מקרא

מילת מפתח / כפתור	תיאור
	לחיצה על כפתור ה- **run** תגרום לקוד שלכם מימין לרוץ. תוכלו לראות את התוצאה על ידי התבוננות בסצנה משמאל.
	כפתור האיפוס (reset) ימחק את כל מה שכתבתם בקוד בצד הימני ויאפס את הקוד למצב שבו הוא היה בתחילת השלב.
	הסרגל הוא כלי שעוזר לכם למדוד את המרחק בין אובייקטים שונים במשחק, למשל, את המרחק בין הקוף והבננה.הסרגל יכול גם לעזור לכם למדוד את הזוויות בהן על הקוף או הצב לפנות כדי לעמוד מול אובייקט אחר שעל המסך, למשל מול הבננה. כדי להשתמש בסרגל, יש ללחוץ עליו פעם אחת, ולאחר מכן להשתמש בעכבר כדי להזיז את הסרגל עד לנקודה שתרצו להתחיל למדוד ממנה. יש ללחוץ על העכבר שוב, ולאחר מכן לגרור אותו לנקודה הסופית. מספר יופיע בנקודה הסופית ויציין את המרחק. השתמשו במספר זה עם הפקודה "step". מספר נוסף יופיע ליד נקודת ההתחלה, והוא יציין את הזווית בין האובייקט הראשון לשני. השתמשו בו עם הפקודה "turn".
	לאחר כל שלב, תקבלו דירוג כוכבים לפתרון שלכם. הכוכבים מחולקים כך: • הכוכב הראשון ניתן אם השגתם את כל הבננות • הכוכב השני ניתן אם השתמשתם בידע החדש שלמדתם בפרק זה של קודמאנקי • הכוכב השלישי ניתן אם הקוד שלכם הוא קצר ולעניין אם לא קיבלתם שלושה כוכבים, רמז יופיע במסך הניצחון כדי לעזור לכם לשפר את הקוד שלכם ולקבל שלושה כוכבים.
	כדי לגרום לקוף לצעוד למרחק מסוים, ז"א לבצע "step", אנחנו צריכים לכתוב "step X" ולהשתמש במספר הצעדים שאנחנו רוצים שהוא יעשה, לדוגמה, "step 10". לחיצה על כפתור ה- **step** תוסיף את המילה "step" בקוד שלכם.
	כדי לגרום לקוף להסתובב אנחנו צריכים להשתמש בפונקציה "turn" אשר צריכה להיות מלווה בכיוון (left / right) או במעלות (45, 90, 180). דוגמאות: "turn 90", "turn right" לחיצה על כפתור ה- **turn** תוסיף את המילה "turn" בקוד שלכם.

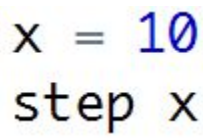
"Left" ו-"Right" משמשים אחרי הפקודה "turn" כדי לגרום לקוף לפנות לכיוון הרצוי.
לחיצה על הכפתורים **left** או **right** תוסיף בהתאמה את המילים "left" או "right" בקוד שלכם.

"TurnTo" היא דרך נוספת לפנות. במקום להשתמש בכיוון או מעלות, אנחנו מבקשים מהקוף לפנות אל אובייקט מסוים, לדוגמה, הפקודה "turnTo banana" תגרום לקוף להסתובב אל הבננה .
לחיצה על כפתור ה- **turnTo** תוסיף את המילה "turnTo" בקוד שלכם.

לולאה פשוטה היא רצף של הוראות החוזר על עצמו למשך מספר מוגדר של פעמים.
לדוגמה:

```
3.times ->
    step 5
    turn left
```

בדוגמה זו, קוף יחזור שלוש פעמים על "step 5, turn left".
ההוראות שאנחנו כותבים בלולאה צריכות להיות כתובות מתחתיה עם אינדנטציה (....). תוכלו להוסיף אינדנטציה על ידי לחיצה על מקש Tab במקלדת.
לחיצה על כפתור ה- **times** תוסיף תחילת לולאה פשוטה בקוד שלכם:
“3.times ->”



השמה למשתנים. משתנה הוא כמו יחידת אחסון. אנחנו מאחסנים בו נתונים, ואנחנו משתמשים בו רק כאשר אנחנו צריכים אותו.
השמה למשתנה מורכבת ממזהה (X) וערך (בדוגמא בצד ימין - 10). הפרדה זו של שם וערך מאפשרת להשתמש בשם באופן עצמאי מהמידע שהוא מייצג. אנחנו יכולים להשתמש ב-X בעת כתיבת התכנית, בלי לדעת מה הערך שלו יהיה כאשר ההוראות תבוצענה.

