



תכנות מערכות דפנסיבי (20605)

רועי מימרן

1.5

משתנים בסיסיים

שמירת מידע במחשב

◀ יחידת המידע הבסיסית במחשב היא סיבית (bit). לכל סיבית שני מצבים:
0 או 1

◀ הסיביות מאורגנות בקבוצות של 8, כל אחת נקראת בית (בייט, byte) מכיוון שכל בית מורכב מ-8 סיביות, שכל אחת מהן יכולה להיות בשני מצבים, אז מספר הצירופים האפשריים של ערכים שונים שאפשר לאחסן בבית אחד הוא:
 $2^8 = 256$

◀ כיום יש במחשבים כמויות גדולות של בתים:

◀ $2^{10} = 1,024$ בתים נקראים קילו-בייט ומסומנים KB

◀ $2^{20} = 1,048,576$ בתים נקראים מגה-בייט ומסומנים MB

◀ 2^{30} בתים, שהם בעצם 1024 מגה-בייט נקראים ג'יגה-בייט ומסומנים GB

מבנה הזיכרון הראשי

כתובות

4802

4803

4804

4805

4806

4807

4808

4809

4810

4811

4812

4813



לכל בית בזיכרון יש כתובת
שמיוצגת על ידי מספר

נתונים גדולים מאוחסנים
במספר ביותים רציפים

כיצד ניגשים לתאי זיכרון?

- ◀ במהלך עבודת התוכנית אנחנו רוצים לגשת לתאי זיכרון, לשמור בהם מידע, לשלוף מידע או לעדכן אותו תוך כדי פעולת התוכנית
- ◀ בשפות נמוכות (או שפות סף) ובייחוד בעבודה עם מערכות משובצות מחשב, קיימת אפשרות לכתוב ולקרוא מידע ישירות מכתובת מסוימת בזיכרון או אליה
- ◀ מהן הבעיות בשיטה הזו?
- ◀ לעיתים קרובות נרצה שתוכנית תוכל לפעול באופן דומה במחשבים שונים ובמערכות הפעלה שונות. מרחב הכתובות בהן יהיה שונה והכתובות יהיו שונות
- ◀ במחשב בו יש מערכת הפעלה ומספר רב של תהליכים ותוכנות, או אפליקציות הפועלות במקביל, מערכת ההפעלה מקצה את הזיכרון במחשב, והתוכנית יכולה בכל פעם לקבל כתובות אחרות בזיכרון

שמירת מידע בשפות עיליות

- ◀ ברוב שפות התוכנות העיליות, כולל פייתון, נהוג לשמור מידע בתוכנית בעזרת משתנים
- ◀ בזמן ריצת התוכנית, מערכת ההפעלה מקצה למפרש את כמות הזיכרון שלה הוא זקוק, והמפרש מקצה לכל משתנה את הכתובת בזיכרון
- ◀ כאשר מבצעים הגדרה או השמה של משתנה חדש, המפרש מקצה לו מקום חדש בזיכרון
- ◀ כאשר התוכנית משתמשת בשם משתנה שכבר קיים, המפרש מזהה את הכתובת שהוקצתה לו וניגש לאותה כתובת
- ◀ הכתובת בזיכרון יכולה להשתנות בכל הפעלה נוספת של אותה תוכנה, וכמובן במחשב אחר יכולה גם להיות כתובת אחרת
- ◀ בשפת פייתון, סוג המשתנה יכול להשתנות תוך כדי התוכנית (מספר שלם, מחרוזת, רשימה או משתנה מורכב יותר) ולכן גם גודל המידע (מספר הבתים שנדרשים לאחסן אותו), הכתובת יכולה להשתנות בזמן ריצת התוכנית

אובייקטים וטיפוסים

- ◀ בשפת פייתון, חלק גדול מהתוכנית עוסק באובייקטים (objects)
- ◀ לכל אובייקט יש טיפוס (type) שלו האובייקט שייך
- ◀ יש טיפוסים פשוטים, ויש טיפוסים מורכבים יותר שלעיתים אפשר לבנות אותם מסדרה של איברים פשוטים
 - ◀ int (מספר שלם)
 - ◀ float (מספר עם נקודה עשרונית)
 - ◀ bool (ערך לוגי: אמת או שקר, או נכון או לא נכון, בפייתון: True / False)
 - ◀ None – טיפוס שמייצג נתון ריק
 - ◀ str (מחרוזת) הוא מורכב יותר כי הוא עשוי מרצף של תווים

ביטויים מספריים

- ◀ **ביטויים** (expressions) הם שילוב של אובייקטים ושל אופרטורים
- ◀ **אופרטורים** (operators) הם סימנים שמבצעים פעולות, ומחזירים ערך (value) שגם הוא אובייקט מטיפוס מסוים
- ◀ לדוגמה, האופרטור == מבצע השוואה, הוא בודק אם הערך מימינו שווה לערך משמאלו. האופרטור != (לא שווה) יחזיר תמיד תשובה הפוכה, ויהיה True אם יש הבדל בין הערכים

נראה כעת את האופרטורים שמבצעים פעולות חישוב על מספרים.

