ניתוח ארכיטקטוני ומדריך פונקציונלי מקיף לספריית NumPy

ספריית NumPy (Numerical Python) מהווה את התשתית המרכזית למחשוב מדעי בשפת Python, והיא הכלי היסודי שעליו נשענים כמעט כל כלי ניתוח הנתונים, הלמידה המכונה והחישובים ההנדסיים המודרניים.[1] פיתוחה של הספרייה סיפק פתרון לצורך במבנה נתונים יעיל עבור מערכים רב-ממדיים, המאפשר ביצועים הקרובים לאלו של שפות הידור נמוכות כמו C ו-Fortran, תוך שמירה על תחביר קריא וגמיש.[1] הארכיטקטורה של NumPy אינה מוגבלת רק לאחסון נתונים, אלא כוללת מערך עצום של שגרות מתמטיות, לוגיות, סטטיסטיות ואלגוריתמיות הפועלות בצורה וקטורית על גבי אובייקט הליבה, ה-ndarray.[1]

אובייקטים של מערכים: ה-ndarray והמבנה הפנימי

אובייקט המערך ה-N-ממדי (ndarray) הוא ישות המייצגת אוסף הומוגני של נתונים המאורגנים בזיכרון רציף.[5] בניגוד לרשימות המובנות של Python, שבהן כל אלמנט הוא אובייקט נפרד המכיל מטא-נתונים ותקורת ניהול, ה-ndarray מאחסן את הנתונים כבלוק בינארי גולמי.[1] יעילות זו מושגת באמצעות הפרדה בין הנתונים הפיזיים לבין "המבט" (view) עליהם, המוגדר על ידי פרמטרים של צורה (shape), סוג נתונים (dtype) וצעדים (strides).[1]

הצעדים (Strides) הם קריטיים להבנת הביצועים ב-NumPy; הם מגדירים את מספר הבייטים שיש לדלג בזיכרון כדי לעבור לאלמנט הבא בכל ציר.[1] מנגנון זה מאפשר פעולות כמו שינוי צורה (reshaping) או שחלוף (transposing) להתבצע כמעט באופן מיידי, שכן הן דורשות רק שינוי במטא-נתונים מבלי להעתיק את הנתונים עצמם.[1]

מאפייני יסוד של אובייקט ה-ndarray

מאפיין	תיאור טכני	השלכות ביצועיות ושימוש
ndim	מספר הממדים או הצירים של המערך [5]	קובע את מורכבות הגישה לאינדקסים ואת אפשרויות השידור [6]
shape	טאפל המציין את אורך המערך בכל מימד [5]	שינוי הצורה דרך מאפיין זה אינו משנה את נתוני המקור [8]
size	מספר האלמנטים הכולל במערך [5]	שווה למכפלת ערכי ה-shape; אינדיקטור לצריכת משאבים [8]
dtype	אובייקט המתאר את סוג	קובע את הדיוק הנומרי

	הנתונים של האיברים [5]	(למשל float64 לעומת float32) [8]
itemsize	הגודל בבייטים של כל אלמנט בודד [5]	מאפשר חישוב של צריכת הזיכרון הפיזית [8]
nbytes	סך הזיכרון בבייטים הנצרך על ידי נתוני המערך [8]	שימושי לניהול תקציב זיכרון במערכות עם משאבים מוגבלים [8]
data	מצביע לבלוק הזיכרון שבו שמורים הנתונים [6]	גישה ישירה לזיכרון הגולמי (לשימוש מתקדם בלבד) [6]

שגרות ליצירת מערכים (Array Creation Routines)

NumPy מציעה מגוון רחב של דרכים לייצור מערכים חדשים, המותאמות לצורכי החישוב השונים.[9] ניתן לחלק את השגרות הללו לקטגוריות של המרה ממבנים קיימים, יצירה של מערכים בעלי ערכים קבועים, יצירת רצפים מספריים ובניית מטריצות מיוחדות.[9]

המרת מבני נתונים של Python, כמו רשימות או טאפלים, מתבצעת באמצעות np.array. פעולה זו בונה ndarray חדש ומנסה להסיק את סוג הנתונים המשותף המינימלי הנדרש לייצוג כל האיברים.[9] עבור יישומים הדורשים ביצועים גבוהים יותר, מומלץ להשתמש ב-np.zeros, np.ones, או np.full ליצירת מערכים המאותחלים מראש בערכים מסוימים, ובכך להימנע מהתקורה של יצירת רשימות Python זמניות.[9] הפונקציה np.empty מהירה אף יותר, כיוון שהיא מקצה זיכרון ללא אתחול הערכים, מה שמשאיר בהם נתוני "זבל" אקראיים מהזיכרון.[10]

יצירת מערכים בעלי ערכים קבועים ורצפים

פונקציה	מטרה ושימוש עיקרי	הערות חשובות
zeros	יצירת מערך המלא באפסים [9]	משמש לרוב כמאגר (buffer) לתוצאות חישוב עתידיות [9]
ones	יצירת מערך המלא באחדות [9]	שימושי לאתחול משקולות או מסכות [9]
full	יצירת מערך המלא בערך קבוע מראש [10]	מאפשר גמישות מלאה בערך האתחול [13]
empty	הקצאת זיכרון ללא אתחול ערכים [10]	המהירה ביותר אך דורשת מילוי ידני של ערכים לאחר