"say" יגרום לכך שבסמוך לקוף תופיע בועת דיבור עם הטקסט שהקלדנו, למשל:

```
say "Boo!"
```

יגרום לקוף לומר "Boo!"

אנחנו משתמשים בגרשיים ("") סביב הביטוי שאנחנו רוצים שהקוף יאמר, כדי שהמחשב יבין שהטקסט שהקלדנו הוא לא משתנה. נסו להשתמש ב"say" כאשר יש עכבר בסביבה. לחיצה על כפתור ה- **say** תוסיף את המילה "say" בקוד שלכם.

בפונקציה "DistanceTo" נשתמש עם פקודה אחרת כמו "step" או "say" ואובייקט. להשתמש ב-"distanceTo" זה כמו לשאול שאלה:, "מה המרחק לבננה?" התשובה היא מספר, המחושב על ידי המחשב, שמייצג את המרחק.

דוגמה:

```
step distanceTo banana
```

המחשב ימדוד את המרחק בין הקוף והאובייקט (הבננה). אז הוא ישתמש במספר שהוא הגיע אליו כדי לבצע מה שהורינו לו, תוך שימוש בערך הנמדד כארגומנט ל"step".

לחיצה על כפתור ה- **distanceTo** תוסיף את המילה "distanceTo" בקוד שלכם.

זוהי לולאת "for". נשתמש בלולאת "for" כאשר יהיה לנו אוסף של אובייקטים ונרצה לחזור על פעולה המתייחסת לכל אחד מהם באופן ספציפי. לולאת ה-"for" תמשיך לפעול עד שכל הפעולות יעשו על כל האובייקטים שבאוסף שלנו (מערך).כאשר המחשב מבצע את הלולאה הזו, הוא מחליף את שם המשתנה עם הפריט הראשון באוסף. אחרי שהוא מסיים עם הפריט הראשון, הוא עובר לשני, וכך הלאה.

דוגמה:

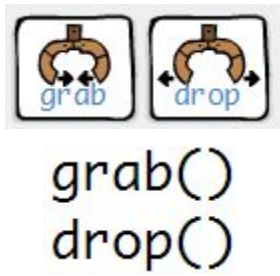
```
for b in bananas
    for c in crocodiles
        c.turnTo b
    turnTo b
    step distanceTo b
```

כמו כן, אנחנו יכולים להשתמש בלולאת "for" בתוך לולאת "for"; הדוגמה מימין נלקחה משלב מספר 70.

לחיצה על כפתור ה- **for** תוסיף את הטקסט הבא בקוד שלכם. שימו לב לשורת ההערה:

```
for name in array
    # Your code here
```

"()grab " ו-"()drop" הן פונקציות ללא ארגומנט המשמשות אותנו כדי לעזור לעכבר לאסוף גפרורים. פונקציות ללא ארגומנט מבצעות פעולה כלשהי, אך לא מחייבות אותנו להעביר קלט כלשהו. לחיצה על הכפתורים **grab** או **drop** תוסיף את המילים "()grab" או "()drop" בקוד שלכם, בהתאמה.



זוהי שורת הערה. היא מסומנת בקוד באמצעות סמל הסולמית (#) בתחילתה. המחשב לא מתייחס אל שורה זו כהוראה, ושורה זו לא תחשב בספירת השורות לקבלת שלושה כוכבים. במקום, נעשה בה שימוש על ידי המתכנתים שכותבים וקוראים את הקוד כדי להבין אחד את השני.

```
# By the way, this is a comment line.
# It starts with a '#', like this!
```

פונקציה היא סט של הוראות אשר מבצעת משימה מסוימת. המחשב יבצע את הפונקציה רק כאשר אנחנו נקרא לה, כלומר, נשתמש בה בקוד שלנו.

