכמת	כל היחידות

ההתקדמות שלכם

סמנו יחידות ומעבדות כשאתם מסיימים אותן, ועשו את הבחנים. ההתקדמות והציונים נשמרים רק בדפדפן הזה ולעולם לא נשלחים אלינו.

יחידות

צעדים ראשונים: משתנים, טיפוסים וביטויים

הריצו את קוד הפייתון הראשון שלכם במחברת בדפדפן, אחסנו ערכים במשתנים, וחשבו עם מספרים וטקסט.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לפתוח מחברת Colab או JupyterLite ולהריץ תאים.
- להשתמש במשתנים ובטיפוסים int, float, str ו-bool.
- להשתמש באופרטורים אריתמטיים, כולל //, % ו-**.
- לעצב פלט בעזרת f-strings.

מושגי מפתח

- תוכנית היא רצף של הוראות. מחברת מאפשרת להריץ אותן תא אחר תא ולראות תוצאות מיד.
- משתנה הוא שם הקשור לערך: mass_kg = 1100. בחרו שמות שאומרים מה ובאילו יחידות.
- לכל ערך יש טיפוס. int (מספרים שלמים), float (עשרוניים), str (טקסט) ו-bool (True/False) הם הבסיס; type() אומר לכם איזה.
- / תמיד נותן float; // היא חלוקה שלמה; % הוא השארית; ** הוא חזקה.
- ערכי float הם קירובים בינאריים: 0.2 + 0.1 אינו בדיוק 0.3, לכן השוו ערכי float עם סבולת.

משוואות מפתח

$$W = mg$$

W=mg

דוגמה 1.1 מחשבת משקל

$$\text{change} = \frac{x_{new} - x_{old}}{x_{old}}$$

change=_new-x_oldx_old

שינוי יחסי (דוגמה 1.2)

דוגמאות פתורות שלב אחר שלב

דוגמה 1.1 חשבו את המשקל של מטוס של 1100 kg ובדקו את הטיפוסים.

```
mass_kg = 1100
g = 9.81
weight = mass_kg * g
print(weight)
print(type(mass_kg), type(weight))
```

1. mass_kg הוא int; הכפלתו ב-g float נותנת float.

פלט

```
10791.0
<class 'int'> <class 'float'>
```

תשובה: 10791 N, מטיפוס float.

דוגמה 1.2 בכמה עלה ה-CO_2 במאונה לואה מהממוצע השנתי של 1959 לזה של 2025 (NOAA)?

```
ppm_1959 = 315.98
ppm_2025 = 427.35
rise = ppm_2025 - ppm_1959
print(f"CO2 rose {rise:.2f} ppm, or {rise / ppm_1959:.1%}")
```

1. חסרו, ואז חלקו בערך ההתחלתי.
2. ה-f-string מעצבת :.2f (2 ספרות אחרי הנקודה) ו-%.1: (אחוזים).

פלט

```
CO2 rose 111.37 ppm, or 35.2%
```

תשובה: 111.37 ppm, עלייה של 35.2%.

דוגמה 1.3 חקרו אופרטורים והפתעות של float.

```
print(7 / 2, 7 // 2, 7 % 2, 2 ** 10)
print(0.1 + 0.2 == 0.3, abs((0.1 + 0.2) - 0.3) < 1e-9)
print(int("42") + 1, str(42) + "1")
```

1. 7/2 = 3.5, 7//2 = 3, 7%2 = 1.
2. 0.1 + 0.2 == 0.3 הוא False; בדיקה עם סבולת היא True.
3. int("42") ממיר טקסט למספר; str(42) + "1" משרשר טקסט.

פלט

```
3.5 3 1 1024
False True
43 421
```

תשובה: השתמשו בסבולות ל-float והמירו טיפוסים במכוון.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. מה נותן 5 % 17?

.2

2. מה הטיפוס של 2.0 * 3?

.float (6.0)

3. כתבו f-string שמדפיסה את pi עם 3 ספרות אחרי הנקודה.

.(import math אחרי) f"{math.pi:.3f}" → 3.142

4. למה mass_kg הוא שם טוב יותר מ-m?

הוא מציין את הגודל ואת היחידה שלו, וכך מונע טעויות יחידות.

תפיסות שגויות נפוצות

מיתוס: = פירושו «שווה».

בפועל: בפייתון = משייך; == משווה.

מיתוס: 0.3 == 0.2 + 0.1 הוא True.

בפועל: זה False בגלל עיגול בינארי.

מיתוס: 5 + "5" נותן 10.

בפועל: זו TypeError; המירו עם int("5").

כרטיסיות

משתנה

שם הקשור לערך.

int / float

מספר שלם / מספר עשרוני.

str

טקסט.

bool

True או False.

f-string	**//, %**
עיצוב `f"{x:.2f}"`.	חלוקה שלמה, שארית.

בוחן לבדיקה עצמית

1. 7 // 2 =

3.5

3

4

1

3. חלוקה שלמה.

2. type(2.0) הוא…

int

float

str

bool

float. יש בו נקודה עשרונית.

3. 2 ** 10 =

20

100

1024

12

1024. חזקה.

4. השמה בפייתון משתמשת ב...

==

=

only :=

<-

=. = קושר שם.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

המספרים בדוגמה 1.2 מגיעים ישירות מה-Global Monitoring Laboratory של NOAA, שמפרסמת את רשומת ה-CO_2 של מאונה לואה כקובץ CSV חינמי. כמה שורות פייתון הופכות אותה לעובדות המרכזיות של מדע האקלים.

ב־Saving Planets: חשבון אינפיניטסימלי 1

מקורות: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · Google Colab (מחברות חינמיות) (https://colab.research.google.com/)

מקורות לימוד חינמיים

- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיפים 3 ו-4.
- Google Colab (מחברות חינמיות) (https://colab.research.google.com/): התחילו מחברת.
- JupyterLite: פייתון בדפדפן (Pyodide) (https://jupyter.org/try-jupyter/lab/): אפשרות בלי חשבון.

סיימתי את היחידה הזו

בקרת זרימה: תנאים ולולאות

לגרום לתוכניות להחליט עם if/elif/else ולחזור על פעולות עם לולאות for ו-while.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לכתוב if/elif/else עם אופרטורי השוואה ואופרטורים בוליאניים.
- לעבור על טווחים ורצפים עם for.
- להשתמש בלולאות while עם תנאי עצירה ברור.
- לצבור סכומים וספירות.

מושגי מפתח

- ההזחה מגדירה בלוקים בפייתון. ארבעה רווחים לכל רמה הם התקן.
- השוואות (<, >=, ==, !=) מפיקות ערכים בוליאניים; שלבו אותן עם and, or, not.
- for x in sequence חוזרת פעם אחת לכל איבר; range(a, b) נותן a, a+1, …, b−1.
- while חוזרת עד שהתנאי שלה הוא False. ודאו שמשהו בתוך הלולאה משנה אותו, אחרת היא לא תסתיים לעולם.
- תבנית הצובר: מתחילים סכום ב-0 ומוסיפים בתוך הלולאה.

משוואות מפתח

$$\sum_{n=1}^{100} n = \frac{100 \cdot 101}{2} = 5050$$

_n=1^100n=(100·101)/(2)=5050

הסכום של גאוס בודק את דוגמה 2.2

$$x_{k+1} = x_k + \Delta$$

x_k+1=x_k+Δ

לולאה שמקדמת ערך

דוגמאות פתורות שלב אחר שלב

דוגמה 2.1 סווגו את חריגת הטמפרטורה העולמית של NASA לשנת 2024.

```
anomaly = 1.28   # NASA GISTEMP, 2024, deg C vs 1951-1980
if anomaly >= 1.0:
    print("at least 1.0 C above the 1951-1980 average")
elif anomaly > 0:
```

```
    print("warmer than average")
else:
    print("at or below average")
```

1. 1.0 ≥ 1.28, לכן הענף הראשון רץ והשאר מדולגים.

פלט

```
at least 1.0 C above the 1951-1980 average
```

תשובה: הענפים נבדקים לפי הסדר; רק הראשון שמתקיים רץ.

דוגמה 2.2 חברו את המספרים 1 עד 100 בלולאה.

```
total = 0
for n in range(1, 101):
    total += n
print(total)
```

1. range(1, 101) עוצר לפני 101.
2. total += n מוסיף כל מספר.

פלט

```
5050
```

תשובה: 5050, בהתאמה לנוסחה של גאוס.

דוגמה 2.3 אם ה-CO_2 ימשיך לעלות ב-2.5 ppm בשנה מהממוצע של 2025, מתי הוא יעבור 450 ppm?

```
ppm = 427.35   # 2025 annual mean, Mauna Loa
year = 2025
while ppm < 450:
    ppm += 2.5
    year += 1
print(year, round(ppm, 2))
```

1. הלולאה מוסיפה 2.5 ppm ושנה אחת עד ש-ppm ≥ 450.

פלט

```
2035 452.35
```

תשובה: ב-2035, ב-452.35 ppm בתרחיש פשוט זה של קצב קבוע (לא תחזית).

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. כמה פעמים רצה for i in range(3, 10, 2)?

   4 פעמים (3, 5, 7, 9).

2. כתבו לולאה שסופרת את המספרים האי-זוגיים מתחת ל-20.

   count = sum(1 for n in range(20) if n % 2): 10.

3. מה שגוי ב: while x < 10: print(x)

   x אף פעם לא משתנה, אז הלולאה אינסופית.