אופרטורים על מספרים

◀ $x+y$ - חיבור

◀ $x-y$ - חיסור

◀ $x*y$ - כפל

◀ $x//y$ - חילוק בלי שארית (הערך השלם בלבד). לא ניתן לבצע אם y הוא 0

◀ x/y - חילוק מלא (מחזיר ערך מסוג float). לא ניתן לבצע אם y הוא 0

◀ $x\%y$ - השארית בחילוק. נקרא: x modulo y או בקיצור $x \bmod y$

◀ $x**y$ - חזקה

עבודה עם ערכים וטיפוסים

Command Prompt - python

```
C:\>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3
3
>>> 3+4
7
>>> 3.0+4.0
7.0
>>> 5!=6
True
>>> type(5)
<class 'int'>
>>> type(6)
<class 'int'>
>>> type(6.0)
<class 'float'>
>>> type(5!=6)
<class 'bool'>
>>>
```

משתנים והשמה

◀ משתנים הם שמות עבור אובייקטים הנשמרים בזיכרון. ניתן ליצור משתנים בעזרת פקודות השמה (assignment):

- ▶ `pi = 3.14159`
- ▶ `radius = 11`
- ▶ `area = pi * (radius ** 2)`

◀ אם נציב ערך אחר במשתנה שכבר הוגדר, הערך הקודם במשתנה יבוטל. שימו לב שזה לא משפיע על הערך המאוחסן במשתנה `area`:

- ▶ `radius = 14`
- ▶ `text = "This is the area"`

סדר הפעולות בחישוב

◀ אם נרצה לחשב ממוצע בין שלושה ציונים, מה יקרה אם:

- ▶ $grade1 = 80$
- ▶ $grade2 = 96$
- ▶ $grade3 = 74$
- ▶ $average = grade1 + grade2 + grade3 / 3$
- ▶ $average2 = (grade1 + grade2 + grade3) / 3$

◀ כל השורות בדוגמה הזו הן פעולות השמה, שבה מציבים ערך לתוך משתנה וכך מאחסנים אותו בזיכרון

השמה תוספתית (Incremental Assignment)

פעולות שלוקחות ערך קיים במשתנה ומעדכנות אותו בעזרת פעולה עם ערך נוסף. למשל, אפשר לקחת ערך קיים ולהוסיף לו מספר, להחסיר ממנו ערך, להכפיל וכן רוב הפעולות הנוספות. סימון פעולות אלה בעזרת סימן הפעולה ביחד עם סימן ההשמה = (רצוף), ולאחריו הערך שאיתו מבצעים את הפעולה. למשל:

הוספת 1 למשתנה a: $a += 1$

החסרת הערך b מהמשתנה a: $a -= b$

להוסיף מע"מ לסכום c בעזרת כפל ב-1.17: $c^* = 1.17$

העלאת המשתנה f בריבוע: $f^{**} = 2$

השמת מספר משתנים בשורה אחת

◀ ניתן לבצע מספר פעולות השמה בשורה אחת, וכך התוכנית תהיה קצרה יותר:

```
grade1, grade2, grade3 = 80, 96, 74
```

◀ כדי שהתוכנית תישאר קריאה, מומלץ לעשות זאת רק אם המשתנים דומים בשמות ובסוג הערכים שלהם. ניתן גם להציב ערך אחד במספר משתנים ביחד:

```
x = y = z = "Orange"
```

◀ אפשר כך גם להחליף בין ערכים של משתנים:

```
grade1, grade2 = grade2, grade1
```

כללים לגבי שמות משתנים

- ◀ שמות משתנים יכולים להכיל אותיות לועזיות גדולות וקטנות, ספרות, ואת סימן הקו התחתון `_` (underscore). עם זאת שם המשתנה לא יכול להתחיל בספרה. למשל: `a3` הוא שם תקין למשתנה, אבל לא `3a`
- ◀ שימו לב שפייתון מבדילה בין אותיות לועזיות גדולות וקטנות. כלומר המשתנים `x` ו-`X` (אות קטנה וגדולה) הם שני משתנים שונים שאין ביניהם קשר
- ◀ אם רוצים לתת שם משתנה המורכב משתי מילים, יש לחבר אותן:
`GoodGrade` או `good_grade`
- ◀ בכל שפת תכנות יש **מילים שמורות** (reserved keywords) שיש להן משמעות מיוחדת בשפה. אסור להשתמש במילים אלו כשמות משתנים, שימוש בהן עלול לשבש הגדרות במפרש

רשימת המילים השמורות בפייתון

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

משתנים – שמות סתמיים או משמעותיים

- ▶ $a = 3.14159$
- ▶ $b = 11.2$
- ▶ $c = a * (b ** 2)$

- ▶ $pi = 3.14159$
- ▶ $diameter = 11.2$
- ▶ $area = pi * (diameter ** 2)$