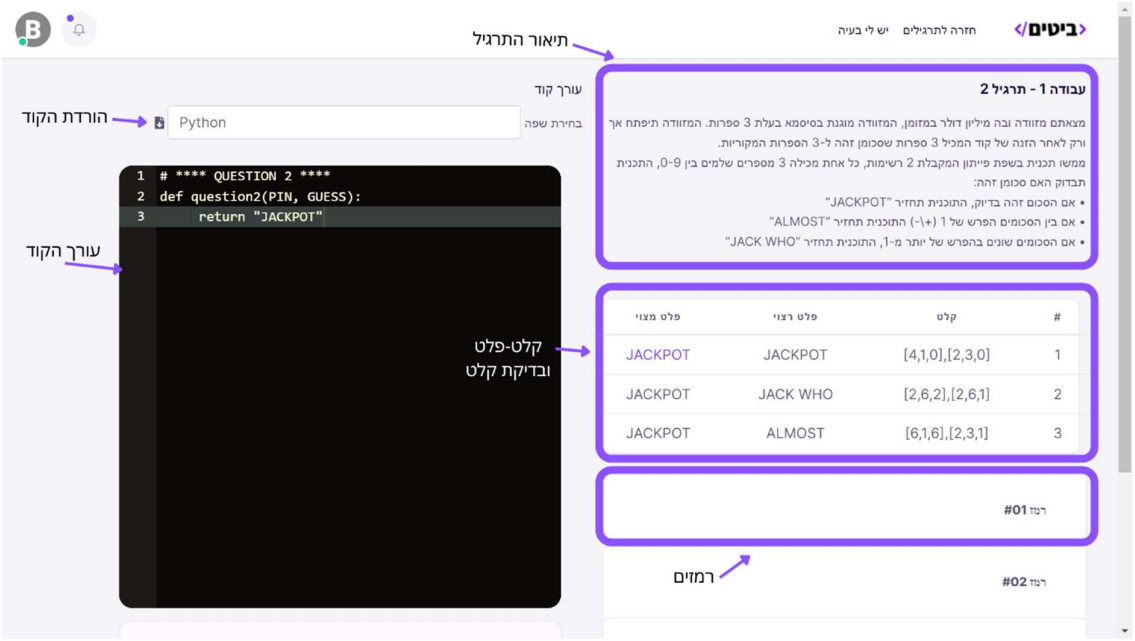
עבודת בית מס' 2:

הנחיות כלליות:

- קראו את כל ההוראות לגבי הגשת תרגילי הבית באתר הקורס.
- קראו את כל העבודה לפני שתתחילו לפתור אותה.
- תאריך פרסום: 02.05.22
- תאריך הגשת התרגיל: 12.05.22 בשעה 23:59.
- יתאפשר איחורי הגשות בסך כולל של 5 ימים עבור כל התרגילים בסמסטר, לתרגיל שלא יוגש במסגרת זמן זה, יינתן ציון 0.
- כתבו תיעוד (הערות) שמסביר את הקוד שלכם. ***אסור לכתוב הערות בעברית!***
- שאלות בנוגע לעבודה יישאלו בפורום המתאים במודל או בשעות הקבלה.
- את הקוד ניתן לכתוב ב-pyCharm כפי שהוסבר בתרגולים, או ישירות באתר ההגשה, הגשת העבודות תהיה באתר ההגשה.
- השימוש בחבילות מוכנות של פייתון **אסור בהחלט** ויגרור ציון 0.
- על הפלטים להיות **בדיוק** כפי שמוגדרים בשאלות (ללא רווחים מיותרים).

מערכת הבדיקות וההגשה האינטרנטית:

במערכת האינטרנטית תוכלו לצפות בתרגילי הבית, לבצע בדיקות אוטומטיות של הקוד למול קלטים מסוימים ולראות כיצד הקוד שלכם מתנהג למול קלטים אלו.