תפיסות שגויות נפוצות

מיתוס: range(1, 10) כולל את 10.
בפועל: הוא עוצר ב-9.

מיתוס: else אחרי if הוא חובה.
בפועל: הוא אופציונלי.

מיתוס: הזחה היא רק עניין של סגנון.
בפועל: בפייתון היא מגדירה את מבנה התוכנית.

כרטיסיות

if / elif / else

בחירת ענף אחד.

for

חזרה לכל איבר.

while

חזרה כל עוד התנאי מתקיים.

range(a, b, s)

מ-a עד b−1, בצעד s.

צובר

break / continue

x += total בתוך לולאה.

יציאה מהלולאה / דילוג לסיבוב הבא.

בוחן לבדיקה עצמית

1. list(range(2, 6)) הוא…

[2, 3, 4, 5, 6]

[2, 3, 4, 5]

[3, 4, 5, 6]

[2, 6]

[2, 3, 4, 5]. עוצר לפני 6.

2. איזו מילת מפתח מוסיפה בדיקה נוספת אחרי if?

else if

elif

then

case

.elif. elif

3. לולאת while עוצרת כאשר…

התנאי שלה הוא False

אחרי 10 פעמים

אף פעם

ב-print

התנאי שלה הוא False. התנאי נבדק בכל סיבוב.

4. not (True and False) הוא…

True

False

None

שגיאה

True. True and False הוא False.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

ספים כמו 1.0 °C או 450 ppm הם בדיוק מה שתוכניות ניטור בודקות במשפטי if, ומסמנות שיאים והתרעות באופן אוטומטי כשמגיעים נתונים חדשים.

ב־Saving Planets: פיזיקה I

מקורות: NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp) · NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

מקורות לימוד חינמיים

- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיף 4: בקרת זרימה.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): הרצאות על הסתעפות ואיטרציה.

סיימתי את היחידה הזו

פונקציות

לארוז לוגיקה בפונקציות שניתן לעשות בהן שימוש חוזר ולבדוק אותן, עם פרמטרים, ברירות מחדל וערכי החזרה.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- להגדיר פונקציות עם def, פרמטרים ו-return.
- להשתמש בארגומנטים עם ברירת מחדל ובארגומנטים בשם.
- לכתוב docstrings.
- להבין טווח הכרה מקומי.

מושגי מפתח

- פונקציה נותנת שם לקטע לוגיקה. כותבים אותה פעם אחת, בודקים אותה פעם אחת, ומשתמשים בה בכל מקום.
- פרמטרים מקבלים ערכים; return מחזיר תוצאה. פונקציה בלי return מחזירה None.
- ארגומנטים עם ברירת מחדל (cl_max=1.6) מקצרים מקרים נפוצים; ארגומנטים בשם הופכים קריאות לקריאות.
- משתנים שנוצרים בתוך פונקציה הם מקומיים ונעלמים כשהיא חוזרת.
- docstring (המחרוזת בשורה הראשונה) מתעד מה הפונקציה עושה.

משוואות מפתח

$$F = \tfrac{9}{5}C + 32$$

F=95C+32

דוגמה 3.1

$$V_s = \sqrt{\frac{2W}{\rho S C_{L,\max}}}$$

V_s=2Wρ SC_L,

דוגמה 3.2, מתוך מבוא להנדסת אווירונאוטיקה וחלל

דוגמאות פתורות שלב אחר שלב

דוגמה 3.1 כתבו ובדקו פונקציה להמרה מצלזיוס לפרנהייט.

```
def c_to_f(c):
    """Convert Celsius to Fahrenheit."""
    return c * 9 / 5 + 32
```

```
for c in (-40, 0, 1.28, 100):
    print(c, "->", round(c_to_f(c), 2))
```

1. 40− זהה בשני הסולמות, בדיקה שימושית.
2. round(…, 2) מסדר את ערכי ה-float.

פלט

```
-40 -> -40.0
0 -> 32.0
1.28 -> 34.3
100 -> 212.0
```

תשובה: חריגה של 1.28 °C היא הפרש של 2.3 °F, אבל 1.28 °C כטמפרטורה הם 34.3 °F; יחידות חשובות.

דוגמה 3.2 הפכו את נוסחת מהירות ההזדקרות לפונקציה והשתמשו בה בשתי צפיפויות.

```
def stall_speed(weight_n, rho, area, cl_max=1.6):
    return (2 * weight_n / (rho * area * cl_max)) ** 0.5

print(round(stall_speed(10791, 1.227, 16.2), 1))
print(round(stall_speed(10791, 0.738, 16.2), 1))
```

1. ל-cl_max יש ברירת מחדל, ולכן הקוראים יכולים להשמיט אותו.
2. 0.5 ** הוא שורש ריבועי.

פלט

```
26.0
33.6
```

תשובה: 26.0 m/s בגובה פני הים ו-33.6 m/s בגובה 5000 m, בהתאמה לקורס האווירונאוטיקה.

דוגמה 3.3 כתבו פונקציה שמחזירה את השינויים משנה לשנה ברשימה (הממוצעים השנתיים במאונה לואה 2022–2025).

```
def growth(values):
    """Year-to-year changes of a list."""
    return [b - a for a, b in zip(values, values[1:])]

co2 = [418.53, 421.08, 424.61, 427.35]   # 2022-2025
print([round(d, 2) for d in growth(co2)])
```

1. zip מצמד כל ערך לערך הבא.
2. list comprehension בונה את התוצאה.

פלט

```
[2.55, 3.53, 2.74]
```

תשובה: עליות של 2.55, 3.53 ו-2.74 ppm.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. כתבו area_circle(r) ובדקו אותה ב-r = 2.

   return math.pi * r**2 → 12.566....

2. מה מחזירה פונקציה בלי return?

   None.

3. למה לבדוק את c_to_f ב-40−?

   שני הסולמות מסכימים שם, ולכן התשובה הצפויה ידועה.

תפיסות שגויות נפוצות

מיתוס: print ו-return הם אותו דבר.
בפועל: print מציג ערך; return מחזיר אותו לשימוש בהמשך.

מיתוס: שינוי פרמטר משנה את המשתנה של הקורא.
בפועל: קשירה מחדש של פרמטר היא מקומית (אובייקטים ברי שינוי הם חריג).

מיתוס: פונקציות צריכות להיות ארוכות ולעשות הכול.
בפועל: פונקציות קטנות בעלות מטרה אחת קלות יותר לבדיקה.

כרטיסיות

def	return
מגדיר פונקציה.	מחזיר ערך.

פרמטר	ארגומנט ברירת מחדל

שם לקלט.

ערך שמשמש אם הוא הושמט.

Docstring

מחרוזת תיעוד.

טווח הכרה

היכן שם נראה.

בוחן לבדיקה עצמית

1. מגדירים פונקציה עם…

func

def

function

רק lambda

:def. def name(params)

2. פונקציה בלי משפט return מחזירה…

0

None

False

שגיאה

None. None מרומז.

3. ב-def f(x, k=2), k הוא…

חובה

ארגומנט ברירת מחדל

גלובלי

קבוע

ארגומנט ברירת מחדל. יש לו ברירת מחדל.

4. משתנים שנוצרים בתוך פונקציה הם...

גלובלי

מקומיים

משותפים

קבועים

מקומיים. טווח הכרה מקומי.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

קודי אקלים והידרולוגיה בנויים מפונקציות קטנות ובדוקות כמו אלה, להמרות יחידות, לקצבי שינוי ולנוסחאות פיזיקליות, וכך מודלים גדולים נשארים אמינים.

ב־Saving Planets: מבוא להנדסת אווירונאוטיקה וחלל

מקורות: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/)

מקורות לימוד חינמיים

- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיף 4.8: הגדרת פונקציות.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): פונקציות ומשתנים.

סיימתי את היחידה הזו

רשימות, מילונים ומחרוזות

לאחסן אוספי נתונים: רשימות לרצפים, מילונים לחיפושים, ומחרוזות לטקסט.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לגשת לרשימות לפי אינדקס, לחתוך ולשנות אותן.
- להשתמש ב-len, sum, min, max ו-sorted.
- לבנות מילונים ולשאול אותם.
- לפצל מחרוזות ולספור מילים.

מושגי מפתח

- רשימה מחזיקה רצף מסודר: flows[0] הוא האיבר הראשון, flows[-1] האחרון, flows[2:5] פרוסה.
- רשימות הן ברות שינוי: append, insert, remove ו-sort משנים אותן במקום.
- מילון ממפה מפתחות לערכים: gistemp[2024] → 1.28. חיפושים מהירים לא משנה כמה הוא גדול.
- list comprehensions בונות רשימות בתמציתיות: [x * 2 for x in xs if x > 0].
- מחרוזות הן רצפים של תווים עם מתודות כמו split, strip, lower ו-startswith.

משוואות מפתח

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

x=1n_i=1^n x_i

ממוצע של רשימה (דוגמה 4.1)

דוגמאות פתורות שלב אחר שלב

דוגמה 4.1 סכמו עשרה ימים של ספיקה יומית ממוצעת של USGS לנהר הפוטומק ב-Little Falls (10–1 ביוני 2025, ברגל מעוקבת לשנייה).

```
flows = [44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600]
print(len(flows), max(flows), min(flows))
print(sum(flows) / len(flows))
print(sorted(flows)[:3])
```

1. len, max ו-min עובדים על כל רשימה.
2. sum/len הוא הממוצע; sorted מחזיר רשימה חדשה.