		מכן [13]
arange	יצירת רצף מספרים עם צעד קבוע [9]	בדומה ל-range, אך תומך בערכי float (זהירות משגיאות עיגול) [9]
linspace	יצירת מספר איברים מוגדר בטווח מסוים [9]	מבטיח דיוק גבוה יותר בנקודות הקצה לעומת arange [9]
logspace	יצירת רצף מספרים במרווחים לוגריתמיים [10]	חיוני לניתוחי תדרים או סולמות לוגריתמיים [12]
eye	יצירת מטריצת יחידה או מטריצה עם אלכסון מוסט [10]	ניתן לשלוט במיקום האלכסון דרך הפרמטר k [12]
identity	יצירת מטריצת יחידה ריבועית [10]	מקרה פרטי ופשוט יותר של eye [10]

מעבר ליצירה בסיסית, קיימות שגרות ליצירת מטריצות מורכבות יותר כמו np.vander (מטריצת ונדרמונד), המשמשת למודלים של פולינומים וריבועים פחותים.[9] שגרות נוספות כמו np.diag מאפשרות לחלץ אלכסון ממטריצה קיימת או לבנות מטריצה חדשה שבה הוקטור הנתון מוצב כאלכסון.[10]

מניפולציה ושינוי מבנה (Array Manipulation Routines)

לאחר יצירת המערך, הצורך בשינוי הצורה שלו, שילובו עם מערכים אחרים או פיצולו הוא תדיר במחקרי נתונים.[5] NumPy מספקת קבוצה עשירה של פונקציות לשינוי מבנה המערך מבלי לשנות את הנתונים הפיזיים בזיכרון, מה ששומר על יעילות מקסימלית.[15]

פעולת ה-reshape היא אולי הנפוצה ביותר, המאפשרת להגדיר מחדש את ממדי המערך כל עוד מספר האלמנטים הכולל נותר קבוע.[8] כאשר יש צורך להפוך מערך רב-ממדי לוקטור חד-ממדי, ניתן להשתמש ב-np.ravel (המחזיר לרוב view) או ב-np.flatten (המחזיר תמיד העתק).[15] בתחום השחלוף והחלפת הצירים, np.transpose וקיצורו .T הופכים את סדר הממדים, בעוד ש-np.moveaxis ו-np.swapaxes מאפשרים שליטה פרטנית על מיקום צירים ספציפיים.[15]

חיבור, פיצול ותיקון צורה

קטגוריה	פונקציות מפתח	הסבר תפקודי
הצטרפות (Joining)	concatenate, stack, vstack,	שילוב מערכים לאורך צירים

	hstack	קיימים או חדשים [8]
פיצול (Splitting)	split, array_split, hsplit, vsplit	חלוקת מערך למספר תתי-מערכים קטנים יותר [15]
שינוי ממדים	expand_dims, squeeze	הוספה או הסרה של צירי יחידה (axis עם אורך 1) [15]
חזרה (Repeating)	tile, repeat	שכפול נתונים ליצירת מבנים מחזוריים או מורחבים [15]
סידור מחדש	flip, roll, rot90	היפוך סדר איברים או סיבוב מטריצות [15]
שינוי תוכן	append, insert, delete	הוספה או הסרה של איברים (יוצר העתק חדש של המערך) [15]

השימוש ב-np.concatenate דורש שהמערכים יהיו בעלי צורה זהה בכל הממדים פרט לזה שבו מתבצע החיבור.[8] לעומתו, np.stack יוצר מימד חדש בתוצאה, מה שמאפשר למשל לקחת רצף של תמונות דו-ממדיות ולהפוך אותן לכרך (volume) תלת-ממדי.[15] פונקציות כמו np.block מאפשרות לבנות מערך גדול מתוך "בלוקים" קטנים יותר בצורה מטריציונית אינטואיטיבית.[15]

פונקציות מתמטיות ואוניברסליות (ufuncs)

אחד המושגים המרכזיים ב-NumPy הוא הפונקציות האוניברסליות (ufuncs), המבצעות פעולות על מערכים בצורה של אלמנט-אלמנט.[5] ה-ufuncs הן למעשה עטיפות סביב קוד C הממוטב לביצוע מהיר על גבי מערכי נתונים גדולים, מה שמייתר את הצורך בלולאות Python איטיות.[1] כל פונקציה אריתמטית בסיסית, כמו חיבור או כפל, מיושמת כ-ufunc, ותומכת במנגנון ה-Broadcasting (שידור).[5]

מנגנון השידור מאפשר ל-NumPy לבצע פעולות בין מערכים בעלי ממדים שונים תחת חוקים מסוימים: אם הממדים בקצה הימני של הצורה זהים או שאחד מהם הוא 1, המערך הקטן יותר "נמתח" וירטואלית כדי להתאים למערך הגדול.[5] הדבר מאפשר, למשל, להוסיף סקלר לכל איבר במערך, או להחסיר ממוצע של עמודה מכל איבר בעמודה זו ללא צורך בשכפול פיזי של הנתונים.[5]