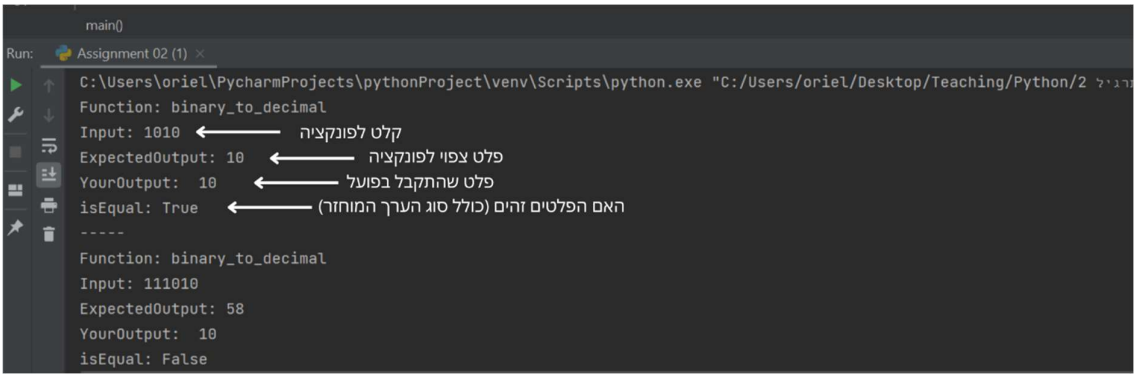


בדיקת הקוד:

בדיקת הקוד תתבצע בלחיצה על כפתור "בדיקה", המערכת תריץ את הקוד שרשמתם אל מול הקלטים המוגדרים ותציג בטבלת הקלט-פלט את הפלט המצוי אל מול הפלט הרצוי, פלט נכון ייצבע בסגול, יש לשים לב כי פלט שלא ייצבע בסגול אינו נכון, גם שגיאה בתו בודד תגרום לפלט שגוי.

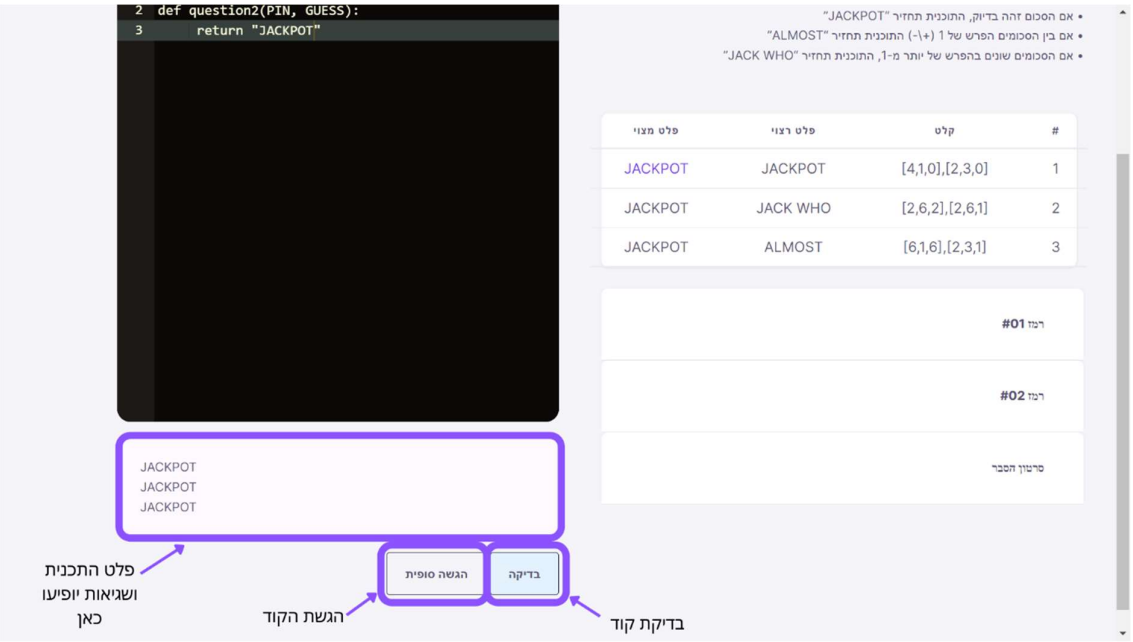
בדיקות ידניות:

לתרגיל זה מצורף קובץ בדיקות ידני, בעזרתו תוכלו לבדוק גם ב-PyCharm את הפונקציות שתממשו בתרגיל. הפונקציות ירוצו למול קלטים שהוגדרו מראש וידפיסו לקונסולה את הקלט שהתקבל, הפלט הצפוי והפלט שלכם באופן הבא:



הגשת התרגיל:

להגשת הקוד, יש ללחוץ על כפתור "הגשה סופית", עם הגשה מוצלחת תופיע הודעה המסמלת על כך.
כמות ההגשות אינה מוגבלת, כך שההגשה האחרונה תהא ההגשה שתיבדק, אנו ממליצים להוריד את קובץ הקוד לצורך גיבוי (ע"י לחיצה על כפתור ההורדה).



שאלה 1:

הנחיות כלליות לסעיפים 1-4:

- אין להמיר מספר בייצוג בינארי ממחרוזת למספר (int) לצורך ביצוע המרה או פעולה לוגית!.
- בהמרה מבסיס בינארי לאחד הבסיסים 8 או 16, יהיה עליכם להמיר את המספר הבינארי ראשית לבסיס דצימאלי(10) ורק לאחר מכן לבסיס היעד (8\16) לדוגמא עבור המספר $(1010)_2$, נמיר ל- $(10)_{10}$ ולאחר מכן ל- $(A)_{16}$.
- אין להשתמש בפונקציות קיימות בפייתון – int(), bin(), hex(), oct() להמרת בסיסים, או לריפוד אפסים – zfill().

בשאלה זו יהיה עליכם לכתוב תכנית בפייתון המבצעת המרות בין הבסיסים השונים (2,10,16,8).

עליכם לממש את הפונקציות הבאות:

1. binary_to_decimal(binary_as_str) – הפונקציה מקבלת כקלט מספר בייצוג בינארי (בסיס 2) **בצורת מחרוזת** ומחזירה את המספר בייצוג עשרוני כ-**int**. לדוגמא: עבור הקלט "1010", יוחזר הפלט 10.

2. decimal_to_binary(decimal_num) – הפונקציה מקבלת כקלט מספר בבסיס עשרוני (בסיס 10) בצורת **int** ומחזירה את המספר בייצוג בינארי (בסיס 2) **כמחרוזת.** לדוגמא: עבור הקלט 10, יוחזר הפלט "1010".

3. binary_to_oct_or_hexa(binary_as_str, dest_base) – הפונקציה מקבלת כקלט מספר בייצוג בינארי **כמחרוזת** ואת בסיס היעד אליו יש לבצע המרה, מספר שלם כך ש- dest_base = [8,16] ומחזירה את המספר בייצוג בבסיס היעד (אוקטלי או הקסאדצימלי) **כמחרוזת**. לדוגמא: עבור הקלט ("110101",16) יוחזר הפלט '35' .

4. oct_or_hexa_to_binary(hexoct_num, orig_base) – הפונקציה מקבלת כקלט מספר **כמחרוזת** ואת בסיס המקור שלו, מספר שלם כך ש orig_base = [8,16] ומחזירה את המספר בייצוג בינארי **כמחרוזת**. לדוגמא: עבור הקלט (8, 24) הפונקציה תמיר את המספר מייצוג אוקטלי לייצוג בינארי ויוחזר הפלט "11000".

*הערה: אין להחזיר ספרות מיותרות שלא לצורך (אפסים מובילים), לדוגמא עבור המרת 29 מבסיס עשר לבסיס שתיים, יש להחזיר 11101 ולא 00011101.