פלט

```
10 44700 12200
22980.0
[12200, 12600, 14100]
```

תשובה: ממוצע 22980 cfs; טווח 12200–44700 cfs.

דוגמה 4.2 מצאו את השנה החמה ביותר במילון של חריגות NASA GISTEMP.

```
gistemp = {2016: 1.01, 2020: 1.01, 2023: 1.17, 2024: 1.28, 2025: 1.18}
warmest = max(gistemp, key=gistemp.get)
print(warmest, gistemp[warmest])
print([y for y, a in gistemp.items() if a > 1.1])
```

1. max עם key=gistemp.get משווה ערכים.
2. comprehension מסנן איברים.

פלט

```
2024 1.28
[2023, 2024, 2025]
```

תשובה: 2024, ב-1.28 °C; השנים 2023–2025 כולן מעל 1.1 °C.

דוגמה 4.3 ספרו מילים בעזרת מילון.

```
words = "reduce reuse recycle reduce repair reduce".split()
counts = {}
for w in words:
    counts[w] = counts.get(w, 0) + 1
print(counts)
```

1. get(w, 0) מחזיר 0 למילה חדשה.

פלט

```
{'reduce': 3, 'reuse': 1, 'recycle': 1, 'repair': 1}
```

תשובה: «reduce» מופיעה 3 פעמים.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. מה זה [10, 20, 30, 40][1:3]?

   [20, 30].

2. איך מוסיפים 2026: 1.2 ל-gistemp?

   gistemp[2026] = 1.2.

3. העלו בריבוע כל מספר ב-xs בעזרת comprehension.

   [x**2 for x in xs].

תפיסות שגויות נפוצות

מיתוס: רשימות מתחילות באינדקס 1.
בפועל: הן מתחילות ב-0.

מיתוס: sorted(xs) משנה את xs.
בפועל: sorted מחזיר רשימה חדשה; xs.sort() משנה אותה.

מיתוס: מפתחות במילון יכולים לחזור.
בפועל: כל מפתח ייחודי; השמה חוזרת מחליפה את הערך.

כרטיסיות

רשימה

רצף מסודר ובר שינוי.

אינדקס / פרוסה

xs[i], xs[a:b].

מילון

מיפוי מפתח → ערך.

Comprehension

[f(x) for x in xs if cond].

Tuple

רצף שאינו בר שינוי.

str.split

טקסט → רשימת חלקים.

בוחן לבדיקה עצמית

1. xs[-1] הוא…

האיבר הראשון

האיבר האחרון

שגיאה

האורך

האיבר האחרון. אינדקסים שליליים סופרים מהסוף.

2. len({"a": 1, "b": 2}) =

1

2

4

3

2. שני מפתחות.

3. "a,b,c".split(",") נותן…

"abc"

["a", "b", "c"]

("a","b","c")

3

["a", "b", "c"]. רשימה של מחרוזות.

4. רשימות הן…

לא ברות שינוי

ברות שינוי

מילונים

תמיד ממוינות

ברות שינוי. הן יכולות להשתנות במקום.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

USGS מפרסם ספיקה יומית לאלפי מדי נהרות. עשרת המספרים בדוגמה 4.1 אמיתיים: גל ספיקה גבוהה בפוטומק שדועך במשך שבוע, סוג הרשומה שמשמשת לתכנון אספקת מים והצפות.

ב־Saving Planets: דף הנהרות

מקורות: USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

מקורות לימוד חינמיים

- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיף 5: מבני נתונים.
- USGS Water Services (ממשק נתונים) (https://waterservices.usgs.gov/): איך לבקש נתונים מ-USGS.

סיימתי את היחידה הזו

קבצים ונתוני CSV

לקרוא ולכתוב קבצים, לנתח נתוני CSV, ולטפל בשורות פגומות בלי לקרוס.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לפתוח, לקרוא ולכתוב קובצי טקסט עם with.
- לנתח CSV עם המודול csv.
- להמיר מחרוזות למספרים ולטפל בשגיאות עם try/except.
- להוריד מערך נתונים אמיתי ולנתח אותו.

מושגי מפתח

- with open(name) as f: פותח קובץ וסוגר אותו אוטומטית.
- CSV (ערכים מופרדים בפסיקים) הוא הפורמט הנפוץ ביותר לנתונים ציבוריים. csv.DictReader הופך כל שורה למילון שמפתחותיו לפי הכותרת.
- כל מה שנקרא מקובץ הוא טקסט: המירו עם int() ו-float().
- בנתונים אמיתיים יש הערות, שדות ריקים וערכים פגומים. try/except ValueError מאפשר לדלג עליהם או לתעד אותם.
- קובץ הממוצעים השנתיים של מאונה לואה של NOAA מתחיל בשורות הערה של #, ואחריהן year,mean,unc.

משוואות מפתח

$$\text{rate} = \frac{x_{last} - x_{first}}{t_{last} - t_{first}}$$

```
rate=_last-x_firstt_last-t_first
```

קצב שינוי ממוצע (דוגמה 5.1)

$$1 \text{ cfs} = 0.0283168 \text{ m}^3/\text{s}$$

```
1\ cfs=0.0283168\ m^3/s
```

המרת יחידות (דוגמה 5.2)

דוגמאות פתורות שלב אחר שלב

דוגמה 5.1 נתחו את חמש השורות האחרונות בקובץ ה-CSV של הממוצעים השנתיים של מאונה לואה של NOAA ומצאו את קצב העלייה הממוצע.

```
import csv, io
text = """year,mean,unc
2021,416.41,0.12
```

```
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
years = [int(r["year"]) for r in rows]
ppm = [float(r["mean"]) for r in rows]
print(years[0], years[-1], ppm[-1])
print(round((ppm[-1] - ppm[0]) / (years[-1] - years[0]), 2), "ppm per year")
```

1. io.StringIO מאפשר ל-csv לקרוא מחרוזת כאילו הייתה קובץ.
2. המירו את שדות הטקסט ל-int ול-float.

פלט

```
2021 2025 427.35
2.73 ppm per year
```

תשובה: 2.73 ppm בשנה מ-2021 עד 2025.

דוגמה 5.2 כתבו CSV קטן של ספיקות USGS, קראו אותו בחזרה והמירו ל-m³/s.

```
with open("flows.csv", "w") as f:
    f.write("date,cfs\n2025-06-01,44700\n2025-06-02,40200\n2025-06-03,29100\n")
with open("flows.csv") as f:
    next(f)                      # skip the header
    for line in f:
        date, cfs = line.strip().split(",")
        print(date, round(int(cfs) * 0.0283168, 1), "m3/s")
```

1. next(f) מדלג על שורת הכותרת.
2. strip מסיר את תו השורה החדשה; split מפריד שדות.

פלט

```
2025-06-01 1265.8 m3/s
2025-06-02 1138.3 m3/s
2025-06-03 824.0 m3/s
```

תשובה: 44700 cfs ≈ 1265.8 m³/s.

דוגמה 5.3 קראו שורות עם הערות וערכים פגומים בלי לקרוס.

```
lines = ["# comment", "year,mean", "2024,424.61", "2025,bad", "2026,"]
good = []
for line in lines:
    if line.startswith("#") or line.startswith("year"):
        continue
    year, value = line.split(",")
    try:
        good.append((int(year), float(value)))
    except ValueError:
        print("skipping", repr(line))
print(good)
```

1. continue מדלג על הערות ועל הכותרת.
2. float("bad") ו-float("") מעלים ValueError, שנתפסת.

פלט

```
skipping '2025,bad'
skipping '2026,'
[(2024, 424.61)]
```

תשובה: רק השורה התקינה נשמרת; הפגומות מדווחות.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. איך קוראים את קובץ NOAA המלא ב-Colab?

   השתמשו ב-urllib.request.urlopen(url) או הורידו אותו, ואז דלגו על שורות שמתחילות ב-#.

2. מה נותן csv.DictReader לכל שורה?

   dict שמפתחותיו שמות העמודות, עם ערכי מחרוזת.

3. למה להשתמש ב-with open(...)?

   הקובץ נסגר אוטומטית, גם אם מתרחשת שגיאה.

תפיסות שגויות נפוצות

מיתוס: מספרים ב-CSV הם מספרים.

בפועל: הם נקראים כמחרוזות עד שממירים אותם.

מיתוס: שורות פגומות הופכות את מערך הנתונים לחסר תועלת.

בפועל: דלגו עליהן ותעדו אותן, ואז דווחו כמה היו.

מיתוס: פתיחת קובץ במצב "w" מוסיפה לסופו.

בפועל: "w" דורס; "a" מוסיף לסוף.

כרטיסיות

with open()

פותח קובץ וסוגר אותו אוטומטית.

CSV

ערכים מופרדים בפסיקים.

csv.DictReader

שורות כמילונים (dict).

try / except

טיפול בשגיאות.

ValueError

המרה פגומה.

מצב "a" / "w";

דריסה / הוספה.

בוחן לבדיקה עצמית

1. נותן float("3.5")...

"3.5"

3.5

3

שגיאה

3.5. ממיר טקסט ל-float.

2. מעלה float("")...

TypeError

ValueError

KeyError

כלום

ValueError. מחרוזת ריקה אינה מספר.

3. מצב הקובץ "a"…

דורס

מוסיף לסוף

קורא

מוחק

מוסיף לסוף. הוספה.

