



חעי מימרן

1.7

פקודות קלט

פקודת קלט (input)

- ▶ הפקודה העיקרית שנשתמש בה כדי לקבל קלט מהמשתמש היא הפונקציה `input`.
- ▶ הפונקציה יכולה לכתוב הודעה למשתמש בתחילת השורה, כדי שיבין שהוא אמור לכתוב קלט מסוים (הודעה זו נקראת `prompt`), התכנית נעצרת עד שהמשתמש מקיש על המקלדת ואז לוחץ על `Enter`.
- ▶ הפונקציה מחזירה לנו מחרוזת, שאותה אפשר כמובן לשמור במשתנה, או להמיר לסוג מידע אחר (למשל מספר). המחרוזת לא כוללת את התו `\n` (ירידת שורה, מקש `Enter`) שהוקש בסוף.

דוגמה לפקודת קלט

- ▶ `name1 = input ("Enter your account number please: ")`
- ▶ **Enter you account number please: 24568785**
- ▶ `sum1 = int (input("Enter how many dollars you wish to withdraw: "))`
- ▶ **Enter how many dollars you wish to withdraw: 400**
- ▶ `print ("You asked to withdraw", sum1, "dollars.")`
- ▶ `confirm = input ("Is this correct? (Y/N)")`
- ▶ **You asked to withdraw 400 dollars.
Is this correct? (Y/N) Y**
- ▶ `if confirm[0]== 'y' or confirm[0]== 'Y':`

כיצד מקודדת הקשת תו במקלדת לזכרון?

- ▶ קיימות טבלאות המתרגמות מקשים שכיחים במקלדת, כגון אותיות אנגליות (גדולות וקטנות), ספרות, סימנים שונים שחלקם נמצאים על המקלדת, ותוי בקרה שונים (כגון Enter, Tab וכו') למספר בין 0-127.
- ▶ באופן הסטורי, רוב המחשבים עובדים על פי טבלה שנקראת **קוד ASCII** - ראשי תיבות של American Standard Code for Information Interchange.
- ▶ לדוגמא, תו רווח (Space) הוא מספר 32, * היא מספר 42, הספרות 0-9 הן מספר 48-57 בהתאמה, A מס' 65 ומשם האותיות הגדולות שאחריה. a לעומת זאת היא מס' 97 ואחריה שאר האותיות האנגליות הקטנות.
- ▶ לאחר מכן החלו להשתמש במספרים 128-255 לסימנים ואותיות לא-לטיניות, גם בעברית. אבל זה דורש לבחור את שיטת הקידוד. כל תו ASCII תופס בית אחד.

קידוד עברית בתווי ASCII

- ▶ קיימות מספר שיטות לקודד עברית בתווי ASCII. השיטה הנפוצה ביותר בשנים האחרונות נקראת Windows-1255, האות א' היא מתורגמת למספר 224 עד האות ת' שהיא מס' 250.
- ▶ הבעיה: שצריך להסכים על שיטת הקידוד. כמו כן, לא ניתן לשלב טקסטים עם אותיות לא-לטיניות ביותר משפה אחת (יוונית, עברית, ערבית וכו') בקובץ טקסט אחד או בדף אינטרנט אחד, כי הם מקודדים (לפי טבלאות שונות) לאותם מספרים.
- ▶ לכן התחילו לעבוד עם שיטות קידוד שבהן כל תו יכול לתפוס יותר מבית אחד. השיטות האלה נקראת Unicode (הנפוצה ביותר מתוכן נקראת UTF-8) שבהן כל תו מאוחסן ב-4-1 בתים. זה מאפשר שימוש בשפות שונות בלי "התנגשות".

תרגיל

- ▶ נכתוב תכנית שמבקשת מהמשתמש שם ומספר תעודת זהות, ולאחר מכן כותבת משפט מסכם ושומרת אותו במשתנה user.
 - ▶ אם השם ריק (אורך 0) זו כנראה טעות והיא תבקש שוב מהמשתמש.
 - ▶ אם השם מעל 20 תוים, היא תשמור רק את 20 הראשונים בגלל מגבלת מקום.
 - ▶ אם מספר הזהות ריק או מעל 10 תוים, זו טעות והיא תבקש שוב מהמשתמש.
 - ▶ אם מספר התוים גדול מ-0 אבל קטן מ-9, היא תשלים אפסים בצד שמאל כדי שיהיו 9 ספרות.
 - ▶ בסוף המשתנה user אמור להיראות כמו בדוגמה:
- ▶ Full name: Moshe Levy, Id: 000603789

פתרון חלק 1 – קליטת שם

- ▶ `name = input("Please enter your name:")`
- ▶ `if len(name)==0:`
 - ▶ `name = input ("Empty name, please enter again:")`
- ▶ `If len(name)>20:`
 - ▶ `name = name[0:20]`

פתרון חלק 2 – קליטת מספר זהות

- ▶ `id = input("Please enter your id:")`
- ▶ `if len(id)==0 or len(id)>9:`
 - ▶ `id = input("Wrong number of digits, please enter your id again:")`
- ▶ `if len(id)<9:`
 - ▶ `id = '0' * (9-len(id)) + id`

שלב 3 – בניית המחרוזת `user`

- ▶ מכיוון שהמחרוזת כוללת שילוב של טקסט ידוע מראש עם ערכים משתנים, נשתמש ב-F-strings שהכרנו בשיעור הקודם:
- ▶ `user = f"Full name: {name}, Id: {id}"`
- ▶ ולא נותר אלא להדפיס את המחרוזת לבדיקה:
- ▶ `print (user)`