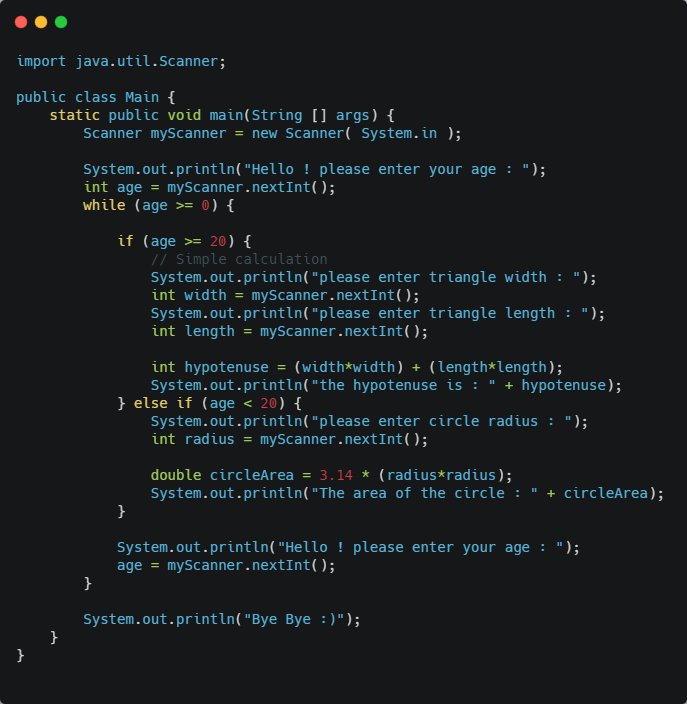
**פונקציות**

שימו לב לקטע הקוד הבא:

[](http://shorturl.at/FJKPU)

מה הבעיה בקוד הנ"ל? כמה זמן לקח לכם להבין מהי מטרת הקוד?

**הקוד ארוך** ולא מובן!

עד כה, כתבנו את הקוד בתכנית שלנו באותו המקום, שורה אחר שורה, בתוך פונקציית הmain.

הבעיה מתחילה כאשר גודל הפרויקט גדול, עד כדי אלפי שורות קוד.

במצב כזה, כאשר כל הקוד יהיה כתוב שורה אחר שורה בתוך הmain, ניתקל בקוד מסורבל, ארוך, מסובך שמקשה גם על תחזוקה שלו לטווח ארוך.

מה הפתרון ? **פונקציות**!

**פונקציה:** קטע לוגי סגור המבצע פעולה מוגדרת. פונקציה יכולה גם **לקבל** פרמטרים, לבצע עליהם פעולה מסוימת, ו**להחזיר** ערך כלשהו.

נגדיר פונקציה בצורה הבאה:

{תוכן הפונקציה} (פרמטרים) [שם הפונקציה] [ערך החזר]

**פרמטרים**

הפונקציה יכולה לקבל משתנים בתור פרמטרים. היא יכולה להשתמש בהם ולבצע באמצעותם פעולות מסוימות – בדומה למה שיכולנו לבצע עד כה.

פונקציה יכולה לקבל פרמטר אחד, שניים, שלושה... כמה שנרצה. הפרמטרים יכולים להיות מטיפוסים שונים. (אחד int, השני char, השלישי double וכו').

* ניתן להשתמש בפונקציה שלא מקבלת פרמטרים כלל

**שם פונקציה**

על כל פונקציה להיקרא בשם ייחודי, באמצעותו נקרא לפונקציה ספציפית.  
ניתן לבחור בכל עבור פונקציות– תמיד נבחר שם בעל משמעות **המשקפת את תפקידה** של הפונקציה.