4. המפתחות של csv.DictReader מגיעים מ…

מספרי השורות

שורת הכותרת

שם הקובץ

באקראי

שורת הכותרת. השורה הראשונה.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

NOAA, NASA ו-USGS משתפים את המדידות שלהם כקובצי CSV חינמיים. היכולת לקרוא אותם בעצמכם מאפשרת לבדוק כותרות מול נתוני המקור.

ב־Saving Planets: חשבון אינפיניטסימלי 2

מקורות: NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · תיעוד פייתון: המודול csv (https://docs.python.org/3/library/csv.html)

מקורות לימוד חינמיים

- תיעוד פייתון: המודול csv (https://docs.python.org/3/library/csv.html): קריאה וכתיבה של CSV.
- NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv): הקובץ האמיתי.
- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיף 7.2: קבצים.

- [ ] **סיימתי את היחידה הזו**

NumPy ו-Matplotlib

לחשב על מערכים שלמים בבת אחת עם NumPy, להתאים מגמות, וליצור גרפים ברורים עם Matplotlib.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- ליצור מערכי NumPy ולהשתמש באריתמטיקה וקטורית.
- להשתמש ב-mean, diff, מסכות בוליאניות ו-polyfit.
- לשרטט נתונים עם צירים מסומנים וכותרות.
- לשמור תרשימים לקבצים.

מושגי מפתח

- מערך NumPy מחזיק מספרים מטיפוס אחד ומחשב על כולם בבת אחת: 0.0283168 * cfs ממיר כל ערך.
- np.diff נותן הפרשים עוקבים; מסכה בוליאנית כמו cfs > 20000 בוחרת או סופרת ערכים.
- np.polyfit(x, y, 1) מתאים קו ישר ומחזיר שיפוע וחותך.
- pyplot של Matplotlib מצייר תרשימים: fig, ax = plt.subplots(); ax.plot(x, y); סמנו צירים עם יחידות; fig.savefig שומר את התמונה.
- לגרף טוב יש צירים מסומנים עם יחידות, כותרת ומקור נתונים מצוטט.

משוואות מפתח

$$y \approx m x + b$$

קו הריבועים הפחותים מ-np.polyfit

$$V = \sum Q_i \, \Delta t$$

```
V= Q_i Δ t
```

נפח מתוך ספיקות יומיות (דוגמה 6.2)

דוגמאות פתורות שלב אחר שלב

דוגמה 6.1 השתמשו ב-NumPy על הממוצעים השנתיים של מאונה לואה 2021–2025: הפרשים, ממוצע ומגמה.

```
import numpy as np
years = np.array([2021, 2022, 2023, 2024, 2025])
ppm = np.array([416.41, 418.53, 421.08, 424.61, 427.35])
print(np.diff(ppm))
print(round(ppm.mean(), 3), round(np.diff(ppm).mean(), 3))
```

```
slope, intercept = np.polyfit(years, ppm, 1)
print(round(slope, 3))
```

1. np.diff נותן ארבע עליות שנתיות.
2. השיפוע של polyfit הוא הקצב בהתאמה הטובה ביותר.

פלט

```
[2.12 2.55 3.53 2.74]
421.596 2.735
2.796
```

תשובה: עלייה בהתאמה הטובה ביותר 2.80 ppm ≈ בשנה ב-2021–2025.

דוגמה 6.2 המירו עשרה ימי ספיקה של הפוטומק ל-m³/s, ספרו ימי ספיקה גבוהה ומצאו את הנפח הכולל.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(np.round(m3s[:3], 1))
print(int((cfs > 20000).sum()), "days above 20,000 cfs")
print(round(m3s.sum() * 86400 / 1e9, 3), "km3 in 10 days")
```

1. הכפילו את כל המערך בבת אחת.
2. (cfs > 20000).sum() סופר ערכי True.
3. סכום של m³/s × 86400 s ליום.

פלט

```
[1265.8 1138.3  824. ]
5 days above 20,000 cfs
0.562 km3 in 10 days
```

תשובה: חמישה ימים מעל 20000 cfs; כ-0.56 km³ של מים עברו בעשרה ימים.

דוגמה 6.3 שרטטו חריגות שנתיות נבחרות של NASA GISTEMP ושמרו את התרשים.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
years = [1880, 1951, 1980, 2000, 2016, 2020, 2023, 2024, 2025]
anom = [-0.17, -0.07, 0.26, 0.39, 1.01, 1.01, 1.17, 1.28, 1.18]
```

```
fig, ax = plt.subplots()
ax.plot(years, anom, "o-")
ax.set_xlabel("Year"); ax.set_ylabel("Anomaly (C)")
ax.set_title("NASA GISTEMP v4, selected years")
fig.savefig("gistemp.png")
print(len(ax.lines[0].get_xdata()), "points saved")
```

1. matplotlib.use("Agg") מאפשר לקוד לשמור תמונות בלי מסך (Colab מציג גרפים בתוך הדף בכל מקרה).
2. סמנו צירים וכותרת; שמרו עם savefig.

פלט

```
9 points saved
```

תשובה: גרף קווי שמור של 9 נקודות; פתחו את gistemp.png כדי לראות אותו.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. np.array([1, 2, 3]) * 2 נותן?

   array([2, 4, 6]).

2. איך בוחרים ערכים מעל 1.0 במערך a?

   a[a > 1.0].

3. איזו קריאה מוסיפה תווית לציר?

   ax.set_xlabel("Year").

תפיסות שגויות נפוצות

מיתוס: מערכי NumPy ורשימות מתחברים באותה דרך.
בפועל: [3, 4] + [1, 2] משרשר; מערכים מתחברים איבר-איבר.

מיתוס: קו מותאם מוכיח סיבה.
בפועל: הוא מסכם מגמה; סיבות דורשות הבנה פיזיקלית.

מיתוס: גרפים בלי יחידות זה בסדר.
בפועל: תמיד סמנו יחידות וציינו את המקור.

כרטיסיות

np.array

מערך מספרי מהיר.

וקטורי

חישוב על מערכים שלמים.

מסכה בוליאנית

a[a > x].

np.polyfit

התאמת פולינום בריבועים פחותים.

plt.subplots

יוצר תרשים וצירים.

savefig

שומר גרף.

בוחן לבדיקה עצמית

1. np.diff([1, 4, 9]) =

[5 ,3]

[9 ,4 ,1]

[5 ,3 ,0]

14

[5 ,3]. הפרשים עוקבים.

2. np.polyfit(x, y, 1) מחזיר…

את הממוצע

שיפוע וחותך

גרף

את המקסימום

שיפוע וחותך. התאמה ממעלה 1.

3. (a > 0).sum() סופר…

ערכים חיוביים

את כל הערכים

אפסים

כלום

ערכים חיוביים. True נספר כ-1.

...4. fig.savefig("x.png")

מדפיס

שומר את התרשים

מוחק אותו

מציג קוד

שומר את התרשים. כותב קובץ תמונה.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

ניתוח GISTEMP של NASA מתפרסם עם הנתונים שלו כדי שכל אחד יוכל לשחזר את רשומת הטמפרטורה העולמית. לשרטט אותו בעצמכם, עם יחידות מסומנות והמקור, זו הדרך הכנה לשתף עובדות אקלים.

ב־Saving Planets: נתוני אקלים באתר הזה: CO2M

מקורות: NASA GISS: GISTEMP v4 surface temperature analysis (/https://data.giss.nasa.gov/gistemp) · NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html)

מקורות לימוד חינמיים

- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): מערכים למתחילים.
- Matplotlib: pyplot tutorial (https://matplotlib.org/stable/tutorials/pyplot.html): יסודות השרטוט.
- NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv): הנתונים האמיתיים.

סיימתי את היחידה הזו

ניפוי שגיאות ובדיקות

למצוא באגים בשיטתיות, לקרוא tracebacks ולכתוב בדיקות שתופסות טעויות לפני שהן משנות.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לקרוא traceback ולאתר את השורה שנכשלה.
- להשתמש ב-print ובהצהרות assert כדי לצמצם באגים.
- להעלות שגיאות ברורות לקלט פגום.
- לכתוב בדיקות יחידה עם assert ו-pytest.

מושגי מפתח

- קראו הודעות שגיאה מלמטה: השורה האחרונה מציינת את סוג השגיאה, והשורות מעליה מראות היכן היא קרתה.
- נפו שגיאות באופן מדעי: נסחו השערה, בדקו אותה עם קלט קטן, שנו דבר אחד.
- מקרי קצה (רשימות ריקות, אפס, מספרים שליליים, נתונים חסרים) גורמים לרוב הבאגים.
- assert מצהיר מה חייב להיות נכון; assert שנכשל עוצר את התוכנית במקום הבאג.
- בדיקות הן פונקציות קטנות שבודקות תשובות ידועות; pytest מריץ אוטומטית כל פונקציה ששמה test_....

משוואות מפתח

$$|x_{computed} - x_{expected}| < \varepsilon$$

בדיקה עם סבולת ל-float

דוגמאות פתורות שלב אחר שלב

דוגמה 7.1 מצאו את הבאג בפונקציית ממוצע שנקראת על רשימה ריקה.

```
def mean(xs):
    return sum(xs) / len(xs)

try:
    print(mean([]))
except ZeroDivisionError as e:
    print("bug found:", e)
```

1. חלוקה ב-0 = len([]) מעלה ZeroDivisionError.
2. תפיסתה מציגה את ההודעה.