פירוט פונקציות אריתמטיות וטריגונומטריות

קטגוריה	פונקציות נבחרות	תיאור מתמטי
אריתמטיקה בסיסית	add, subtract, multiply,	ביצוע פעולות חשבון

	divide, power	אלמנט-אלמנט [19]
טריגונומטריה	sin, cos, tan, arcsin, arccos, arctan	עבודה עם רדיאנים; כולל פונקציות למעבר בין מעלות לרדיאנים [19]
היפרבוליות	sinh, cosh, tanh, arcsinh, arccosh, arctanh	פונקציות היפרבוליות וההופכיות שלהן [21]
לוגריתמים וחזקות	exp, log, log10, log2, sqrt, cbrt	חישוב אקספוננציאלי ושורשים בצורה ממוטבת [20]
עיגול (Rounding)	around, floor, ceil, trunc, rint	שליטה על דיוק התוצאות והמרת float לשלמים [20]
טיפול במספרים	absolute, sign, clip, mod, remainder	ערך מוחלט, סימן, הגבלת ערכים לטווח וחישוב שארית [20]

מלבד הפעולות הסטנדרטיות, קיימות פונקציות מתקדמות לטיפול במספרים מרוכבים כמו np.real לחילוץ החלק הממשי, np.imag לחלק המדומה, ו-np.conj לחישוב הצמוד המרוכב.[20] בתחום הסטטיסטיקה והצבירה, פונקציות כמו np.sum, np.prod, np.cumsum (סכום מצטבר) ו-np.cumprod פועלות על המערך כולו או לאורך ציר נבחר, ומאפשרות ניתוח מהיר של מגמות בנתונים.[5]

אלגברה ליניארית (numpy.linalg)

היכולות של NumPy בתחום האלגברה הליניארית הן מהמקיפות והיעילות ביותר בעולם ה-Open Source.[14] המודול numpy.linalg מספק ממשק לספריות ה-Fortran וה-C הוותיקות BLAS ו-LAPACK, המבטיחות ביצועים ברמה עולמית בחישובי מטריצות.[23] בעוד שפעולת הכפל * היא אלמנט-אלמנט, כפל מטריצות (Matrix Multiplication) מתבצע באמצעות האופרטור @ (בגרסאות Python 3.5+) או הפונקציות np.dot ו-np.matmul.[5]

האלגוריתמים הכלולים במודול זה מאפשרים פתרון של בעיות מורכבות, החל מפתרון מערכות משוואות ליניאריות ועד לפירוק מטריצות לגורמים לצורך דחיסת נתונים או הפחתת ממדים.[23]

פונקציות מפתח באלגברה ליניארית

פונקציה	תיאור	שימושים נפוצים

linalg.solve	פתרון מערכת המשוואות $Ax = b$ [23]	פתרון מודלים הנדסיים וכלכליים בצורה יציבה נומרית [23]
linalg.inv	חישוב מטריצה הפוכה [23]	שימושי בתיאוריה, אם כי פחות מומלץ חישובית לעומת solve [23]
linalg.det	חישוב דטרמיננטה [23]	זיהוי מטריצות סינגולריות (שאינן הפיכות) [23]
linalg.eig	חישוב ערכים עצמיים ווקטורים עצמיים [23]	בסיס לניתוח רכיבים עיקריים (PCA) בדינמיקה וסטטיסטיקה [23]
linalg.svd	פירוק לערכים סינגולריים [23]	דחיסת תמונות, המלצת תכנים ועיבוד אותות [23]
linalg.norm	חישוב נורמה של וקטור או מטריצה [23]	מדידת "מרחקים" או שגיאות בחישובים [23]
linalg.cholesky	פירוק חולסקי [23]	יעיל ביותר לפתרון משוואות עם מטריצות סימטריות חיוביות [23]

השגרות ב-linalg תומכות גם בעבודה על "ערימות" (stacks) של מטריצות בו-זמנית, מה שמאפשר להריץ פעולות אלגברה ליניארית על עשרות אלפי מטריצות קטנות בפקודה אחת, תוך ניצול מקסימלי של מקביליות המעבד.[23]

סטטיסטיקה וניתוח נתונים (Statistics Routines)

NumPy מספקת סט מקיף של כלים לניתוח תיאורי של נתונים.[14] שגרות אלו מאפשרות לחשב מדדי מרכז, פיזור, קורלציה והתפלגות בצורה מהירה מאוד על פני מיליוני נקודות נתונים.[24] אחד היתרונות של השגרות הסטטיסטיות ב-NumPy הוא היכולת לציין את הציר (axis) שעליו יבוצע החישוב: לדוגמה, ניתן לחשב את הממוצע של כל שורה בנפרד או של המטריצה כולה.[5]

התמודדות עם נתונים חסרים (NaN - Not a Number) היא אתגר נפוץ, ועבורו NumPy מציעה גרסאות ייעודיות המתעלמות מערכי ה-NaN בחישוב, כגון np.nanmean, np.nanstd ו-np.nanmedian.[24]