<u>דוגמאות הרצה:</u>

```
binary_as_str = "11101"
print(binary_to_decimal(binary_as_str))
# *** output: 29
```

```
decimal_num = 27
print(decimal_to_binary(decimal_num))
# *** output: 11011
```

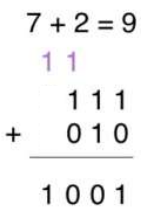
הנחיות כלליות לסעיפים 5-7:

- אין לבצע המרה מבסיס בינארי לבסיס אחר לצורך ביצוע פעולות חיבור וחיסור.
- יש להשתמש בפונקציה שמומשה בסעיף 1 להמרת בינארי לדצימאלי לפני החזרת הערך בפונקציה.

פעולה החיבור הבינארי:

חיבור של מספרים בינאריים מתבצע באופן הבא:

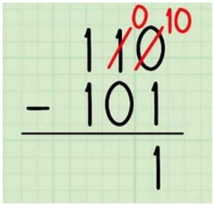
- ניקח כדוגמא את המספרים $(10)_2$ ו- $(111)_2$.
- נוסיף אפסים למספר הקצר יותר (משמאל, לאחר ה-MSB) ע"מ לבטא את שני המספרים במספר ספרות זהה, המספר החדש שיתקבל – $(010)_2$.
- נגדיר **שארית**, כשארית פעולת החיבור הבינארי בין זוג ספרות.
- נבצע חיבור ספרות מימין לשמאל, מה- LSB ועד ה-MSB של המספר כך ש:
  - 0 + 0 = 0, שארית 0
  - 0 + 1 = 1, שארית 0
  - 1 + 1 = 0, שארית 1
- פירוט הפעולות: 1+0 = 1 שארית 0, 1+1 = 0 שארית 1, 1+0+1(שארית) = 0 שארית 1, 1+0+0(שארית) = 1 שארית 0.
- להרחבה (פרינסטון)



פעולת החיסור הבינארי:

חיסור של מספרים בינאריים מתבצע באופן הבא:

- ניקח כדוגמא את המספרים $(101)_2$ ו- $(110)_2$.
- נגדיר **מושאל**, כמושאל לצורך החיסור הבינארי בין זוג ספרות (בדומה לחיסור מספרים עשרוניים).
- נבצע חיסור ספרות מימין לשמאל, מה- LSB ועד ה-MSB של המספר כך ש:
  - 0 מינוס 0 = 0
  - 1 מינוס 0 = 1
  - 0 מינוס 1 = 0, **נשאיל**
  - 1 - 1 = 0
- פירוט הפעולות: נשאיל 10 לצורך ביצוע החיסור, 10 מינוס 1 = 1, 0 מינוס 0 = 0, 1 מינוס 1 = 0
- ניתן להניח כי תוצאת החיסור בתרגילים אלו תמיד תהיה חיובית.



5. pad_zeros(binary_as_str, length) – הפונקציה תקבל כקלט מספר בינארי כמחרוזת ואורך כמספר שלם, תוסיף אפסים משמאל למספר הבינארי ותחזיר את המחרוזת החדשה שמתקבלת. לדוגמא: עבור הקלט (5,"1001") תוחזר המחרוזת "01001". הנ"ל תשמש כפונקציית עזר לסעיפים הבאים.

6. binary_addition(binary_as_str1, binary_as_str2) – הפונקציה מקבלת כקלט שני מספרים בייצוג בינארי **כמחרוזות** ומחזירה כפלט את תוצאות החיבור של שני המספרים **כמחרוזת** בייצוג בינארי *אין להמיר את המספרים לבסיס אחר לצורך ביצוע החיבור.

7. binary_substraction(binary_as_str1, binary_as_str2) - הפונקציה מקבלת כקלט שני מספרים בייצוג בינארי **כמחרוזות** ומחזירה כפלט את תוצאות החיסור של שני המספרים **כמחרוזת** בייצוג בינארי *אין להמיר את המספרים לבסיס אחר לצורך ביצוע החיסור.