פלט

```
bug found: division by zero
```

תשובה: הפונקציה צריכה לטפל בקלט ריק.

דוגמה 7.2 תקנו אותה עם שגיאה ברורה והוסיפו בדיקות assert.

```
def mean(xs):
    if not xs:
        raise ValueError("mean() of empty list")
    return sum(xs) / len(xs)

assert mean([1, 2, 3]) == 2
assert abs(mean([0.1, 0.2]) - 0.15) < 1e-12
try:
    mean([])
except ValueError:
    print("all tests passed")
```

1. העלו ValueError עם הודעה עבור קלט ריק.
2. תוצאות float מושוות עם סבולת.

פלט

```
all tests passed
```

תשובה: כל הבדיקות עוברות.

דוגמה 7.3 כתבו בדיקה בסגנון pytest ל-f_to_c בעזרת נקודות ידועות.

```
def f_to_c(f):
    return (f - 32) * 5 / 9

def test_f_to_c():
    assert f_to_c(32) == 0
    assert f_to_c(212) == 100
    assert abs(f_to_c(-40) - (-40)) < 1e-12

test_f_to_c()
print("ok")
```

1. זוגות ידועים: 32 °F = 0 °C, 212 °F = 100 °C, −40 = −40.

2. בקובץ, pytest היה מגלה את test_f_to_c אוטומטית; כאן אנחנו קוראים לה.

פלט

```
ok
```

תשובה: הפונקציה עוברת את כל שלוש הבדיקות.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. מה בדרך כלל פירוש «IndexError: list index out of range»?

   השתמשתם באינדקס len(list) ≥, לרוב טעות של סטייה באחד.

2. כתבו assert ש-x חיובי.

   assert x > 0, "x must be positive";.

3. ציינו שלושה מקרי קצה לפונקציה שמחשבת ממוצע של נתונים.

   רשימה ריקה, ערך יחיד, וערכים חסרים או לא מספריים.

תפיסות שגויות נפוצות

מיתוס: אם זה רץ, זה נכון.
בפועל: קוד יכול לרוץ ולתת תשובות שגויות; בדקו אותו מול תוצאות ידועות.

מיתוס: השוו ערכי float עם ==.
בפועל: השתמשו בסבולת.

מיתוס: בדיקות הן רק לפרויקטים גדולים.
בפועל: בדיקות קטנות חוסכות זמן מהיום הראשון.

כרטיסיות

Traceback

דוח שגיאה; קוראים מלמטה.

מקרה קצה

קלט חריג בגבול.

assert

עוצר אם תנאי שקרי.

בדיקת יחידה

בודקת פונקציה אחת מול תשובות ידועות.

pytest	**raise**
מריץ בדיקות.	מאותת על שגיאה במכוון.

בוחן לבדיקה עצמית

1. השורה השימושית ביותר ב-traceback היא בדרך כלל…

הראשונה

האחרונה

האמצעית

אף אחת

האחרונה. היא מציינת את השגיאה.

2. pytest מריץ פונקציות ששמן…

run_*

test_*

check_*

main

test_*. מוסכמה.

3. assert שנכשל מעלה…

ValueError

AssertionError

כלום

TypeError

.AssertionError. AssertionError

4. צעד ראשון טוב בניפוי שגיאות הוא...

לשכתב הכול

לשחזר עם קלט קטן

להוסיף print אקראיים בכל מקום

להתעלם

לשחזר עם קלט קטן. הקטינו את הבעיה.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

קוד מדעי שמשפיע על מדיניות חייב להיבדק. הקורס הזה עושה מה שהוא מטיף לו: כל דוגמת קוד בדף הזה מורצת על ידי סקריפט הבנייה שלנו, והפלט שלה חייב להתאים בדיוק לפני שהדף עולה.

ב־Saving Planets: קורסים

מקורות: תיעוד pytest (https://docs.pytest.org/) · המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/)

מקורות לימוד חינמיים

- תיעוד pytest (https://docs.pytest.org/): כתיבת בדיקות.
- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): סעיף 8: שגיאות וחריגות.

צפו בהרצאה

Lecture 7 testing debugging exceptions and assertions (https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/resources/lecture-7-testing-debugging-exceptions-and-assertions/) (באנגלית)

MIT 6.0001 Introduction to Computer Science and Programming in Python (Fall 2016), MIT OpenCourseWare. רישיון: CC BY-NC-SA 4.0 (https://creativecommons.org/licenses/by-nc-sa/4.0/) · תנאי שימוש (https://ocw.mit.edu/pages/privacy-and-terms-of-use/). הקישור מיועד ללימוד חינמי בלבד; Saving Planets אינו קשור לאוניברסיטה זו ואינו נתמך על ידה.

סיימתי את היחידה הזו

אלגוריתמים וסיבוכיות

חיפוש, מציאת שורשים ואיך זמן הריצה גדל עם גודל הקלט: חשיבת O גדול.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- לממש חיפוש לינארי וחיפוש בינארי.
- להשתמש בשיטת החציה לפתרון משוואות.
- להעריך סיבוכיות: O(1), O(log n), O(n), O(n²).
- לבחור מבני נתונים לפי מהירות.

מושגי מפתח

- אלגוריתם הוא שיטה מדויקת צעד אחר צעד. העלות שלו היא איך מספר הצעדים גדל עם גודל הקלט n.
- חיפוש לינארי בודק איברים אחד-אחד: O(n). חיפוש בינארי חוצה רשימה ממוינת בכל צעד: O(log n).
- שיטת החציה מוצאת שורש של f ב-[a, b] שבו f מחליפה סימן, על ידי חציית הקטע; אחרי n צעדים הקטע הוא (b − a)/2ⁿ.
- לולאות מקוננות על n איברים הן O(n²): בסדר ל-1000 איברים, חסר סיכוי למיליון.
- חיפושים במילון ובקבוצה הם O(1) בממוצע, וחיפושים ברשימה O(n).

משוואות מפתח

$$\text{steps}_{binary} \approx \lceil \log_2 n \rceil$$

צעדי חיפוש בינארי

$$n = \left\lceil \log_2 \frac{b - a}{\text{tol}} \right\rceil$$

`n=_2-atol`

איטרציות של שיטת החציה

דוגמאות פתורות שלב אחר שלב

דוגמה 8.1 חפשו את האיבר האחרון ברשימה ממוינת של מיליון מספרים זוגיים, בשתי הדרכים.

```
def linear_search(xs, target):
    for i, x in enumerate(xs):
        if x == target:
            return i
    return -1
```

```
def binary_search(xs, target):
    lo, hi, steps = 0, len(xs) - 1, 0
    while lo <= hi:
        steps += 1
        mid = (lo + hi) // 2
        if xs[mid] == target:
            return mid, steps
        if xs[mid] < target:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1, steps

data = list(range(0, 2_000_000, 2))
print(linear_search(data, 1_999_998))
print(binary_search(data, 1_999_998))
```

1. חיפוש לינארי בודק 1000000 איברים כדי להגיע לאינדקס 999999.
2. חיפוש בינארי צריך רק 20 צעדים.

פלט

```
999999
(999999, 20)
```

תשובה: 20 צעדים לעומת מיליון: $\log_2(10^6) \approx 20$.

דוגמה 8.2 מצאו את $\sqrt{2}$ בשיטת החציה על $x^2 - 2$ מ-$[0, 2]$ עד 10^{-6}.

```
def bisect_root(f, a, b, tol=1e-6):
    n = 0
    while b - a > tol:
        m = (a + b) / 2
        if f(a) * f(m) <= 0:
            b = m
        else:
            a = m
        n += 1
    return (a + b) / 2, n

root, n = bisect_root(lambda x: x**2 - 2, 0, 2)
print(round(root, 6), n)
```

1. כל צעד חוצה את הקטע.
2. $2/2^{21} < 10^{-6}$ אחרי 21 צעדים.

פלט

```
1.414214 21
```

תשובה: 1.414214 אחרי 21 איטרציות.

דוגמה 8.3 השוו את הגדילה של $\log_2 n$ ושל n^2.

```
import math
for n in (10, 1_000, 1_000_000):
    print(n, math.ceil(math.log2(n)), n * n)
```

1. math.ceil(math.log2(n)) הוא מספר הצעדים של חיפוש בינארי.

פלט

```
10 4 100
1000 10 1000000
1000000 20 1000000000000
```

תשובה: עבור $n = 10^6$, $\log_2 n$ הוא 20 בעוד n^2 הוא 10^{12}.

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. כמה צעדי חיפוש בינארי ל-1 מיליארד איברים ממוינים?

   $\lceil \log_2 10^9 \rceil = 30$.

2. מה הסיבוכיות של סכימת רשימה?

   $O(n)$.

3. שיטת החציה על $[0, 1]$ עד 10^{-3}: כמה צעדים?

   $\lceil \log_2 1000 \rceil = 10$.

תפיסות שגויות נפוצות

מיתוס: מחשבים מהירים יותר הופכים את הסיבוכיות ללא רלוונטית.

בפועל: שיטה של $O(n^2)$ על מיליון איברים היא 10^{12} צעדים; אף מחשב לא יתקן זאת.

מיתוס: חיפוש בינארי עובד על כל רשימה.

בפועל: רק על נתונים ממוינים.

מיתוס: O גדול נותן זמן מדויק.