מדדים סטטיסטיים עיקריים

מדד	פונקציות	הסבר ושימוש
מדדי מרכז	mean, median, average	ממוצע חשבוני, חציון וממוצע משוקלל [24]
מדדי פיזור	std, var, ptp	סטיית תקן, שונות והפרש בין מקסימום למינימום [24]
אחוזונים וקצוות	min, max, percentile, quantile	זיהוי ערכי קיצון וחישוב אחוזונים (כמו ה-25 או ה-75) [24]
קורלציה וקשרים	corrcoef, cov, correlate	מקדם מתאם פירסון, מטריצת קוריאציה וקורלציה צולבת [24]
היסטוגרמות	histogram, bincount, digitize	ספירת מופעים בטווחים (bins) או ספירת שלמים [14]

פונקציות כמו np.percentile ו-np.quantile הן קריטיות לניתוח התפלגויות לא סימטריות, שבהן הממוצע אינו מייצג נאמנה את הנתונים.[24] בנוסף, np.bincount היא פונקציה יעילה להפליא לספירת מופעים של מספרים שלמים חיוביים, המשמשת לעיתים קרובות בתהליכי קידוד (label encoding).[24]

דגימה אקראית וסימולציות (numpy.random)

החל מגרסאות 1.17 ומעלה, NumPy עברה לארכיטקטורה חדשה ומבוססת-אובייקטים ליצירת מספרים אקראיים, שנועדה להחליף את המצב הגלובלי הישן שהיה מועד לשגיאות בסביבות מקביליות.[6] המערכת החדשה מבוססת על שני רכיבים: ה-BitGenerator, שמייצר זרם של ביטים אקראיים (כמו PCG64 המהווה את ברירת המחדל), וה-Generator, שמשתמש בביטים אלו כדי להפיק דגימות מהתפלגויות מתמטיות שונות.[6]

יצירת מספרים אקראיים מודרנית מתחילה בקריאה ל-np.random.default_rng(seed), המחזירה אובייקט מסוג Generator.[5] שיטה זו מבטיחה הדירות (reproducibility) של תוצאות המבוססת על ה-seed שניתן, תוך שימוש באלגוריתמים סטטיסטיים חזקים יותר מה-Mersenne Twister הישן.[6]

התפלגויות ושגרות אקראיות נפוצות

שיטה ב-Generator	תיאור	הקשר סטטיסטי
integers	מספרים שלמים בטווח מוגדר	התפלגות אחידה דיסקרטית [6]

	[6]	
random	ערכי float בטווח $[0.0, 1.0)$ [6]	התפלגות אחידה רציפה [6]
standard_normal	דגימה מהתפלגות נורמלית ($mu=0, \sigma=1$) [6]	התפלגות גאוסיאנית [6]
choice	בחירה אקראית מתוך מערך נתון [6]	מאפשר דגימה עם או בלי החזרה [6]
permutation	החזרת העתק מעורבב של המערך [13]	פרמוטציה של סדר האיברים
shuffle	ערבוב המערך "במקום" (In-place) [13]	שינוי המערך המקורי ללא העתקה

מלבד אלו, ה-Generator כולל תמיכה בעשרות התפלגויות נוספות כמו Binomial (לניסויי הצלחה/כישלון), Poisson (לאירועים המתרחשים בקצב קבוע), ו-Exponential (לזמן המתנה בין אירועים).[13] עבור יישומים קריפטוגרפיים, NumPy מזהירה ששגרות אלו אינן בטוחות לשימוש וממליצה על מודול ה-secrets של Python.[6]

מיון, חיפוש וספירה (Sorting, Searching & Counting)

היכולת לארגן ולמצוא מידע בתוך מערכים רחבי ממד היא הכרחית לכל תהליך ניתוח נתונים.[31] NumPy מציעה אלגוריתמי מיון ממוטבים שניתן להחיל על המערך כולו או לאורך ציר מסוים.[31] המיון יכול להתבצע "במקום" (in-place) על ידי המטודה ndarray.sort() או על ידי הפונקציה np.sort() המחזירה העתק ממוין.[31]

בחירת אלגוריתם המיון ב-NumPy מתבצעת דרך הפרמטר kind, הכולל אפשרויות כמו quicksort (המהיר לרוב), mergesort (יציב, שומר על סדר המופעים המקורי של ערכים זהים) ו-heapsort.[32]

שגרות מיון וחיפוש

פונקציה	פעולה	יתרון מרכזי
sort	מיון של המערך או ציר [31]	תמיכה במספר אלגוריתמים [35]
argsort	החזרת האינדקסים שהיו ממיינים את המערך [31]	מאפשר למיין מערך אחד לפי ערכים של מערך אחר [32]

lexsort	מיון עקיף יציב לפי מספר מפתחות [31]	פועל בדומה למיון עמודות מרובות בטבלה [32]
partition	מיון חלקי המציב את האיבר ה-k במקומו [31]	יעיל למציאת החציונים או עשרת הערכים הגדולים ביותר [35]
argmax / argmin	מציאת האינדקס של הערך המקסימלי/מינימלי [31]	תומך ב-axis למציאת קיצון בכל שורה/עמודה [36]
where	החזרת אינדקסים או ערכים לפי תנאי לוגי [31]	הכלי העוצמתי ביותר לסינון נתונים (Filtering) [31]
searchsorted	מציאת מיקום להכנסת ערך תוך שמירה על סדר [31]	משתמש בחיפוש בינארי לביצועים מהירים ב-log(n) [31]