זוהי דוגמה לאיך אנחנו מגדירים פונקציה:

```
goto = (t) ->
    turnTo t
    step distanceTo t
```

ברגע שהגדרנו פונקציה, יופיע לחצן ריק חדש עם השם של הפונקציה.
כאשר אנחנו רוצים לקרוא לפונקציה בקוד, אנחנו קוראים לפונקציה עם שם האובייקט שעליו אנחנו רוצים שיבוצעו הפעולות שבתוך הפונקציה:

```
goto match
grab()
goto pile
drop()
```

לחיצה על כפתור ה- **function** תוסיף את הטקסט הבא בקוד שלכם. שימו לב לשורת ההערה:

```
function_name = (argument) ->
….#your code here
```

לולאת "until" מכילה בלוק של קוד שימשיך לפעול עד שתנאי ספציפי יתמלא. תנאי זה נקרא תנאי לולאה. המחשב בודק את המצב בהתחלה. אם התשובה כוזבת, הלולאה תמשיך לפעול. היא תעצור רק כאשר התשובה תהיה אמת.
לחיצה על כפתור ה- **until** תוסיף את הטקסט הבא בקוד שלכם. שימו לב לשורת ההערה:

```
until condition
….#your code here
```

דוגמה:

```
until near match
    step 1
```

אנחנו יכולים להשתמש בפונקציה "near" כתנאי לולאה. הערך שמוחזר על ידי הפונקציה "near" יקבע מתי לולאת ה-"until" תפסיק להתבצע. לחיצה על כפתור ה- **near** תוסיף את המילה " near" בקוד שלכם.

בשלב 96 אנחנו פוגשים את החתול. החתול יתקוף את העכבר אם הוא יראה אותו, ולכן אנחנו חייבים לחכות שהחתול יירדם.
"()sleeping" היא פונקציה שבה אנחנו יכולים להשתמש כתנאי לולאה. כשנשתמש ב-"sleeping()", אנחנו צריכים לכתוב:

```
until cat.sleeping()
    wait()
```

כדי לחכות שהחתול יירדם, נצטרך להשתמש בפונקציה "()wait".
זוהי פונקציה ללא ארגומנט אשר גורמת לקוף לחכות למשך שניה אחת. כאשר נשתמש ()wait עם until, נגרום לקוף לחכות עד שתנאי הלולאה יתממש, לדוגמא:

```
until cat.sleeping()
    wait()
```

פונקציית ה- "goto" (לך ל) גורמת לקוף לעשות "turnTo" (להסתובב ל) לאובייקט ו"step distanceTo"- (לצעוד את המרחק ל) מאובייקט מסוים זה. שימוש בפונקציה "goto" יקצר את הקוד ויהפוך אותו לקריא יותר. השתמשו בו עבור כל אובייקט שעל המסך כארגומנט: בננה ("goto banana"), גשר ("goto bridge"), וכו'.
לחיצה על כפתור ה-**goto** תוסיף את המילה "goto" בקוד שלכם.

בשלב 101 אנו פוגשים את העז אשר עוזרת לשבור את הקרח סביב הבננות הקפואות. הפונקציה "hit()" (הכה) תגיד לעז להכות בבננות הקפואות כדי שהקוף יוכל להשיג אותן, אך היזהרו מלהשתמש בה עם בננות לא קפואות.
כדי להשתמש ב- "hit()", אנו צריכים לכתוב "goat.hit()", שימו לב שהשניים מופרדים על ידי נקודה (.)
לחיצה על כפתור ה-**hit**, תוסיף את המילה "hit()." בקוד שלכם.

"frozen()" היא פונקציה ללא ארגומנט אשר משתמשים בה כדי לשאול "is the banana frozen?" ("האם הבננה קפואה"), אנו משתמשים בה על ידי כתיבת "banana.frozen()". הפונקציה אז מחזירה ערך שהוא כן או לא לפי מצב האובייקט. השתמשו ב- "frozen()" כתנאי לפקודת ה-if, כך:

```
if banana.frozen ()
```

בכדי להורות למחשב לבצע פעולות רק if!
לחיצה על כפתור ה- **frozen** תוסיף את המילה "frozen()" לקוד שלכם.