בפועל: הוא מתאר גדילה ומתעלם מקבועים.

כרטיסיות

אלגוריתם

שיטה מדויקת צעד אחר צעד.

Big-O

קצב הגדילה של העלות.

חיפוש לינארי

O(n).

חיפוש בינארי

O(log n), נתונים ממוינים.

שיטת החציה

מציאת שורש על ידי חציה.

O(n²)

לולאות מקוננות.

בוחן לבדיקה עצמית

1. חיפוש בינארי הוא…

O(n)

O(log n)

O(n²)

O(1)

O(log n). חוצה בכל צעד.

2. שיטת החציה דורשת…

נגזרת

החלפת סימן ב-[a, b]

נתונים ממוינים

מספרים שלמים

החלפת סימן ב-[a, b]. משפט ערך הביניים.

3. לולאה בתוך לולאה על n איברים היא...

O(n)

O(n²)

O(log n)

O(1)

O(n²). n × n.

4. חיפוש ממוצע ב-dict הוא...

O(n)

O(1)

O(n²)

O(log n)

O(1). גיבוב (Hashing).

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

אלגוריתמים יעילים מאפשרים לעבד את מבול נתוני תצפית כדור הארץ, מלוויינים ועד אלפי מדי נהרות. בחירה ב-O(n log n) במקום O(n²) יכולה להפוך שבוע של חישוב לשניות, ופחות חישוב פירושו גם פחות אנרגיה.

ב־Saving Planets: חשבון אינפיניטסימלי 1

מקורות: USGS Water Services (ממשק נתונים) (/https://waterservices.usgs.gov) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (/https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022)

מקורות לימוד חינמיים

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): הרצאות על חיפוש, שיטת החציה וסיבוכיות.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): תרגול פתרון בעיות.

- [ ] **סיימתי את היחידה הזו**

סימולציה נומרית

לקדם מודלים פיזיקליים בזמן: קליע, מסלול ואוכלוסייה, הליבה של כל מודל אקלים והנדסה.

מטרות הלמידה

בסוף היחידה הזו תוכלו:

- להשתמש בצעדי זמן (אוילר ואוילר-קרומר) כדי לדמות תנועה.
- לבדוק סימולציות מול תשובות מדויקות.
- לדמות מסלול מעגלי ואוכלוסייה לוגיסטית.
- לחקור איך גודל הצעד משפיע על הדיוק.

מושגי מפתח

- סימולציה מעדכנת את המצב (מיקום, מהירות, אוכלוסייה) בצעדי זמן קטנים dt בעזרת קצבי השינוי: x_new = x + v·dt.
- אוילר-קרומר מעדכנת קודם את המהירות, ואז את המיקום עם המהירות החדשה; היא שומרת על יציבות מסלולים הרבה יותר טוב מאוילר פשוט.
- תמיד אמתו: השוו לפתרון מדויק במקרה שבו קיים כזה (בלי גרר אוויר, מסלול מעגלי, הנוסחה הלוגיסטית).
- dt קטן יותר מדויק יותר אבל איטי יותר; חצו את dt ובדקו אם התשובה משתנה.
- מודלים של מזג אוויר ואקלים הם גרסאות ענקיות של הלולאות האלה, שמקדמות את משוואות הפיזיקה על פני רשת עולמית.

משוואות מפתח

$$v_{k+1} = v_k + a_k\,\Delta t, \quad x_{k+1} = x_k + v_{k+1}\,\Delta t$$

```
v_k+1=v_k+a_k Δ t, x_k+1=x_k+v_k+1 Δ t
```

צעד אוילר-קרומר

$$R = \frac{v_0^2 \sin 2\theta}{g}$$

```
R=(v_0^22θ)/(g)
```

טווח קליע בלי גרר

$$\vec{a} = -\frac{GM}{r^3}\vec{r}$$

```
a=-(GM)/(r^3) r
```

כבידה למסלול

$$P_{k+1} = P_k + h\,r P_k\left(1 - \frac{P_k}{K}\right)$$

דוגמאות פתורות שלב אחר שלב

דוגמה 9.1 דמו קליע במהירות 20 m/s בזווית 45° והשוו לטווח המדויק.

```
import math
v0, angle, g, dt = 20.0, math.radians(45), 9.81, 0.001
x, y = 0.0, 0.0
vx, vy = v0 * math.cos(angle), v0 * math.sin(angle)
while y >= 0:
    vy -= g * dt
    x += vx * dt
    y += vy * dt
print(round(x, 2), "m simulated;", round(v0**2 / g, 2), "m exact")
```

1. קדמו את המהירות, ואז את המיקום, עד שהוא נוחת.
2. טווח מדויק $= v_0^2 \sin 90°/g$.

פלט

```
40.77 m simulated; 40.77 m exact
```

תשובה: שניהם נותנים 40.77 m בשתי ספרות אחרי הנקודה עם dt = 0.001 s.

דוגמה 9.2 דמו מסלול בגובה 400 km מעל כדור הארץ בצעדים של 1 שנייה.

```
import math
GM = 3.986e14
r0 = 6.771e6
x, y = r0, 0.0
vx, vy = 0.0, math.sqrt(GM / r0)
dt, t = 1.0, 0
period = 2 * math.pi * math.sqrt(r0**3 / GM)
while t < period:
    r3 = (x * x + y * y) ** 1.5
    vx -= GM * x / r3 * dt          # Euler-Cromer: update velocity first
    vy -= GM * y / r3 * dt
    x += vx * dt
    y += vy * dt
```

```
    t += dt
print(round(period / 60, 1), "min;", round(math.hypot(x, y) / 1000), "km from Earth's center
```

1. התחילו ב-r_0 במהירות המעגלית $\sqrt{(GM/r_0)}$.
2. עדכנו את המהירות מהכבידה, ואז את המיקום.

פלט

```
92.4 min; 6771 km from Earth's center
```

תשובה: אחרי זמן מחזור אחד (92.4 דקות) החללית עדיין במרחק 6771 km ממרכז כדור הארץ: המסלול נשאר מעגלי.

דוגמה 9.3 קדמו אוכלוסיית דגים לוגיסטית ($r = 0.5$, $K = 1000$, $P_0 = 100$) במשך ארבע שנים.

```
r, K, P, h = 0.5, 1000, 100.0, 1.0
for year in range(1, 5):
    P += h * r * P * (1 - P / K)
    print(year, round(P, 1))
```

1. כל שנה מוסיפה $rP(1 - P/K)$.

פלט

```
1 145.0
2 207.0
3 289.1
4 391.8
```

תשובה: 145.0, 207.0, 289.1, 391.8, בהתאמה למעבדה A בחשבון אינפיניטסימלי II (שם 391.9, הפרש עיגול).

תרגילים

פתרו כל אחד קודם על נייר, ואז בדקו.

1. מה קורה לשגיאה של דוגמה 9.1 אם dt הוא 0.1 s?

   הנחיתה מתגלה באיחור, ולכן שגיאת הטווח גדלה; נסו והשוו.

2. למה אוילר פשוט גורם למסלול להסתלסל החוצה?

   הוא מוסיף מעט אנרגיה בכל צעד; אוילר-קרומר כמעט משמרת אותה.

3. הוסיפו גרר $-k\,v\,v$ לדוגמה 9.1. מה קורה לזווית הטובה ביותר?

היא יורדת מתחת ל-45° (מעבדה D).

תפיסות שגויות נפוצות

מיתוס: סימולציה שרצה היא נכונה.

בפועל: אמתו מול מקרים מדויקים ובדקו שחציית dt כמעט לא משנה את התשובה.

מיתוס: dt קטן יותר תמיד טוב יותר.

בפועל: זה עולה זמן ויכול להוסיף שגיאת עיגול; בחרו dt לפי בדיקות התכנסות.

מיתוס: מודלים חוזים את העתיד במדויק.

בפועל: הם קירובים שיש לציין את הנחותיהם.

כרטיסיות

צעד זמן dt

גודל כל עדכון.

שיטת אוילר

x += v·dt.

אוילר-קרומר

קודם מהירות, אחר כך מיקום.

אימות

בדיקה מול תשובות ידועות.

התכנסות

התשובה מפסיקה להשתנות כש-dt קטן.

מצב

משתנים שמתארים את המערכת כעת.

בוחן לבדיקה עצמית

1. אוילר-קרומר מעדכנת…

קודם את המיקום

קודם את המהירות, ואז את המיקום עם המהירות החדשה

רק את המהירות

כלום

קודם את המהירות, ואז את המיקום עם המהירות החדשה. יציבה למסלולים.

2. כדי לאמת סימולציה...

לסמוך עליה

להשוות לפתרון מדויק

להגדיל מאוד את dt

להסיר את הפיזיקה

להשוות לפתרון מדויק. מקרים ידועים.

3. טווח מדויק ב-20 m/s, g = 9.81 ,45°:

20.4 m

40.8 m

81.5 m

9.8 m

.40.8 m. 400/9.81

4. חציית dt בלי לראות שינוי פירושה...

באג

התוצאה התכנסה

המודל שגוי

כלום

התוצאה התכנסה. בדיקת התכנסות.

לנסות שוב את הבוחן

איך זה עוזר להציל את כדור הארץ

מודלים של אוכלוסייה כמו דוגמה 9.3 הם חלק מהדרך שבה מדעני דיג מעריכים מלאים וקובעים מכסות דיג בנות קיימא.