בתחום הספירה, הפונקציה np.count_nonzero מאפשרת לספור כמה אלמנטים במערך אינם אפסים (או כמה אלמנטים מקיימים תנאי מסוים, אם מעבירים לה מערך בוליאני).[33] שגרה זו מהירה משמעותית מלולאות Python ומספקת דרך מיידית להעריך את צפיפות הנתונים.[36]

לוגיקה ופעולות בינאריות (Logic and Bitwise Routines)

השכבה הלוגית של NumPy מאפשרת לבצע השוואות ובדיקות תקינות על מערכי ענק בו-זמנית.[38] תוצאה של השוואה (כמו arr > 5) היא תמיד מערך בוליאני בעל צורה זהה למקור, המהווה את הבסיס לשיטת ה-"Boolean Indexing" (אינדוקס בוליאני) שבו משתמשים במערך זה כדי לחלץ תת-קבוצה של הנתונים.[19]

בנוסף ללוגיקה ברמת האלמנט, קיימות פונקציות להפחתה לוגית כמו np.all (האם כל האיברים אמת) ו-np.any (האם קיים איבר אמת), שהן חיוניות לבדיקות תקינות בקוד.[38]

פונקציות לוגיות ותוכן

קטגוריה	פונקציות נבחרות	מטרה
בדיקת תוכן	isnan, isinf, isfinite, isnat	זיהוי ערכים חסרים, אינסופיים או חסרי זמן [38]
בדיקת סוג	iscomplex, isreal, isscalar	סיווג של אלמנטים לפי סוג הנתונים שלהם [38]
אופרטורים לוגיים	logical_and, logical_or,	שילוב של תנאים בוליאניים

	logical_not, logical_xor	מרובים [38]
השוואת מערכים	allclose, array_equal, array_equiv	בדיקת שוויון בין מערכים תוך התחשבות בסובלנות נומרית [38]
פעולות ביטים	bitwise_and, bitwise_or, invert, left_shift	מניפולציה ישירה של ייצוג הביטים במספרים שלמים [41]

השגרות הבינאריות (Bitwise) הן כלי חיוני בעיבוד תמונה, תקשורת נתונים ודחיסה.[41] הן מאפשרות לבצע פעולות AND, OR ו-XOR ברמת הביט הבודד, וכן להזיז ביטים שמאלה וימינה, מה שמהווה לעיתים דרך מהירה לביצוע כפל או חילוק בחזקות של 2.[41]

עבודה עם מחרוזות וטקסט (numpy.strings)

בגרסאות המודרניות של NumPy (2.0 ואילך), המודול numpy.strings החליף את המודול הוותיק numpy.char.[44] מודול זה מספק פונקציות וקטוריות לטיפול במערכי מחרוזות, המאפשרות לבצע פעולות טקסטואליות על מיליוני מחרוזות בו-זמנית ביעילות של קוד C.[44] כל פונקציה במודול פועלת בצורה של אלמנט-אלמנט על המערך הנתון.

פונקציות מחרוזת נבחרות

פונקציה	פעולה
capitalize, upper, lower	שינוי רישיות של כל מחרוזת במערך [40]
strip, lstrip, rstrip	הסרת תווים לבנים או תווים ספציפיים מקצוות המחרוזות [40]
replace	החלפת תת-מחרוזת ישנה בחדשה בתוך כל איבר [40]
split, partition	פיצול של מחרוזות לפי תו מפריד [44]
join	חיבור אלמנטים של מחרוזת עם מפריד ספציפי [44]
find, index, count	חיפוש מיקום של תת-מחרוזת או ספירת מופעים [44]

isalnum, isalpha, isdigit	בדיקה האם המחרוזות מכילות רק תווים מסוג מסוים [44]

המערכות החדשות ב-NumPy תומכות גם במחרוזות באורך משתנה (UTF-8), מה שמשפר את התאימות עם Python המודרנית ועם ספריות כמו Pandas.[44]

פולינומים וסדרות (numpy.polynomial)

הטיפול בפולינומים ב-NumPy עבר מהפכה עם הצגת חבילת ה-numpy.polynomial (החל מגרסה 1.4), שהחליפה את מחלקת poly1d הישנה.[6] החבילה החדשה מציעה "מחלקות נוחות" (convenience classes) לטיפול לא רק בפולינומים רגילים, אלא גם בבסיסים אורתוגונליים כמו Chebyshev, Legendre ו-Hermite, שהם יציבים הרבה יותר מבחינה נומרית בדרגות גבוהות.[45]

אחת התכונות החשובות בחבילה החדשה היא שהמקדמים מסודרים מהדרגה הנמוכה לגבוהה (למשל, `` מייצג את $1 + $2x + 3x^2), מה שתואם את האינדוקס של המערך.[46]

השוואת יכולות פולינומים

תכונה	(legacy (poly1d	(modern (polynomial
סדר מקדמים	גבוה לנמוך [6]	נמוך לגבוה [6]
סוגי סדרות	כוח בלבד [6]	Power, Chebyshev, Legendre, וכו' [45]
התאמת נתונים	np.polyfit [45]	Polynomial.fit [45]
יציבות נומרית	נמוכה בדרגות גבוהות [6]	גבוהה (דרך חלונות ודומיינים) [6]

