**דף עזר לתלמיד - שפת Python גרסה 3**

**מבנים בשפה**

|  |  |
| --- | --- |
| **משפטי תנאי** | **if** <תנאי>:  הוראות לביצוע |
| **if** <תנאי>:  הוראות לביצוע  **else**:  הוראות לביצוע |
| **if** <תנאי>:  הוראות לביצוע  **elif** <תנאי>:  הוראות לביצוע  **else**:  הוראות לביצוע |
| **לולאות** | **while** <תנאי>:  הוראות לביצוע |
| **for** index **in** range(start, end, step):  הוראות לביצוע |
| **פעולות** | הגדרת פעולה בשם func שאינה מקבלת פרמטרים:  **def func():**  הוראות לביצוע  הגדרת פעולה בשם func שמקבלת פרמטרים:  **def func(**param1, param2, …**):** הוראות לביצוע |

**אופרטורים**

|  |  |
| --- | --- |
| **אופרטורים חשבוניים** | - + \* / \*\* // % |
| **אופרטורים חשבוניים מקוצרים (כולל השמה)** | -= += \*= /= \*\*= //= %=  לדוגמה:  x += 2 שקול כמו x = x+2 |
| **אופרטורים לוגיים** | > < >= <= == != |

**הוראות קלט - פלט**

|  |  |
| --- | --- |
| **קלט** | פעולה הקולטת **מחרוזת** מהמשתמש:  string = **input**(‘Enter your input:’)  ניתן להסב (casting) משתנים מטיפוס אחד לאחר.  לדוגמה: פעולת הסבה של מחרוזת הנקלטת לטיפוס מספר שלם (integer)  number = **int**(**input**(‘Enter your integer number:’))  ניתן לקלוט ערך שלם מהמשתמש ולשלב בפעולה חישוב מתמטית, כמו צבירה לערך קיים. לדוגמה:  total = total+ **int**(**input**(‘Enter your integer number:’)) |
| **פלט** | פעולת הדפסה למסך:  **print** (‘your output!’)  **print** (<שם משתנה>) |
| **המרה** | var = **int**(var)  var = **float**(var)  var = **str**(var) |

**פעולות שימושיות מתוך ספריות**

|  |  |
| --- | --- |
| **ייבוא ספריה מתמטית** | **import** math  במידה והפעולה היא פעולת ספרייה (כגון sqrt) |
| **פעולות מתמטיות** | n = abs(number)  n = max(arg1, arg2, arg3, ...)  n = min(arg1, arg2, arg3, ...)  n = math.sqrt(number) |
| **יבוא ספריית random** | **import** random |
| **פעולות ליצירת מספר אקראי** | הגרלת מספר אקראי לא שלם בין 0 ל - 1:  num = random.**random**()  הגרלת מספר אקראי (num) בטווח start <= **num** < end:  num = random.**randrange**(start, end)  הגרלת מספר אקראי (num) בטווח start <= **num** <= end (כולל 2 הקצוות):  num = random.**randint**(start, end) |

**מבנה נתונים - מחרוזות**

|  |  |
| --- | --- |
| **מחרוזות**  מחרוזת הינה סדרה של תווים (אותיות, סימנים, מספרים)  מוקפים מצדם בגרש בודד או בגרשיים (מרכאות). | |
| אתחול מחרוזת ריקה | msg = **str**()  msg = ’ ’ |
| אתחול מחרוזת עם ערך התחלתי | msg = ‘Hi’  note = ‘a123$’ |
| החזרת איבר במחרוזת במיקום (index) מסוים (החל מ-0) | ch = msg[index] |
|
| החזרת אורך מחרוזת | length = **len**(msg) |
| שרשור מחרוזות (חיבור שלהן) | new\_msg = msg + note >>>Hia123$ |
| כפל מחרוזות, במספר שלם חיובי num | msg \* num |
| new\_msg = msg \* 3 >>>HiHiHi |
| החזרת מחרוזת חדשה על פי חיתוך מחרוזת קיימת  start - מאיפה  end - עד איפה, לא כולל  step - בקפיצות | msg[start: end: step]  דוגמה:  msg = ‘Hello’  new\_msg = msg[ : : -1]  >>>‘olleH’  new\_msg = msg[1:3: ]  >>>‘el’ |
|
| בדיקת האם תת-מחרוזת שבמשתנה ch קיימת במחרוזת msg | **if** ch **in** msg: |
| החזרת מציין ראשון של המחרוזת s במחרוזת msg.  **אם לא קיים מחזיר 1-** | place = msg.**find**(‘s’) |
| החזרת מציין ראשון של המחרוזת s במחרוזת msg.  **אם לא קיים מחזיר הודעת שגיאה בזמן ריצה!** | place = msg.**index**(‘s’) |
| החזרת ערך אסקי של התו ch | ch = ‘a’  ord(ch)  >>>97 |
| החזרת התו שהערך האסקי שלו הוא val | val = 97  chr(val)  >>>‘a’ |
| החזרת מחרוזת חדשה עם אותיות גדולות / קטנות בהתאמה | msg.**upper**()  msg.**lower**() |
| בדיקה האם המחרוזת msg במיקום start מתחילה / מסתיימת במחרוזת s | msg.**startswith**(s, start, end)  msg.**endswith**(s, start, end) |
| החלפת המחרוזת old\_s במחרוזת new\_s במחרוזת msg מספר פעמים count | msg.**replace**(old\_s, new\_s, count) |
| ספירת מספר הפעמים שתת-המחרוזת שבמשתנה ch מופיעה במחרוזת msg | times = msg.**count**(ch) |