אותו רעיון של צעדי זמן, בהגדלה עצומה, הוא הדרך שבה בונים מודלים של מזג אוויר ואקלים.

ב־Saving Planets: חשבון אינפיניטסימלי II, מעבדה A · מדריך מסלולים

מקורות: NOAA Fisheries: הערכות אוכלוסייה (https://www.fisheries.noaa.gov/topic/population-assessments) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/)

מקורות לימוד חינמיים

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): הרצאות על סימולציה ומידול.
- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): וקטוריזציה של סימולציות.

סיימתי את היחידה הזו

תרגילים

התרגילים האלה נמצאים בתוך כל יחידה. פתרו על הנייר, ואז פתחו את התשובה.

1. צעדים ראשונים: משתנים, טיפוסים וביטויים
   4 תרגילים
2. בקרת זרימה: תנאים ולולאות
   3 תרגילים
3. פונקציות
   3 תרגילים
4. רשימות, מילונים ומחרוזות
   3 תרגילים
5. קבצים ונתוני CSV
   3 תרגילים
6. NumPy ו-Matplotlib
   3 תרגילים
7. ניפוי שגיאות ובדיקות
   3 תרגילים
8. אלגוריתמים וסיבוכיות
   3 תרגילים
9. סימולציה נומרית
   3 תרגילים

מעבדות תכנות

ארבע מעבדות תכנות משתמשות בנתונים ציבוריים אמיתיים ובסימולציה: רשומת ה-CO_2 של מאונה לואה, רשומת הטמפרטורה העולמית של נאס״א, תחנת מדידה נהרית של USGS וקליע עם גרר אוויר.

קוד הפתיחה של כל מעבדה רץ כפי שמוצג; התוצאות המודפסות הופקו מהרצתו. אחר כך הורידו את מערך הנתונים המלא והרחיבו את הניתוח.

מקור בטיחות: המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/)

מעבדה A: רשומת ה-CO_2 של מאונה לואה (NOAA)

שבוע 6

מטרה: טענו את קובץ ה-CSV של ממוצעי ה-CO_2 השנתיים של NOAA, חשבו את העלייה הכוללת ואת קצב הגידול האחרון, ושרטטו את העקומה.

$$\text{rate} = \text{slope of } \text{np.polyfit}(\text{year}, \text{ppm}, 1)$$

```
rate=slope of np.polyfit(year, ppm, 1)
```

קצב גידול מהתאמה לינארית

מהלך הניסוי

1. הריצו את קוד הפתיחה (כמה שורות אמיתיות מהקובץ של NOAA).
2. הורידו את קובץ ה-CSV המלא מ-NOAA (קישור למטה), דלגו על שורות #, וטענו את כל השנים.
3. שרטטו ppm מול שנה עם צירים מתויגים ועם המקור.
4. התאימו קווים ל-1960–1969 ול-2016–2025. איך השתנה הקצב?

```
import csv, io
import numpy as np
text = """year,mean,unc
1959,315.98,0.12
1960,316.91,0.12
2020,414.21,0.12
2021,416.41,0.12
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
yr = np.array([int(r["year"]) for r in rows])
ppm = np.array([float(r["mean"]) for r in rows])
print(round(ppm[-1] - ppm[0], 2), "ppm since 1959")
recent = yr >= 2020
slope = np.polyfit(yr[recent], ppm[recent], 1)[0]
print(round(slope, 2), "ppm per year, 2020-2025")
```

פלט

```
111.37 ppm since 1959
2.65 ppm per year, 2020-2025
```

טבלת נתונים

שנה	ממוצע CO_2 שנתי (ppm)
1959	315.98
1960	316.91
2020	414.21
2023	421.08
2024	424.61
2025	427.35

ערכים אמיתיים מקובץ הממוצעים השנתיים של מאונה לואה ב-NOAA GML, נשלפו ב-24 באוק' 2026.

חישוב לדוגמה

1. 427.35 − 315.98 = 111.37 ppm מאז 1959.
2. קצב ההתאמה הטובה ביותר 2.65 ppm ≈ 2020–2025 לשנה.

בטיחות

- אין, מעבר לשימוש רגיל במחשב.

חלופה ביתית או וירטואלית

- Colab או JupyterLite בדפדפן.

NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · Google Colab (מחברות חינמיות) (/https://colab.research.google.com)

הקשר לכדור הארץ: זו הרשומה שמאחורי עקומת קילינג, המדידה הרציפה הארוכה ביותר של עליית ה-CO_2. NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

☐ **השלמתי את המעבדה הזו (רטובה או וירטואלית)**

מעבדה B: טמפרטורה עולמית מ-NASA GISTEMP

שבוע 8

מטרה: נתחו את חריגות הטמפרטורה השנתיות העולמיות של נאס"א: ממוצעים, השנה החמה ביותר והמגמה.

$$\Delta T \text{ per decade} = 10 \times \text{slope}$$

```
Δ T\ per decade=10×slope
```

מגמה לעשור

מהלך הניסוי

1. הריצו את קוד הפתיחה עם החריגות השנתיות (J–D) של 2016–2025.
2. הורידו את קובץ ה-CSV העולמי המלא של GISTEMP; דלגו על שורת הכותרת הראשונה; טפלו בערכים חסרים ***.
3. שרטטו את 1880–2025 וחשבו את המגמה מאז 1975.
4. השוו התאמה של 10 שנים להתאמה של 50 שנה והסבירו למה מגמות קצרות רועשות יותר.

```
import numpy as np
years = np.arange(2016, 2026)
anom = np.array([1.01, 0.91, 0.85, 0.98, 1.01, 0.85, 0.89, 1.17, 1.28, 1.18])
print(round(anom.mean(), 3))
print(int(years[anom.argmax()]), anom.max())
print(round(np.polyfit(years, anom, 1)[0] * 10, 2), "C per decade (10-year fit)")
```

פלט

```
1.013
2024 1.28
0.32 C per decade (10-year fit)
```

טבלת נתונים

שנה	חריגה (°C ביחס ל-1980–1951)
2016	1.01
2020	1.01
2023	1.17
2024	1.28
2025	1.18

ערכים אמיתיים מ-NASA GISTEMP v4 (GLB.Ts+dSST, עמודת J–D), נשלפו ב-24 באוק' 2026. נאס"א מעדכנת את הניתוח, ולכן הורדות מאוחרות יותר עשויות להיות שונות מעט.

חישוב לדוגמה

1. ממוצע 1.013 °C = 2016–2025.
2. התאמה של עשר שנים נותנת כ-0.32 °C לעשור, אבל עשר שנים הן תקופה קצרה ורועשת; השתמשו בתקופות ארוכות יותר למגמה ארוכת הטווח.

בטיחות

- אין, מעבר לשימוש רגיל במחשב.

חלופה ביתית או וירטואלית

- Colab או JupyterLite.

NASA GISS: NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv) · GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

הקשר לכדור הארץ: לשחזר בעצמכם את הרשומה של נאס"א זו התשובה הטובה ביותר לשאלה "איך אנחנו יודעים?" NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

☐ **השלמתי את המעבדה הזו (רטובה או וירטואלית)**

מעבדה C: נהר דרך תחנת מדידה של USGS

שבוע 9

מטרה: נתחו ספיקה יומית מתחנת USGS על נהר הפוטומק בליטל פולס: יחידות, סטטיסטיקה, נסיגה ונפח.

$$V = \sum_{i} Q_i \times 86{,}400 \text{ s}$$

```
V=_iQ_i×86,400\ s
```

נפח מספיקות יומיות ממוצעות

מהלך הניסוי

1. הריצו את קוד הפתיחה עם עשרה ימים אמיתיים (10–1 ביוני 2025).
2. בקשו שנה מלאה מ-USGS Water Services (format=rdb) עבור אתר 01646500 ופרמטר 00060, ונתחו את הקובץ המופרד בטאבים.
3. שרטטו ספיקה יומית, מצאו את השיא וחשבו את הנפח השנתי ב-km³.
4. כתבו בדיקות לפונקציית המרת היחידות שלכם.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(round(m3s.mean(), 1), round(np.median(m3s), 1))
drops = np.diff(cfs)
print(int(drops.min()), "cfs biggest one-day drop")
print(round(m3s.sum() * 86400 / 1e6, 1), "million m3 in 10 days")
```

פלט

```
650.7 564.9
-11100 cfs biggest one-day drop
562.2 million m3 in 10 days
```

טבלת נתונים

תאריך (2025)	ספיקה יומית ממוצעת (cfs)
1 ביוני	44,700
3 ביוני	29,100
5 ביוני	17,300
8 ביוני	12,200
10 ביוני	22,600

ערכים יומיים אמיתיים ומאושרים מאתר USGS 01646500, נשלפו ב-24 באוק' 2026.

חישוב לדוגמה

1. ממוצע 650.7 m³/s, חציון 564.9 m³/s (השיא של 1 ביוני מושך את הממוצע למעלה).
2. הירידה הגדולה ביותר ביום אחד הייתה 11100 cfs; כ-562 מיליון m³ עברו בעשרה ימים.

בטיחות

- אין, מעבר לשימוש רגיל במחשב.

חלופה ביתית או וירטואלית

- Colab או JupyterLite.

USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · USGS Water Services (ממשק נתונים) (https://waterservices.usgs.gov/)

הקשר לכדור הארץ: תחנות מדידה נהריות כמו זו תומכות באספקת מי שתייה, בהתרעות שיטפון ובזרימות אקולוגיות. USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/)

☐ **השלמתי את המעבדה הזו (רטובה או וירטואלית)**

מעבדה D: קליע עם גרר אוויר

שבוע 14

מטרה: הרחיבו את סימולציית הקליע עם גרר ריבועי ומצאו את זווית השיגור הטובה ביותר ככל שהגרר גדל.

$$\vec{a} = -g\hat{y} - k\,|\vec{v}|\,\vec{v}$$

a=-g y-k | v| v

גרר ריבועי (k למטר)

מהלך הניסוי

1. הריצו את קוד הפתיחה, שבודק זוויות שלמות מ-20° עד 60°.
2. בדקו ש-k = 0 נותן 45° ואת הטווח המדויק.
3. חצו את dt וודאו שהתוצאות כמעט לא משתנות.
4. שרטטו טווח מול זווית לכל k.