השגרות המודרניות מאפשרות לבצע פעולות חשבון על פולינומים, לחשב שורשים (roots), גזירה (deriv) ואינטגרציה (integ), וכן לבצע התאמה לנתונים בשיטת הריבועים הפחותים בצורה שהיא "חסינה" לשגיאות עיגול בזכות שימוש בטרנספורמציות של דומיין וחלון (domain and window).[45]

קלט, פלט ואינטראקציה עם קבצים (Input and Output)

ניהול הנתונים ב-NumPy מחייב יכולת לשמור ולטעון מערכים בצורה מהירה ואמינה.[14] הספרייה מציעה פורמטים בינאריים ייעודיים (.npy ו-.npz) שהם מהירים משמעותית מפורמטים מבוססי טקסט, כיוון שהם שומרים את הנתונים בדיוק כפי שהם בזיכרון ה-RAM.[49]

עבור שיתוף פעולה עם תוכנות אחרות או קריאת נתונים גולמיים, NumPy מספקת שגרות לקריאת

קבצי טקסט ו-CSV.[49]

שגרות I/O נבחרות

פונקציה	פורמט	הערות ושימוש
save	.npy (בינארי)	שמירת מערך יחיד בפורמט בינארי פשוט [49]
savez / savez_compressed	.npz (ארכיון)	שמירת מספר מערכים בקובץ אחד (עם או בלי דחיסה) [49]
load	.npz. / npy	טעינת מערכים שנשמרו בפורמט בינארי [49]
loadtxt	טקסט / CSV	טעינה מהירה של קבצי טקסט בעלי מבנה קבוע [10]
savetxt	טקסט / CSV	שמירת מערך בפורמט טקסטואלי קריא לאדם [49]
genfromtxt	טקסט מורכב	טעינה עם טיפול בערכים חסרים ועמודות מעורבות [49]
fromfile / tofile	בינארי גולמי	קריאה/כתיבה ללא מטא-נתונים (דורש ידיעת ה-dtype) [10]
memmap	מיפוי זיכרון	עבודה עם קבצי ענק על הדיסק כאילו היו בזיכרון [49]

השימוש ב-np.memmap הוא קריטי לעיבוד נתונים בנפחים של "Big Data" החורגים מגודל הזיכרון הזמין, שכן הוא מאפשר למערכת ההפעלה לנהל את טעינת הדפים מהדיסק בצורה שקופה למשתמש.[49]

מערכים ממוסכים (Masked Arrays - numpy.ma)

במקרים רבים של איסוף נתונים מהעולם האמיתי, חלק מהמידע עשוי להיות חסר, פגום או לא רלוונטי.[50] המודול numpy.ma מספק את מחלקת ה-MaskedArray, המשלבת מערך נתונים רגיל עם מסכה בוליאנית באותה צורה.[50] כאשר המסכה מכילה את הערך True במיקום מסוים, האלמנט התואם נחשב ל-"masked" (ממוסך), ושגרות NumPy יתעלמו ממנו אוטומטית בחישובי ממוצעים, סכומים

וכו'.[51]

שגרות מפתח ב-Masked Arrays

פונקציה	פעולה
ma.array	יצירת מערך ממוסך עם הגדרת נתונים ומסכה [50]
masked_invalid	מיסוך אוטומטי של ערכי NaN ואינסוף [50]
masked_equal	מיסוך של ערכים השווים לערך ספציפי [50]
masked_inside / outside	מיסוך ערכים בתוך או מחוץ לטווח מסוים [50]
getmask / getdata	חילוץ המסכה או הנתונים הגולמיים מהמערך הממוסך [50]
filled	החזרת המערך כאשר הערכים הממוסכים מוחלפים בערך קבוע [51]

מערכים ממוסכים הם כלי רב עוצמה המאפשר לשמור על שלמות המבנה של המערך (למשל, תמונה שנשארת דו-ממדית) תוך התעלמות מפיקסלים "שרופים" או נתונים חסרים, מבלי להזדקק למחיקה פיזית של שורות או עמודות.[50]

תמיכה בזמן ותאריכים (Datetime and Timedelta)

החל מגרסה 1.7, NumPy תומכת בסוגי נתונים המיועדים לטיפול בזמן: datetime64 ו-timedelta64.[55] סוגי נתונים אלו ממוטבים לעבודה וקטורית מהירה, והם מאפשרים חישובים אריתמטיים ישירים על תאריכים (למשל, החסרת תאריך אחד ממשנהו לקבלת משך זמן).[55]

המאפיין הייחודי של datetime64 הוא הדיוק המשתנה (Precision); ניתן להגדיר תאריכים ברמת יום, שעה, מילי-שנייה או אפילו אטו-שנייה (10^{-18} שניות).[55]

שגרות זמן ועבודה עם ימי עסקים

פונקציה	תיאור
datetime_as_string	המרת מערך תאריכים למחרוזות בפורמט ISO

	8601 [58]
busday_count	ספירת ימי עסקים בין שני תאריכים [55]
is_busday	בדיקה האם תאריך מסוים הוא יום עסקים [55]
busday_offset	הוספת ימי עסקים לתאריך תוך התחשבות בסופי שבוע וחגים [58]
datetime_data	קבלת מידע על יחידות הזמן של המערך [58]