אנו משתמשים ב- "if" כשאנו רוצים שהקוד שלנו יחליט מה לבצע בהסתמך על תנאי מסוים. יכול להיות שהמצב לא יהיה ידוע לנו כשאנו כותבים את הקוד שלנו, והמחשב יצטרך להחליט מה לעשות בזמן שהקוד מבוצע.

לחיצה על כפתור ה- **if** תוסיף את הקוד הבא לקוד שלכם:

```
if condition
    # Your code here
```

"green()" היא פונקציה ללא ארגומנט אשר יכולה להיות כתנאי לפקודת ה-if. פונקציה זו תשאל "is the banana green?" ("האם הבננה ירוקה?"), ותחזיר ערך שהוא כן או לא על פי מצב האובייקט.
אנו משתמשים ב- "green()" כך:

```
if banana.green()
```

לחיצה על כפתור ה- **green** תוסיף את המילה "()green" לקוד שלכם.

פקודת if עוזרת למחשב להחליט מה לעשות בהתבסס על תנאי מסוים. כשאנו משתמשים בפקודת if-else, אנו אומרים למחשב מה לבצע במידה והתנאי אמת (תחת ה- "if"), ומה לבצע במידה והתנאי לא התקיים, ז"א הוא שקר (תחת ה- "else").

if [condition]

code to be executed if the condition is true

else

code to be executed if the condition is false

יכול להיות שהתנאי לא ידוע לנו כשאנו כותבים את הקוד שלנו, אך אנו אומרים למחשב מה לעשות בשני המקרים, והוא יחליט מה לעשות בזמן הרצת הקוד.

לחיצה על כפתור ה- **if-else** תוסיף את הקוד הבא לקוד שלכם:

```
if condition
    # Your code here
else
    # And more here
```

בשלב 120 אנו פוגשים את הנמר והדוב. הם יתקפו את הקוף שלנו במידה ויראו אותו, לכן אנו צריכים להמתין עד שהם יירדמו או שדעתם תוסח על ידי משחק. "()playing" ("משחק()") היא פונקציה נוספת שאנו יכולים להשתמש בתנאי הלולאה שלנו. כשמשתמשים ב- "playing", אנו צריכים לכתוב:


```
until tiger.playing()
    wait()
```

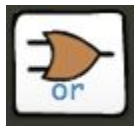
לחיצה על כפתור ה- **playing** תוסיף את המילה "()playing" לקוד שלכם.

---

and הוא *אופרטור לוגי*: הוא מבצע פעולה לוגית. אנו משתמשים בו עם לולאת until בכדי ליצור תנאי לולאה אשר עשוי מ-2 תנאים שונים. בצורה זו אנחנו גורמים למחשב לחכות עד ש-2 התנאים אמת, לפני ביצוע הקוד שלמטה, למשל:


```
until tiger.sleeping() and bear.sleeping()
    wait()
```

לחיצה על כפתור ה- **and**, תוסיף את המילה "and" לקוד שלכם.



or הוא *אופרטור לוגי*: הוא מבצע פעולה לוגית. אנו משתמשים בו עם לולאת until בכדי ליצור תנאי לולאה אשר עשוי מ-2 תנאים שונים. בצורה זו אנו גורמים למחשב לחכות עד **שלפחות אחד** מהתנאים אמת , לפני הרצת הקוד שלמטה, לדוגמא:

```
until tiger.sleeping() or tiger.playing()
    wait()
```

לחיצה על כפתור ה- **or** תוסיף את המילה "or" בקוד שלכם.

not הוא *אופרטור לוגי*: כשאנו משתמשים באופרטור ה-"not" אנו למעשה הופכים את ה-כן וה-לא להיפוך, או במילים אחרות: not הופך כן ל-לא ולא ל-כן. בשימוש ב- "not" אנו יכולים להורות לקוף ללכת לבננה אם היא לא רקובה. זה יראה כך:

```
if not banana.rotten()
    goto banana
```

לחיצה על כפתור ה-**not** תוסיף את המילה "not" לקוד שלכם.

חלק מהבננות צהובות וטעימות, אך חלק מהבננות רקובות ואנו לא באמת רוצים לאכול אותם. ()rotten היא פונקציה ללא ארגומנט אשר מחזירה כן (yes) אם הבננה רקובה, ולא (no) אם לאו. ניתן להשתמש בפונקציה זו כתנאי עם פקודת if. אנו יכולים גם להשתמש ב- ()rotten עם not, כדי להורות לקוף לגשת לבננה לא רקובה ((not banana.rotten()).