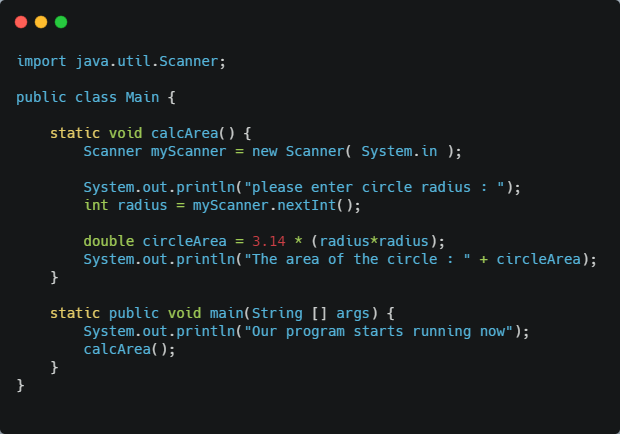
**תוכן הפונקציה**

בתוכן הפונקציה נכתוב את הפעולות אותן נרצה לבצע עבור כל קריאה לפונקציה.  
ברגע שנקרא לפונקציה שלנו, הקומפיילר אוטומטית ייכנס לתוכן הפונקציה ויבצע את הפעולות שהיא כוללת. לאחר שריצת הפונקציה תסתיים, הקומפיילר יחזור למקום הקודם ממנו הגיע לפונקציה.

**ערך החזר**

ניתן להחזיר ערך מסוים בסיום ריצת הפונקציה. בסוף ביצוע הפעולות הנכללות בפונקציה, ניתן להחזיר תוצאה מסוימת שככל הנראה משקפת את ביצוע הפעולות.

**דוגמה לשימוש בפונקציה:**



בדוגמה זו הוגדרה פונקציה בשם "calcArea".

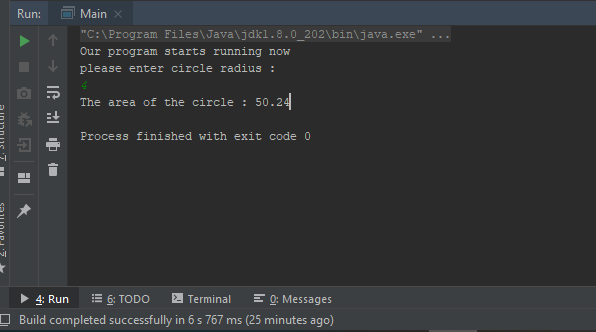
פונקציה זו לא מקבלת פרמטרים ולא מחזירה ערך החזר.  
הפונקציה קולטת ערך של רדיוס ולפי נוסחה מחשבת את שטח המעגל.

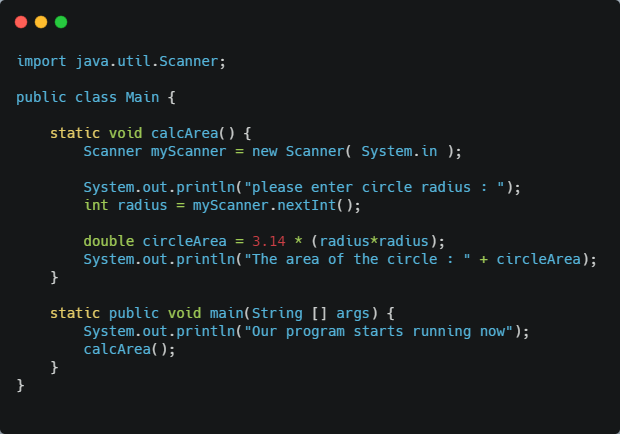
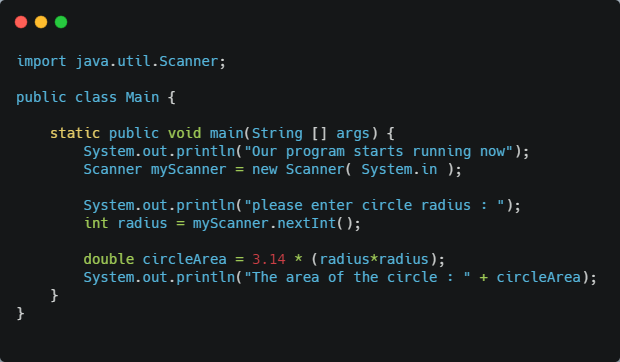
**ריצת הקוד תמיד תתחיל מפונקציה הmain!**

אז למה שנשתמש בפונקציות?