מנגנון ימי העסקים מאפשר למשתמש להגדיר "מסכות שבוע" (weekmasks) וחגים מותאמים אישית, מה שהופך את NumPy לכלי עבודה חשוב ביישומים פיננסיים וניהול פרויקטים.[55]

תכנות פונקציונלי וכלים מתקדמים

NumPy מספקת מספר שגרות המאפשרות ליישם עקרונות של תכנות פונקציונלי על גבי מערכים, מה שמקל על כתיבת קוד נקי וקצר.[6] הפונקציה np.vectorize היא אולי הידועה ביותר; היא לוקחת פונקציית Python רגילה הפועלת על סקלרים וממירה אותה לפונקציה הפועלת על מערכים, תוך שימוש בחוקי ה-Broadcasting.[6] חשוב לציין ש-vectorize נועדה לנוחות בלבד ואינה משפרת את הביצועים כמו ufunc אמיתית, כיוון שהיא עדיין מריצה לולאת Python פנימית.[6]

כלי יישום וקטורי

פונקציה	פעולה
apply_along_axis	הפעלת פונקציה על פרוסות (slices) של המערך לאורך ציר נבחר [6]
apply_over_axes	הפעלת פונקציה שוב ושוב על מספר צירים [6]
frompyfunc	יצירת ufunc מותאמת אישית מכל פונקציית Python [6]
piecewise	הערכה של פונקציה בצורה של "חלקים" לפי תנאים שונים [6]

השימוש ב-np.apply_along_axis הוא נפוץ במיוחד כאשר רוצים להפעיל פונקציה מורכבת (שאינה וקטורית במקור) על כל שורה במטריצה, מבלי לכתוב לולאת for מפורשת, מה ששומר על מבנה קוד נקי יותר.[6]

סיכום ותובנות מערכתיות

הניתוח המעמיק של פונקציות ומטודות NumPy חושף ספרייה שהיא הרבה מעבר לאוסף של שגרות מתמטיות. זוהי מערכת שלמה המתוכננת סביב יעילות בזיכרון וביצועים חישוביים.[1] המעבר ל-NumPy 2.x וההתמקדות באובייקט ה-Generator ליצירת מספרים אקראיים, במודול ה-strings החדש ובחבילת הפולינומים היציבה, מעידים על תהליך של מודרניזציה המותאם לצורכי המחקר המדעי במאה ה-21.[6]

עבור איש המקצוע, ה-API של NumPy מציע שלוש רמות של עבודה: רמת המניפולציה הבסיסית (יצירה ושינוי צורה), רמת החישוב הוקטורי (ufuncs ושידור), ורמת האלגוריתמיקה המתקדמת (אלגברה ליניארית, FFT וסטטיסטיקה).[5] הבנה עמוקה של יחסי הגומלין בין המטא-נתונים של המערך (כמו strides) לבין הפעולות המתבצעות עליו היא המפתח לכתיבת קוד שאינו רק נכון, אלא גם מהיר וחסכוני במשאבים. ככל ששפת Python ממשיכה להתבסס כסטנדרט בתחומי ה-AI ומדעי הנתונים, NumPy נשארת הלב הפועם המאפשר את כל הקדמה הזו.[2]

Works cited

1. NumPy - Wikipedia, accessed on January 7, 2026, https://en.wikipedia.org/wiki/NumPy
2. NumPy documentation — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.4/
3. numpy/numpy: The fundamental package for scientific ... - GitHub, accessed on January 7, 2026, https://github.com/numpy/numpy
4. Libraries in Python - GeeksforGeeks, accessed on January 7, 2026, https://www.geeksforgeeks.org/python/libraries-in-python/
5. NumPy quickstart — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/user/quickstart.html
6. NumPy reference — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/reference/index.html
7. NumPy Reference, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.11.0/numpy-ref-1.11.0.pdf
8. The Basics of NumPy Arrays | Python Data Science Handbook, accessed on January 7, 2026, https://jakevdp.github.io/PythonDataScienceHandbook/02.02-the-basics-of-numpy-arrays.html
9. Array creation — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/user/basics.creation.html
10. Array creation routines — NumPy v2.5.dev0 Manual, accessed on January 7, 2026, https://numpy.org/devdocs/reference/routines.array-creation.html
11. Array creation — NumPy v1.10 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.10.1/user/basics.creation.html
12. Array creation routines — NumPy v1.6 Manual (DRAFT), accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.6.0/reference/routines.array-creation.html