לדוגמה:

```
if b not.rotten()
    goto b
```

לחיצה על כפתור ה- **rotten** תוסיף את המילה "()rotten" לקוד שלכם.

()health (בריאות) היא פונקציה ללא ארגומנט. משתמשים בה החל משלב 142, כשהקוף נלחם בגורילה. בדרך כלל בקודמאנקי אנו לא מסיימים שלב אם נכשלנו ולא אספנו את כל הבננות. אך החל מנקודה זו, אנו יכולים להיכשל גם אם בריאותנו אוזלת.

משתמשים בפונקציית ה- ()health יחד עם האופרטורים == (שווה ל) ו- > (קטן מ). אנו יכולים לראות את מצב בריאותנו בפינה הימנית העליונה של הבמה, שם נמצא מונה אשר מסומן עם לב כחול.

לחיצה על כפתור ה- **health** תוסיף את המילה "()health" לקוד שלכם.

אנו משתמשים באופרטור השוויון (סימני שווה כפולים) בכדי להשוות בין 2 ערכים, כמו בפקודת if. לדוגמה: "if a == 3" אנו קוראים כ-"if a is equal to three" (אם a שווה ל-3).

בקודמאנקי אנו משווים בין ערכים כך:

```
if health() == 100

    goto banana
```

לחיצה על כפתור ה- **==** תוסיף "==" לקוד שלכם.

---

less than (פחות מ) הוא אופרטור שאנו משתמשים בו בכדי להשוות ערכים נומריים כמו מרחקים או ערכי בריאות. אנו יכולים להשתמש בו כדי לאפשר למחשב להחליט לאיזה healthZone (אזור בריאות) ללכת על ידי חישוב איזה healthZone קרוב יותר.

לדוגמה:


```
if d0 < d1

    goto healthZones[0]

else

    goto healthZones[1]
```

לחיצה על כפתור ה- **<** תוסיף את הסימן "<" לקוד שלכם.

פקודות return (חזור) מאפשרות לפונקציה להגדיר ערך חזרה לקוד שקרא לאותה הפונקציה. לדוגמה:

```
yummy = (x) ->

    return not x.rotten()
```

הפונקציה "yummy" מבקשת מהמחשב להחזיר ערך של "yes" או "no" עבור השאלה: "is x not rotten?" ("האם x לא רקוב?").

לחיצה על כפתור ה- **return** תוסיף את המילה "return" לקוד שלכם.

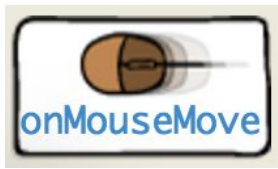
בשלב 160 אנו פוגשים את העורבים. הם יושבים גבוה למעלה ומתבוננים בנו, ואם לא ניזהר הם יתקפו אותנו. בכדי לעבור אותם, אנו צריכים להבריח אותם. אנו משתמשים בפונקציה watching() (צופים) כדי לוודא שהם באמת נבהלו ולא מתבוננים בנו יותר:


```
until not crow.watching()

    say "Boo!"

goto banana
```

לחיצה על כפתור ה- **watching** תוסיף את המילה "watching()" לקוד שלכם.

פונקציית ה-onKey נקראת בכל פעם שאנו לוחצים על מקש במקלדת. הארגומנט של פונקציה זו הוא המקש עליו לחצנו במקלדת. לחיצה על כפתור ה-**onKey** תוסיף את תחביר פונקציית ה-onKey לקוד שלכם עם הערות כדוגמה לשימוש בארגומנט בקוד שלכם:

```
onKey = (key) ->
    # Example:
    #if key == 'w'
     #   step 1
```



הפונקציה onMouseMove נקראת בכל פעם שבה אנו מזיזים את העכבר. הארגומנט של פונקציה זו הוא מיקום העכבר pos (המקום בו נמצא הסמן).

מיקום העכבר כולל את מיקום על ציר האיקס (pos.x) ואת מיקום על ציר ה-וואי - (pos.y).