1. חלוקה לוגית - פישוט הבעיה , כל קטע לוגי נכתוב בפונקציה שונה ( למשל – מיון ניתן לכתוב בשגרה שונה כי זהו קטע לוגי סגור).
2. קריאות התוכנית (בלי פונקציית main ראשית מסורבל וקשה להבנה).
3. שימוש חוזר בקוד – ניתן להשתמש בפונקציה כמה פעמים במהלך הקוד. במקרה כזה צריך להוסיף שורת קוד אחת שקוראת לפונקציה וגורמת לה לפעול. במידה ולא היינו משתמשים בפונקציה, היה עלינו לרשום את אותו קטע הקוד בשנית אם היינו רוצים להשתמש בו במקומות שונים בקוד.
4. מאפשר עבודת צוות – כל חבר צוות יעבוד על פונקציה שונה.

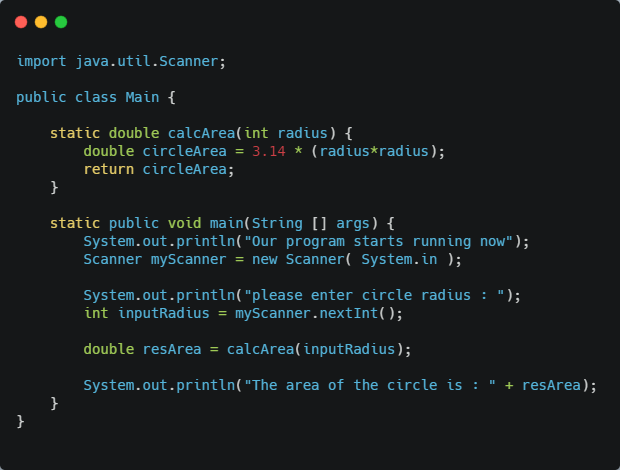
במידה ונריץ את הפרויקט, נראה שקודם כל תודפס ההודעה הראשונה הכתובה בmain, ורק לאחר מכן רצות תבוצענה הפעולות הנמצאות בפונקציה " calcArea".

לאחר ריצת הפונקציה חלון הconsole ייראה כך:

[](https://carbon.now.sh/?bg=rgba(178%2C192%2C204%2C0.98)&t=seti&wt=none&l=text%2Fx-csrc&ds=true&dsyoff=20px&dsblur=68px&wc=true&wa=true&pv=0px&ph=0px&ln=false&fl=1&fm=Hack&fs=14px&lh=133%25&si=false&es=1x&wm=false&code=import%2520java.util.Scanner%253B%250A%250Apublic%2520class%2520Main%2520%257B%250A%250A%2520%2520%2520%2520static%2520public%2520void%2520main(String%2520%255B%255D%2520args)%2520%257B%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522Our%2520program%2520starts%2520running%2520now%2522)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520Scanner%2520myScanner%2520%253D%2520new%2520Scanner(%2520System.in%2520)%253B%250A%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522please%2520enter%2520circle%2520radius%2520%253A%2520%2522)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520int%2520radius%2520%253D%2520myScanner.nextInt()%253B%250A%250A%2520%2520%2520%2520%2520%2520%2520%2520double%2520circleArea%2520%253D%25203.14%2520*%2520(radius*radius)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522The%2520area%2520of%2520the%2520circle%2520%253A%2520%2522%2520%252B%2520circleArea)%253B%250A%2520%2520%2520%2520%257D%250A%257D%250A)דוגמא לפיצול של קוד לפונקציה:

**קבלת פרמטרים**  
נגדיר את שם וטיפוס כל פרמטר, על מנת שנדע איך להשתמש בו בתוך הפונקציה עצמה.

**החזרת ערך**

נציין את **סוג המשתנה** בלבד ועל מנת לבצע החזר של אותו משתנה, נשתמש במילה השמורה return.  
[](https://carbon.now.sh/?bg=rgba(178%2C192%2C204%2C0.98)&t=seti&wt=none&l=text%2Fx-csrc&ds=true&dsyoff=20px&dsblur=68px&wc=true&wa=true&pv=0px&ph=0px&ln=false&fl=1&fm=Hack&fs=14px&lh=133%25&si=false&es=1x&wm=false&code=import%2520java.util.Scanner%253B%250A%250Apublic%2520class%2520Main%2520%257B%250A%250A%2520%2520%2520%2520static%2520double%2520calcArea(int%2520radius)%2520%257B%250A%2520%2520%2520%2520%2520%2520%2520%2520double%2520circleArea%2520%253D%25203.14%2520*%2520(radius*radius)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520return%2520circleArea%253B%250A%2520%2520%2520%2520%257D%250A%250A%2520%2520%2520%2520static%2520public%2520void%2520main(String%2520%255B%255D%2520args)%2520%257B%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522Our%2520program%2520starts%2520running%2520now%2522)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520Scanner%2520myScanner%2520%253D%2520new%2520Scanner(%2520System.in%2520)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522please%2520enter%2520circle%2520radius%2520%253A%2520%2522)%253B%250A%2520%2520%2520%2520%2520%2520%2520%2520int%2520inputRadius%2520%253D%2520myScanner.nextInt()%253B%250A%250A%2520%2520%2520%2520%2520%2520%2520%2520double%2520resArea%2520%253D%2520calcArea(inputRadius)%253B%250A%250A%2520%2520%2520%2520%2520%2520%2520%2520System.out.println(%2522The%2520area%2520of%2520the%2520circle%2520is%2520%253A%2520%2522%2520%252B%2520resArea)%253B%250A%2520%2520%2520%2520%257D%250A%257D%250A)

הפונקציה הבאה, מקבלת את המשתנה "inputRadius" הנקלט מהמשתמש דרך פונקציית הmain.

לאחר מכן מתבצע **זימון** (קריאה) לפונקציה, אליה מועבר בתור פרמטר המשתנה "inputRadius".

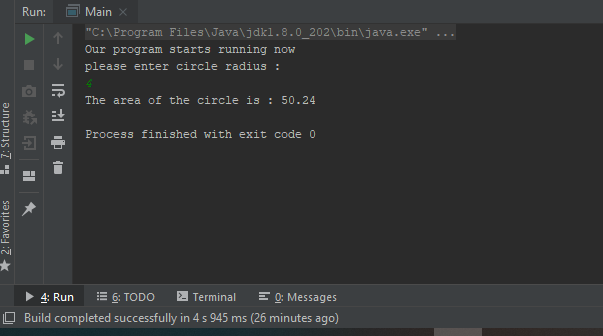
בתוך הפונקציה "calcArea" משתנה זה יקרא "radius".

לבסוף, ערך המשתנה “circeArea” מוחזר באמצעות פקודת return בתוך הפונקציה.



לאחר ריצת שורת קוד זו, המשתנה "resArea" יכיל את הערך שהפונקציה calcArea מחזירה.

פלט הערך המוחזר יהיה:



**טווח הכרה ואורח חיים בפונקציות**

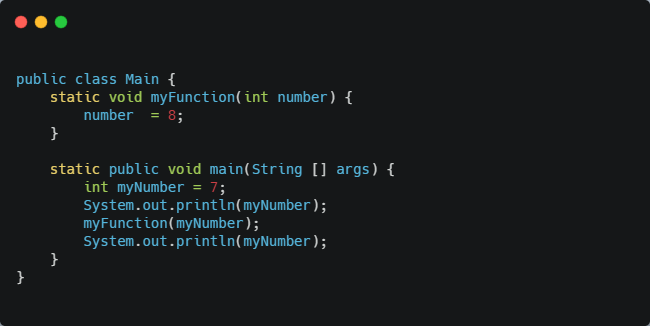
**טווח הכרה של משתנה -** מאילו מקומות בקוד ניתן לגשת לערכו של משתנה מסוים ולשנות את ערכו.