```
import math
def fly(v0, deg, k=0.0, dt=0.001, g=9.81):
    vx, vy = v0 * math.cos(math.radians(deg)), v0 * math.sin(math.radians(deg))
    x = y = 0.0
    while y >= 0:
        v = math.hypot(vx, vy)
        vx -= k * v * vx * dt
        vy -= (g + k * v * vy) * dt
        x += vx * dt
        y += vy * dt
    return x
for k in (0.0, 0.01, 0.05):
    best = max(range(20, 61), key=lambda a: fly(20, a, k))
    print(k, best, round(fly(20, best, k), 1))
```

פלט

```
0.0 45 40.8
0.01 43 31.4
0.05 39 17.8
```

טבלת נתונים

גרר k (1/m)	הזווית הטובה ביותר	טווח (m)
0	45°	40.8
0.01	43°	31.4

גרר k (1/m)	הזווית הטובה ביותר	טווח (m)
0.05	39°	17.8

חושב באמצעות קוד הפתיחה (v_0 = 20 m/s, dt = 0.001 s).

חישוב לדוגמה

1. בלי גרר הזווית הטובה ביותר היא 45° והטווח 40.8 m (מדויק 40.77 m).
2. גרר מקצר את הטווח ומנמיך את הזווית הטובה ביותר.

בטיחות

- אין, מעבר לשימוש רגיל במחשב.

חלופה ביתית או וירטואלית

- Colab או JupyterLite.

Google Colab (מחברות חינמיות) (https://colab.research.google.com/) · JupyterLite: פייתון בדפדפן (Pyodide) (https://jupyter.org/try-jupyter/lab/)

הקשר לכדור הארץ: גרר הוא אותה פיזיקה שקובעת את צריכת הדלק של מטוס (מבוא להנדסת אווירונאוטיקה וחלל).

☐ **השלמתי את המעבדה הזו (רטובה או וירטואלית)**

מערכי תרגול למבחן

מדדו זמן: כ־10 דקות לבעיה. אחר כך השוו לפתרונות המלאים.

סט תרגול לבחינת אמצע (יחידות 5–1)

1. מה מדפיס print(7 // 2, 7 % 2)?

   1. חילוק שלמים ושארית.

   תשובה: 3 1

2. כתבו לולאה שסוכמת את המספרים מ-1 עד 100.

   1. total = 0; for n in range(1, 101): total += n.

   תשובה: 5050

3. כתבו stall_speed(W, rho, S, cl_max=1.6) וחשבו אותה ב-10791 N, 1.227 kg/m³, 16.2 m².

.(2W/(ρSC))**0.5 .1

תשובה: 26.0 m/s

4. בהינתן co2 = [418.53, 421.08, 424.61, 427.35], חשבו את השינויים משנה לשנה.

.[b - a for a, b in zip(co2, co2[1:])] .1

תשובה: [2.55, 3.53, 2.74]

5. למה float("") צריך try/except בקריאת נתונים?

1. הוא מעלה ValueError.

תשובה: כדי לדלג על שורות פגומות או לרשום אותן ביומן במקום לקרוס.

מערך תרגול מסכם (כל היחידות)

1. המירו 44700 cfs ל-m³/s.

.× 0.0283168 .1

תשובה: ≈ 1,265.8 m³/s

2. כמה צעדי חיפוש בינארי נדרשים למיליון פריטים ממוינים?

.⌈log₂ 10⁶⌉ .1

תשובה: 20

3. כמה צעדי חצייה מביאים את [0, 2] אל מתחת ל-10^{-6}?

.$2/2^n < 10^{-6}$ .1

תשובה: 21

4. מהו הטווח המדויק של קליע ב-20 m/s בזווית (g = 9.81) °45?

.v_0^2/g .1

תשובה: 40.77 m

5. ארבעה צעדי אוילר של P += 0.5P(1 − P/1000) החל מ-100?

1. 145.0, 207.0, 289.1, …

תשובה: 391.8

6. כתבו בדיקה ל-f_to_c.

1. assert f_to_c(212) == 100, ועוד.

תשובה: זוגות ידועים: −40 → −40, 100 → 0, 212 → 32.

דף נוסחאות

$$7//2 = 3,\ 7\,\%\,2 = 1,\ 2^{10} = 1024$$

$$\bar{x} = \tfrac{1}{n}\sum x_i$$

$$\text{rate} = \Delta y / \Delta x$$

$$y \approx mx + b\ (\text{np.polyfit})$$

$$1\ \text{cfs} = 0.0283168\ \text{m}^3/\text{s}$$

$$\text{binary search} \approx \lceil \log_2 n \rceil$$

$$n_{bisect} = \lceil \log_2((b-a)/\text{tol}) \rceil$$

$$v\ +\!= a\,\Delta t,\ x\ +\!= v\,\Delta t$$

$$R = v_0^2 \sin 2\theta / g$$

$$P\ +\!= hrP(1 - P/K)$$

קבועים

קבוע	ערך
1 רגל מעוקבת לשנייה (cfs)	0.0283168 m³/s
שניות ביממה	86,400
CO_2 במאונה לואה, ממוצע שנתי 1959	315.98 ppm
CO_2 במאונה לואה, ממוצע שנתי 2025	427.35 ppm
חריגת GISTEMP לשנת 2024 (ביחס ל-1951–1980)	1.28 °C
GM של כדור הארץ	3.986 × 10¹⁴ m³/s²
g	9.81 m/s²

מקור: NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

כלים ומקורות חינמיים

הריצו פייתון בחינם בדפדפן וחפשו כאן מידע.

- Google Colab (מחברות חינמיות) (https://colab.research.google.com/): מחברות חינמיות עם NumPy ו-Matplotlib מותקנים מראש.
- JupyterLite: פייתון בדפדפן (Pyodide) (https://jupyter.org/try-jupyter/lab/): פייתון בדפדפן, בלי חשבון, מבוסס Pyodide.
- המדריך הרשמי של פייתון (https://docs.python.org/3/tutorial/): מדריך הייחוס.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): קורס המבוא המלא של MIT.

- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): קורס הפייתון החינמי של הרווארד.

מילון מונחים

אלגוריתם

שיטה מדויקת צעד אחר צעד.

מערך

מכל של NumPy לחישוב מספרי מהיר.

טענת assert

בדיקה שעוצרת את התוכנית אם היא שקרית.

Big-O

איך העלות גדלה עם גודל הקלט.

שיטת החציה

מציאת שורשים בחציית קטע.

CSV

קובץ ערכים מופרדים בפסיקים.

מילון

מיפוי מפתח → ערך.

מקרה קצה

קלט חריג בגבול.

אוילר-קרומר

צעדי זמן יציבים: קודם המהירות.

חריגה

שגיאה שמועלית בזמן הריצה.

פונקציה

קטע קוד בעל שם שאפשר להשתמש בו שוב.

רשימה

רצף מסודר ובר שינוי.

לולאה

ביצוע חוזר.

מחברת

מסמך של תאי קוד שניתן להריץ.

Pyodide

פייתון מהודר לריצה בדפדפן אינטרנט.

סימולציה

קידום מודל קדימה בזמן.

Traceback

דוח על המקום שבו קרתה שגיאה.

בדיקת יחידה
בדיקה קטנה של פונקציה אחת.

משתנה
שם הקשור לערך.

f-string
מחרוזת מעוצבת מילולית.

מחוון להערכה עצמית

לא ניתנים ציון או נקודות זכות. השתמשו בזה כדי להעריך בכנות איפה אתם עומדים בכל יחידה.

רמה	איך זה נראה
4 · שליטה מלאה	כותב פונקציות ברורות ובדוקות; מאמת תוצאות מול תשובות ידועות; מסביר את בחירותיו.
3 · מיומן	התוכניות עובדות, עם באגים מזדמנים שמתגלים בבדיקות.
2 · מתפתח	יודע לשנות דוגמאות אבל מתקשה לכתוב מאפס.
1 · מתחיל	מעתיק קוד בלי להבין; חזרו על היחידה.

שקלול מוצע אם רוצים ציון

חלק	משקל
תרגילים	20%
מעבדות תכנות	35%
מערך תרגול לאמצע הקורס	20%
מערך תרגול מסכם	25%

שאפו לרמה 3 ומעלה בכל יחידה לפני שעוברים לקורס הבא.

קורסים קשורים

- שרטוט הנדסי / CAD ENGR 120
- חשבון אינפיניטסימלי 3 MATH 210
- כימיה כללית + 1 מעבדה CHEM 110 + CHEM 110L
- כל הקורסים