לחיצה על כפתור **onMouseMove** תוסיף את תחביר פונקציית ה-onMouseMove לקוד שלכם עם הערות כדוגמה לשימוש בארגומנט בקוד.

```
onMouseMove = (pos) ->
    # Example:
    #turnTo pos
```

הפונקציות setX ו-setY קובעות את מיקום x ו-y, בהתאמה, בהתבסס על הארגומנט. הארגומנט הוא מיקום x ו-y החדש.

פונקציות setX ו-setY מתייחסות לדמות העטלף, כשאנו קוראים לפונקציות אלה עלינו לכתוב bat.setX ו-bat.setY.

לחיצה על כפתורי **setX** ו-**setY** תוסיף את המילים "setX" ו-"setY" לקוד שלכם, בהתאמה.

"onClick" היא פונקציה ללא ארגומנט. פונקציית ה-onClick נקראת כשאנו לוחצים על העכבר. אנו מגדירים את הפונקציה לדמות שעליה ברצוננו ללחוץ. לדוגמה, אם נרצה ללחוץ על הקוף ועל התנין נגדיר שתי פונקציות:

- monkey.onClick
- crocodile.onClick.

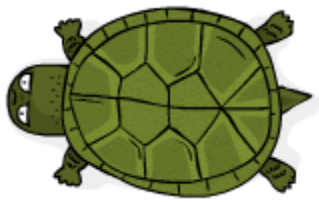
לחיצה על כפתור **onClick** תוסיף את תחביר פונקציית ה-onClick לקוד שלכם. שימו לב שעליכם להוסיף את הדמות לה הגדרתם את ה-onClick לפני הנקודה:

```
.onClick = () ->

    # Your code here
```

"()toss" היא פונקציה ללא ארגומנט המשמשת בשלבים האחרונים של קודמאנקי לזריקת אגוזי קוקוס על ההיפופוטם ועל הגורילה.

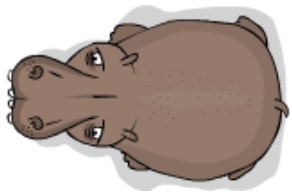
לחיצה על כפתור **toss** תוסיף את המילה ()toss בקוד שלכם.

סקירת הדמויות

דמות	תיאור
	גורדו, שנקרא על שמו של הקוף הראשון בחלל, הוא המדריך שיעזור לכם וייתן לכם הוראות לאורך הדרך. ההערות שלו מצחיקות ומועילות תמיד אפשר ללחוץ עליו ולקרוא את ההוראות בשנית.
	הקוף הוא הדמות הראשית. תצטרכו לעזור לו לאסוף בננות על ידי כתיבת שורות קוד. רק שתדעו, קופים לא אוהבים להירטב והם מאוד ידידותיים.
	בשלב מספר 13 ,תוכלו לפגוש את הצב הנאמן שלנו. הצב יעזור לך להשיג את הבננות הערמומיות האלו. כדי להורות לצב להסתובב "turn" או "step" אנחנו צריכים קודם כל ללחוץ עליו. זה יכתוב את השם שלו בקוד, ולאחר מכן יפריד אותו מהפעולה שאנחנו רוצים שהוא יבצע באמצעות שימוש בנקודה (.). **דוגמה:** `turtle.step 10`
	בשלב 50 נפגוש את הבונים. הבונים אוהבים מאוד עץ, והם הסכימו לעזור לכם לחצות את המים ולקבל יותר בננות על ידי דריכה על העץ שלהם. הם יכולים לבצע רק "step" קדימה ואחורה. כדי להשתמש בהם אנחנו צריכים להשתמש בנקודה (.) בין שמם לבין הפעולה שאנחנו רוצים שהם יבצעו. לדוגמה: `beavers[0].step 10`

	את התנינים נפגוש בשלב מספר 56. הם משמשים ליצירת גשר על המים, כדי לעזור לקוף להגיע לבננות. הם יכולים לבצע רק "turn" או "turnTo". אנחנו בדרך כלל נשתמש בתנינים עם לולאות "for".
	`for c in crocodiles`
	`····c.turn right`