8. **שאלת בונוס:**
binary_multiplication(binary_as_str1, binary_as_str2) - הפונקציה מקבלת כקלט שני מספרים בייצוג בינארי **כמחרוזות** ומחזירה כפלט את תוצאות הכפל של שני המספרים בייצוג בינארי **כמחרוזת.**

9. memory_map(binary_as_str) – הפונקציה מקבלת כקלט מספר בייצוג בינארי **כמחרוזת**, מוסיפה למספר השלמה ל- Byte של אפסים משמאל על מנת לייצג את המספר בצורתו האמיתית בזכרון,
לדוגמא:
עבור הקלט - "1000101010"
יתקבל הפלט –"00000001000101010"
ומחזירה כפלט את המספר שיתקבל **כמחרוזת.**

10. twos_compliment(num_dec) - הפונקציה מקבלת כקלט מספר בייצוג עשרוני ומחזירה כפלט את המשלים ל-2 של המספר בייצוג בינארי **כמחרוזת**, בבתים שלמים, כאשר הביט השמאלי ביותר מייצג האם המספר שלילי או לא. **הערה**: מספר חיובי שבו הביט השמאלי ביותר (MSB) דלוק יושלם לעד Byte של אפסים (Byte = 8 bits).
דוגמא:
עבור הקלט 127, (המספר 127 בייצוג עשרוני כמספר חיובי) המספר יומר לייצוג בינארי – "1111111" ואליו יתווסף מינימום אפסים משמאל כהשלמה לByte שלם (כך שאורך המספר המוחזר יהיה בכפולות של 8), ע"מ לייצג את המספר בצורתו האמיתית בזיכרון כמספר חיובי (MSB=0), הפלט הצפוי יהיה – "01111111". *במידה ולא נוסיף 0 משמאל, המספר עשוי לייצג את המספר 1-.
דוגמאות נוספות:
- **"1000000" (64)** יוחזר כ- "0100000".
- **"11111111"(255)** יוחזר כ- "0000000011111111"
- "**10000000"(-128)** יוחזר כפי שהוא.

מילונים:

בסעיפים הבאים יהיה עליכם לממש תכנית בשפת פייתון שתשתמש במילון לביצוע המרות בין הבסיסים השונים.

11. create_dict(destination_base) – הפונקציה מקבלת כקלט את בסיס היעד (8 או 16) כמספר שלם ומחזירה מילון כך שהמפתחות במילון הם הייצוג הבינארי של המספר והערכים הם הייצוג בבסיס היעד שהתקבל, על המפתחות להכיל 4 ספרות בייצוג בינארי כמחרוזת.
לדוגמא:
עבור הקלט – 8
יתקבל המילון – {"0000":0,"0001":1,"0010":2,"0011":3,"0100":4,"0101":5,"0110":6,"0111":7}
*אין לכתוב את המילון בצורה ידנית, ניתן להשתמש בפונק' המקבלת מספר בינארי ומוסיפה 1 בינארית כפי שהוראתה בהרצאה.

12. dict_reversal(dictionary) – הפונקציה מקבלת כקלט מילון, מבצעת היפוך, כך שהערכים במילון החדש יהיו המפתחות והמפתחות יהיו הערכים ומחזירה את המילון.
לדוגמא:
עבור הקלט – {"a":1, "b":2, "c":3}
יתקבל הפלט – {1:"a",2:"b",3:"c"}

13. base_conversion_w_tests(number_as_str, orig_base, dest_base) – הפונקציה מקבלת כקלט מספר בייצוג כלשהוא **כמחרוזת**, את בסיס המקור של המספר ואת בסיס היעד להמרה בהתאמה, הפונקציה תבצע בדיקה ותחזיר המרה של המספר מבסיס המקור לבסיס היעד כך ש:
- בסיס דצימאלי יוחזר כ-int
- בסיסים 2,8,16 יוחזרו כ-string
*חובה להשתמש בפונקציות שכתבתם.

בדיקת קלט ניתן לבצע בPyCharm- או בפלטפורמה האינטרנטית, הגשה יש לבצע בפלטפורמה האינטרנטית.

בהצלחה! אוריאל.