**מבנה נתונים - רשימות**

|  |  |
| --- | --- |
| **רשימות**  רשימה היא מבנה נתונים המורכב מאיבר אחד או יותר, המופרדים ביניהם בפסיקים. | |
| יצירת רשימה ריקה | my\_list = **list**()  my\_list = [ ] |
| קבלת ערך במציין (=במיקום) index | my\_list [index] |
| מספר איברים ברשימה | **len**(my\_list) |
|
| החזרת כמות המופעים של value ברשימה | my\_list.**count**(value) |
| החזרת המציין **הראשון** ברשימה בו מופיע value  **אם לא קיים מחזיר הודעת שגיאה בזמן ריצה!** | my\_list.**index**(value) |
| בדיקה האם item קיים ברשימה | **if** item **in** my\_list: |
| החזרת רשימה ע"פ חיתוך רשימה קיימת  start - מאיפה  end - עד איפה, לא כולל  step - בקפיצות | My\_list [ start : end : step ] |
| הוספת איבר item לסוף רשימה my\_list | my\_list.**append**(item) |
| הוספת איבר item במציין/מיקום pos של הרשימה | my\_list.**insert**(pos, item) |
| מחיקת ערך מהרשימה, לפי ערך (מוחק את המופע הראשון של value). אם לא נמצא ברשימה, מחזיר שגיאה. | my\_list.**remove**(value) |
| מחיקת ערך מהרשימה, לפי מיקום (מוחק את האיבר במקום ה-index), והחזרת ערכו. אם אין מיקום כזה, מחזיר שגיאה.  מוחק את האיבר האחרון ברשימה. | x = my\_list.**pop**(index)  x = my\_list.**pop**() |
| מיון רשימה בסדר עולה (מהקטן לגדול)  מיון רשימה בסדר יורד (מהגדול לקטן) | my\_list.**sort**()  my\_list.**sort**(reverse=True) |
| מיון רשימה ע"פ מפתח  לדוגמה:  המפתח key=str.lower ממיין קודם את האותיות הקטנות, ואחריהן את האותיות הגדולות. | lst = ['This', 'is', 'a', 'test', 'List']  lst.sort(key=str.lower)  print(lst)  ['a', 'is', 'List', 'test', 'This'] |
| מיון רשימה, ללא שינוי הרשימה המקורית,  מחזיר רשימה ממויינת ע"פ הבקשה. | lst = ['This', 'is', 'a', 'test', 'List']  lst2 = sorted(lst, key=str.lower)  print(lst2)  print(lst)  ['a', 'is', 'List', 'test', 'This']  ['This', 'is', 'a', 'test', 'List'] |
| הפיכת סדר הפריטים ברשימה | my\_list.**reverse**() |
| החזרת מחרוזת מתוך איברי הרשימה  באמצעות הוספת delimiter בין האיברים | my\_list = [‘I’, ‘love’, ‘Python’]  delimiter = ‘\_’  s = delimiter.**join**(my\_list)  **print** (s)  >>> ‘I\_love\_Python’ |
| החזרת רשימה חדשה באמצעות פיצול מחרוזת s  לאיברים לפי delimiter | s = ‘I\_love\_Python’  delimiter = ‘\_’  my\_list = s.**split**(delimiter)  **print** (my\_list)  >>> [‘I’, ‘love’, ‘Python’] |
| החזרת רשימה חדשה באמצעות פיצול מחרוזת s  לאיברים לפי רווח | my\_list = s.**split**() |