**אורח חיים של משתנה -** באיזה שלב בריצת התוכנית המשתנה נוצר במערכת ומוקצה עבורו מקום בזיכרון, ובאיזה שלב בריצת התוכנית הוא נמחק מהזיכרון לצמיתות.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

בעת העברת פרמטרים לפונקציה, במידה ואלו משתנים פרימיטיביים רגילים (למשל int או double) ולא משתנים שהם אובייקטים (למשל String) - משתנים אלו מועברים **byValue**.

כלומר, במידה והפונקציה תשנה את ערכם של המשתנים מסוג זה במהלך אחת מפקודותיה, ערכם המקורי (שנשלח לפונקציה מבחוץ) לא ישנה.

לדוגמה:

  
(התוכנית תדפיס 7 לפני ריצת הפונקציה , ו7 אחרי הריצה של הפונקציה – נסו בעצמכם!)

בנוסף, על המשתנים המוגדרים בתוך בלוק הפונקציה, ניתן לבצע קריאה ועריכה אך ורק **מתוך הפונקציה** ולא משום מקום אחר.

יתרה מכך, בסיום ריצת הפונקציה, כל המשתנים שהוגדרו בבלוק ימחקו מהזיכרון.



המשתנה myVariable שמוגדר בתוך הפונקציה, מקבל את הערך 8.  
אולם מחוץ לפונקציה (למשל מהmain) לא ניתן כלל לגשת למשתנה myVariable, ולאחר סיום ריצת הפונקציה, המשתנה יימחק מהזיכרון.

במידה ונעביר לפונקציה משתנים שהם אובייקטים (Reference variables) למשל String

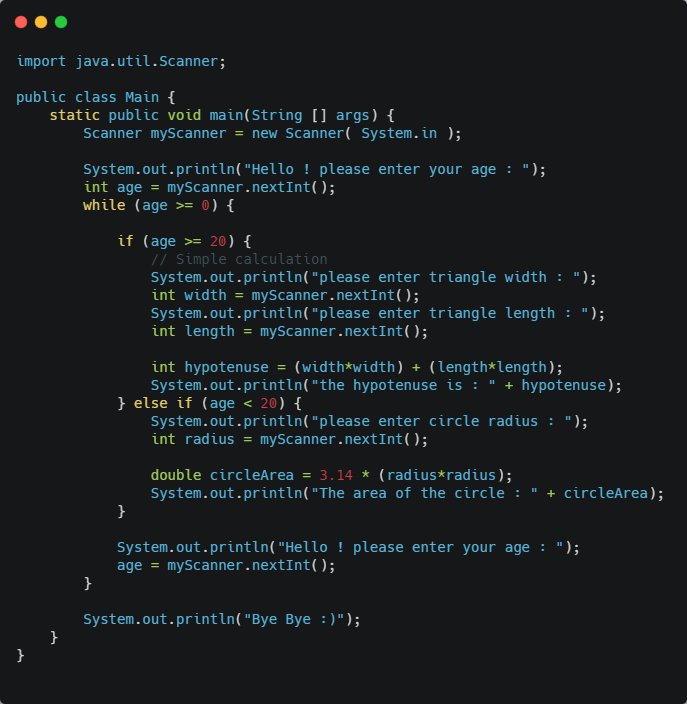
כאשר הפונקציה תשנה פרמטר זה, הוא **אכן גם ישתנה מחוץ לפונקציה.**הדבר נכון אך רק למשתנים שהם אובייקטים!

**קווים מנחים לחלוקה נכונה לפונקציות**

כאמור, לחלוקת הקוד שלנו לפונקציות יש יתרון משמעותי.  
חשוב להקפיד על מספר כללים שיסייעו לנו לכתוב את הקוד שלנו בצורה טובה יותר.