	אנחנו פוגשים את העכברים כמה פעמים במשחק, אבל רק בשלב מספר 71 אנחנו באמת יכולים לשלוט בהם. עכברים אוהבים לאסוף פריטים. בחלק השלישי, אנחנו עוזרים להם לאסוף גפרורים. כדי לעשות את זה, נשתמש בפונקציות הפשוטות הבאות: "grab()" ו-"drop()".
	את הנמלים נפגוש בשלב מספר 89 והן יעזרו לנו להדגים את לולאת "until". הנמלים גוררות את הגפרורים היקרים שלנו, ואנחנו צריכים לבצע "step" עד שנגיע אליהם וניקח את הגפרורים בחזרה.
	עם החתול ניפגש בשלב מספר 96 והוא יעזור לנו להדגים את לולאת "until" עם הפונקציה "wait()".החתול יתקוף את העכבר אם הוא יראה אותו, ולכן אנחנו חייבים לחכות שהוא יירדם לפני שנצא לאסוף גפרורים.
	בשלב 101 אנו מכירים את העז אשר תעזור לנו לשבור את הקרח סביב בננות קפואות. הפונקציה "hit()" תאמר לעז להכות בבננות הקפואות כדי שהקוף יוכל להשיג אותן, אך היזהרו מלהשתמש בה על בננות לא קפואות.
	בננות קפואות משחקות אותה קשות להשגה. כדי לאסוף אותן, אנו צריכים לשלוח קודם את העז לשבור את הקרח, ורק אז נוכל לאסוף אותן.
	קופים לא אוהבים בננות ירוקות, אך עזים כן! תנו לעז לאסוף בננות ירוקות כדי להשאיר אותה מרוצה.
	אנו מכירים את הנמר בשלב 120, הוא יתקוף את הקוף אם הוא יראה אותו, אז אנו צריכים לחכות שהוא יירדם או ישחק לפני שנוכל לצאת לאסוף בננות.
	אנו מכירים את הדוב בשלב 121, הוא יתקוף את הקוף אם הוא יראה אותו, אז אנו צריכים לחכות שהוא יירדם או ישחק לפני שנוכל לצאת לאסוף בננות.
	אף אחד לא אוהב בננה רקובה, היזהרו לא לדרוך על אחת כי אתם עלולים להצטער על כך....
	אנו מכירים את ה- healthZone (אזור הבריאות) בשלב 145. זהו מרחב על הבמה שעליו הקוף יכול לעמוד בכדי להבריא. אתם

	יכולים להבריא את הקוף בחזרה ל-100 על ידי עמידה על ה-healthZone והמתנה.
	הגורילה גנבה את הבננות שלנו בתחילת הדרך כשהתחלנו את ההרפתקה שלנו. אבל עכשיו, היא חזרה, והיא כועסת. בשלב 143 הגורילה חוזרת, והיא זורקת עלינו קוקוסים! ודאו שאתם ממלאים את בריאותכם ב- healthZone כשיש גורילה בסביבה והבריאות שלכם מתדרדרת. בשלב 200 עלינו לזרוק אגוזי קוקוס על הגורילה עד שבריאותה תגיע ל-0. אפשר לראות את בריאותה של הגורילה בצדו השמאלי של הסרגל מעל לשלב.
	אנו מכירים את העורבים בשלב 160. הם יושבים גבוה למעלה ומתבוננים בנו, ואם לא ניזהר הם יתקפו אותנו. בכדי לעבור אותם, אנו צריכים להבריח אותם בעזרת "say"..
	בשלב 177 נכיר את השער. השער הוא מכשול שהקוף לא יכול לעבור דרכו מבלי לפתוח אותו תחילה. כשהקוף מתקרב לשער, סיסמת השער מופיעה מעליו. הקוף "אומר" את הסיסמה (בלחיצה על אותיות הסיסמה במקלדת) כדי לפתוח את השער.
	בשלב 182 נכיר את העטלף. העטלף יכול לעוף, לקחת את הבננה ולהביא אותה לקוף שלנו. אנו שולטים בעטלף בהזזת העכבר ובשינוי מיקום x ו-y של העטלף.
	בשלב 196 נכירו את ההיפופוטם. ההיפופוטם חוסם את דרכו של הקוף. על מנת להזיז את ההיפופוטם, על הקוף לקחת תחילה קוקוס ואז בלחיצה על ההיפופוטם הקוף פונה אל ההיפופוטם וזורק עליו את הקוקוס. ההיפופוטם זז והקוף יכול לעבור.