**מבנה נתונים - תור**

|  |  |
| --- | --- |
| **תור**  תור הוא מבנה נתונים המורכב מאיברים סדורים.  לתור יש סדר הכנסה (מכניסים לסוף התור) וסדר הוצאה (מוציאים מראש התור).  סדר זה נקרא FIFO = First In First Out כלומר: הראשון שנכנס הוא הראשון שיוצא. | |
| קריאה לספריה של תור | **import** queue |
| יצירת-תור-ריק | q = queue.Queue() |
| הכנסה-לתור | q.**put**(ערך\_שנרצה\_להכניס\_לתור) |
| הוצאה-מהתור | q.**get**() |
| בדיקה-האם-התור-ריק? | q.**empty**() |
| בדיקת-ערך-בראש-התור | לא ממומש בספריה |

**מודל Turtle**

|  |  |  |
| --- | --- | --- |
| **תחילת עבודה** | קריאה למחלקה הגרפית | **import** turtle |
| יצירת מסך עבודה עבור צב | wn = turtle.Screen() |
| יצירת צב בשם player | player= turtle.Turtle() |
| **פעולות תנועה** | התקדמות קדימה steps צעדים | player.**forward**(steps)  player.**fd**(steps) |
| התקדמות אחורה steps צעדים | player.**backward**(steps)  player.**back**(steps) |
| קפיצה למיקום על המסך | player.**goto**(x, y) |
| שינוי מיקום של הצב - x  שינוי מיקום של הצב - y | player.**setx**(x)  player.**sety**(y) |
| קבלת ערכי מיקום הצב למשתנים | x = player.**pos**()[0]  y = player.**pos**()[1]  או:  x = player.**xcor**()  y = player.**ycor**() |
| ציור עיגול שלם ברדיוס r | player.**circle**(r) |
| ציור עיגול חלקי (קשת a מעלות) ברדיוס r | player.**circle**(r,a) |
| חתימת צב במיקום הנוכחי | player.**stamp**() |
| **פעולות סיבוב** | סיבוב צב שמאלה ב alpha מעלות (נגד כיוון השעון) יחסית לזוית נוכחית | player.**left**(alpha)  player.**lt**(alpha) |
| סיבוב צב ימינה ב alpha מעלות (עם כיוון השעון) יחסית לזוית נוכחית | player.**right**(alpha)  player.**rt**(alpha) |
| סיבוב צב לזוית alpha. בזוית מוחלטת יחסית לזוית ה-0 מעלות.  (זוית חיובית היא נגד כיוון השעון) | player.**setheading**(alpha) |
| מחזיר את זוית הסיבוב הנוכחית | player.**heading**() |

|  |  |  |
| --- | --- | --- |
| **פעולות מאפייני צב** | עט מטה – הצב יצייר מסלול  כשיבצע הוראות | player.**pendown**()  player.**down**()  player.**pd**() |
| עט מעלה – הצב **לא** יצייר מסלול  כשיבצע הוראות | player.**penup**()  player.**up**()  player.**pu**() |
| שינוי עובי העט שאיתו מציירים | player.**pensize**(num) |
| שינוי צבע העט שאיתו מציירים | player.**pencolor**('blue') |
| שינוי צבע הצב | player.**color**('red') |
| שינוי צורת הצב  הצורות הקיימות בשפה: | player.**shape**("turtle")  **'turtle', 'arrow', 'square', 'circle', 'triangle', 'classic', 'blank'** |
| הסתר צב | player.**hideturtle**() |
| הצג צב | player.**showturtle**() |
| האם הצב מוצג? (מחזיר True או False) | player.**isvisible**() |
| שנה גודל צב לערך size (הערה: כל גודל צב מיוצג כ 20 פיקסלים בציור) | player.**turtlesize**(size) |
| **האזנה ללחיצת מקלדת** | לחיצה על מקש מקלדת. יש להגדיר פעולה שקוראים לה בלחיצה. דוגמה: בלחיצה על מקש חץ שמאלה, נקרא לפעולה kL:  wn.**onkeypress**(kL,'Left') | wn.**onkeypress**(func,key) |
| עזיבה של מקש מקלדת. יש להגדיר פעולה שקוראים לה בעזיבה. דוגמה: בעזיבה של מקש חץ שמאלה, נקרא לפעולה rL:  wn.**onkeyrelease**(rL,'Left') | wn.**onkeyrelease**(func,key) |
| **האזנה ללחיצת עכבר** | בלחיצה על העכבר, קוראים לפעולה function\_name, שנגדיר מראש. | wn.**onclick**(function\_name) |
| **פקודות סיום** | הרצת פעולה ברקע.  יציאה בלחיצת עכבר. | turtle.**mainloop**()  wn.**exitonclick**() |