13. NumPy - Array Creation Routines - Tutorials Point, accessed on January 7, 2026, https://www.tutorialspoint.com/numpy/numpy_array_creation_routines.htm
14. Routines — NumPy v1.15 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.15.4/reference/routines.html
15. Array manipulation routines — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/routines.array-manipulation.html
16. Array manipulation routines — NumPy v1.8 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.8.0/reference/routines.array-manipulation.html
17. NumPy reference — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/reference/
18. 3. Strings, Lists, Arrays, and Dictionaries - NYU Physics department, accessed on January 7, 2026, https://physics.nyu.edu/pine/pymanual/html/chap3/chap3_arrays.html
19. NumPy Math Functions (With Examples) - Programiz, accessed on January 7, 2026, https://www.programiz.com/python-programming/numpy/math-functions
20. Mathematical functions — NumPy v1.10 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.10.1/reference/routines.math.html
21. Mathematical functions — NumPy v2.2 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.2/reference/routines.math.html
22. Mathematical functions — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/routines.math.html
23. Linear algebra (numpy.linalg), accessed on January 7, 2026, https://numpy.org/doc/2.2/reference/routines.linalg.html
24. Statistics — NumPy v2.1 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.1/reference/routines.statistics.html
25. NumPy Statistical Functions (With Examples) - Programiz, accessed on January 7, 2026, https://www.programiz.com/python-programming/numpy/statistical-functions
26. 006-numpy-statistical-functions.md - GitHub, accessed on January 7, 2026, https://github.com/tinitiate/python-numpy/blob/master/006-numpy-statistical-functions.md
27. mindspore.numpy.bitwise_and, accessed on January 7, 2026, https://www.mindspore.cn/docs/en/r1.9/api_python/numpy/mindspore.numpy.bitwise_and.html
28. Random Sampling — Xorbits 0.8.2+9.g8c60cb6 documentation, accessed on January 7, 2026, https://xorbits.readthedocs.io/en/latest/reference/numpy/random.html
29. NumPy v2.2 Manual - random.Generator., accessed on January 7, 2026, https://lira.no-ip.org:8443/doc/python-numpy/html/reference/random/generated/numpy.random.Generator.random.html
30. Statistical functions (scipy.stats) — SciPy v1.16.2 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/scipy/reference/stats.html
31. Numpy Sorting, Searching, and Counting - TechVidvan, accessed on January 7, 2026, https://techvidvan.com/tutorials/numpy-sorting-searching-and-counting/

32. NumPy - Sort, Search & Counting Functions - Tutorials Point, accessed on January 7, 2026, https://www.tutorialspoint.com/numpy/numpy_sort_search_counting_functions.htm
33. Sorting, searching, and counting — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/routines.sort.html
34. Numpy | Sorting, Searching and Counting - GeeksforGeeks, accessed on January 7, 2026, https://www.geeksforgeeks.org/numpy/numpy-sorting-searching-and-counting/
35. numpy.sort — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/reference/generated/numpy.sort.html
36. NumPy Sort, Search and Count Functions - DataFlair, accessed on January 7, 2026, https://data-flair.training/blogs/numpy-search-sort-and-count/
37. Comprehensive list of NumPy functions with example code · GitHub, accessed on January 7, 2026, https://gist.github.com/MBoustani/344feb83ae238be931a79dc2431926d9
38. Logic functions — NumPy v1.8 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc//numpy-1.8.1/reference/routines.logic.html
39. Logic functions — NumPy v2.2 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.2/reference/routines.logic.html
40. Python Numpy Tutorial (with Jupyter and Colab), accessed on January 7, 2026, https://cs231n.github.io/python-numpy-tutorial/
41. Bit-wise operations — NumPy v2.1 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.1/reference/routines.bitwise.html
42. NumPy Bitwise Operations (With Examples) - Programiz, accessed on January 7, 2026, https://www.programiz.com/python-programming/numpy/bitwise-operations
43. Bitwise Operators in Python, accessed on January 7, 2026, https://realpython.com/python-bitwise-operators/
44. String functionality — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.org/doc/stable/reference/routines.strings.html
45. Polynomials — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/routines.polynomials.html
46. routines.polynomials.txt - Numpy and Scipy Documentation, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.7.0/_sources/reference/routines.polynomials.txt
47. numpy.polynomial — NumPy v2.5.dev0 Manual, accessed on January 7, 2026, https://numpy.org/devdocs/reference/routines.polynomials-package.html
48. Polynomial Package — NumPy v1.15 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc//numpy-1.15.0/reference/routines.polynomials.package.html
49. Input and output — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/routines.io.html
50. The numpy.ma module, accessed on January 7, 2026,

https://numpy.org/doc/stable/reference/maskedarray.generic.html
51. Masked arrays — NumPy v2.3 Manual, accessed on January 7, 2026, https://numpy.org/doc/2.3/reference/maskedarray.html
52. Masked array operations — NumPy v1.3 Manual (DRAFT), accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.3.x/reference/routines.ma.html
53. Masked arrays — NumPy v1.11 Manual, accessed on January 7, 2026, https://docs.scipy.org/doc/numpy-1.11.0/reference/maskedarray.html
54. Masked array operations — NumPy v2.5.dev0 Manual, accessed on January 7, 2026, https://numpy.org/devdocs/reference/routines.ma.html
55. Datetimes and timedeltas — NumPy v2.4 Manual, accessed on January 7, 2026, https://numpy.com.cn/doc/stable/reference/arrays.datetime.html
56. numpy::datetime - Rust - Docs.rs, accessed on January 7, 2026, https://docs.rs/numpy/latest/numpy/datetime/index.html
57. python - Converting between datetime, Timestamp and datetime64, accessed on January 7, 2026, https://stackoverflow.com/questions/13703720/converting-between-datetime-timestamp-and-datetime64
58. Datetime support functions — NumPy v2.5.dev0 Manual, accessed on January 7, 2026, https://numpy.org/devdocs/reference/routines.datetime.html