* שמות משמעותיים לפונקציות  
  יש להשתמש בשם משמעותי וקצר המתאר את מטרת הפונקציה.  
  הדבר יהפוך את הקוד שלנו לפשוט וקריא יותר.
* **קטעים לוגיים חוזרים**  
  במידה ויש קטעי קוד החוזרים על עצמם שוב ושוב לאורך התוכנית, מומלץ לחלץ אותם לפונקציה נפרדת שמבצעת את אותה הפעולה. במקום להשתמש באותו הקוד בכל מקום בו נרצה להשתמש באותה הפעולה, נוכל כל פעם לקרוא לפונקציה, באמצעות שורת קוד אחת בלבד.  
  בצורה זו גם נחסוך בשורות קוד וגם במידה ונרצה לבצע תיקון, נתקן רק פעם אחת בפונקציה במקום בכל מיני מקומות בקוד.
* **להימנע מקטעי קוד ארוכים**  
  במידה ויש קטע קוד ארוך מידי, כנראה שניתן לפצל גם את הפונקציה הזו לפונקציה נוספת.

**תרגול**

1. **[](http://shorturl.at/FJKPU)חלוקה לפונקציות**

מה הקטע קוד הבא עושה? קשה להבין?  
נסו לפתור זאת שוב בעצמכם, לאחר שתבצעו **חלוקה לפונקציות.**

יש לבצע חלוקה **הגיונית** לפונקציות. ניתן לקרוא לפונקציה מסוימת **בתוך** פונקציה אחרת.

לסיום, יש להריץ את הפרויקט ולוודא שהוא עובד כנדרש.

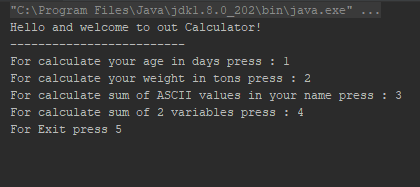
1. **מחשבון**

עלינו לכתוב תוכנית המדמה מחשבון!

המחשבון שלנו בכל פעם מציג תפריט למשתמש (כלומר את הפעולות שניתן לבצע במחשבון).  
המשתמש בוחר איזו פעולה הוא מעוניין לבצע, מכניס פרמטרים והתוכנית שלנו תחשב את התוצאה ותציג אותה למשתמש.

* התוכנית תדפיס בכל פעם את התפריט שוב ושוב מחדש למשתמש, וכאשר המשתמש יבקש לצאת, התוכנית תדפיס הודעה מתאימה והריצה תסתיים.
* **חשוב:** עליכם לפצל את התוכנית לפונקציות נפרדות לפי קטעים לוגיים ספציפיים בצורה חכמה!

התפריט שיוצג למשתמש בכל פעם (עד אשר אשר המשתמש יבחר לצאת) ייראה כך:



**להלן פירוט על הפעולות הרלוונטיות:**

1. חישוב גיל המשתמש בימים. עלינו לכתוב פעולה המקבלת את גילו בשנים ומחשבת את גילו בימים. (בהנחה שיש 365 ימים בשנה).  
   הפעולה הזו לא מחזירה ערך החזר אלא מדפיסה את גילו של המשתמש בלבד.
2. חישוב משקל המשתמש בטונות. עלינו לכתוב פעולה המקבלת את משקל המשתמש בקילוגרמים. הפעולה תחזיר בערך החזר את משקלו בטונות **וגם** תדפיס את משקלו בטונות.
3. פעולה המדפיסה מספר אשר מהווה את סכום כל ערכי הASCII של שם המשתמש.  
   פעולה זו לא תקבל פרמטרים. כחלק מהחלק הלוגי של הפונקציה היא תקלוט מהמשתמש את שמו.  
   לאחר שהמשתמש הכניס את שמו המלא, הפונקציה מחשבת את סכום ערכי הASCII של שמו ומדפיסה בחזרה למשתמש (הפונקציה לא מחזירה ערך החזר).
4. פונקציה המקבלת שתי פרמטרים מסוג double, ומחזירה את הסכום שלהם.
5. הפונקציה לא מקבלת ולא מחזירה כלום. היא רק מדפיסה “Bye Bye” למשתמש.

**חשוב!**במידה ואתם רואים לנכון ליצור **עוד פונקציות** עזר שיעזרו לכם בפתרון התרגיל, הרי זה משובך – כל עוד שומרים על כללי החלוקה לפונקציות!

**בהצלחה!**