

python

FILE CREATION

Create a New File

כדי ליצור קובץ חדש ב-Python, יש להשתמש בפונקציה `open()` עם אחד מהפרמטרים הבאים:

"x" - Create - will create a file, returns an error if the file exist

```
f = open("myfile.txt", "x")
```

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

```
f = open("myfile.txt", "w")
```

FILE MODIFICATIONS

Write to an Existing File

`open()`: כדי לכתוב לקובץ קיים, יש להוסיף פרמטר לפונקציה:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

```
#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

```
#open and read the file after the overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

```
Woops! I have deleted the content!
```

CONTROL FLOW

הבקרה על הזרימה של התוכנית קובעת את הסדר שבו הקוד של התוכנית מתבצע.

הבקרה על זרימת התוכנית ב-Python נעשית בעזרת הצהרות תנאים, לולאות וקריאות לפונקציות.

ל-Python יש שלושה סוגים של מבני בקרה:

Sequential

רציף - מצב ברירת המחדל שבו הפקודות מתבצעות אחת אחרי השנייה בצורה רציפה. בעזרת פקודות רציפות ניתן לפתח תוכניות פשוטות.

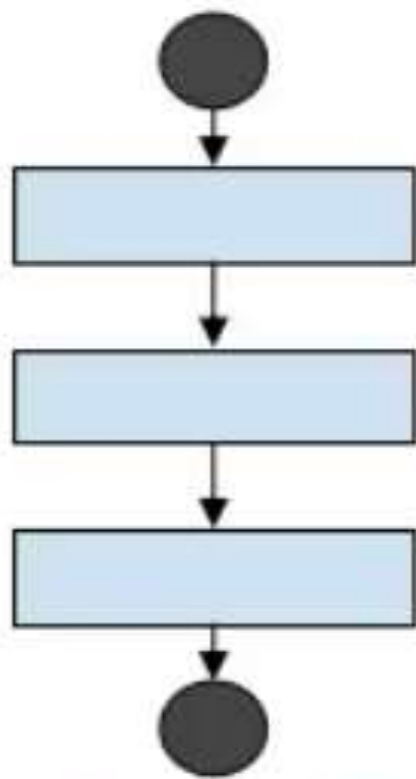
Conditional

תנאי - פקודות מתבצעות על בסיס תנאי. אם התנאי נכון, קבוצה אחת של פקודות מתבצעת, ואם לא - קבוצה אחרת. הצהרות תנאים משמשות לרוב בתוכניות מורכבות.

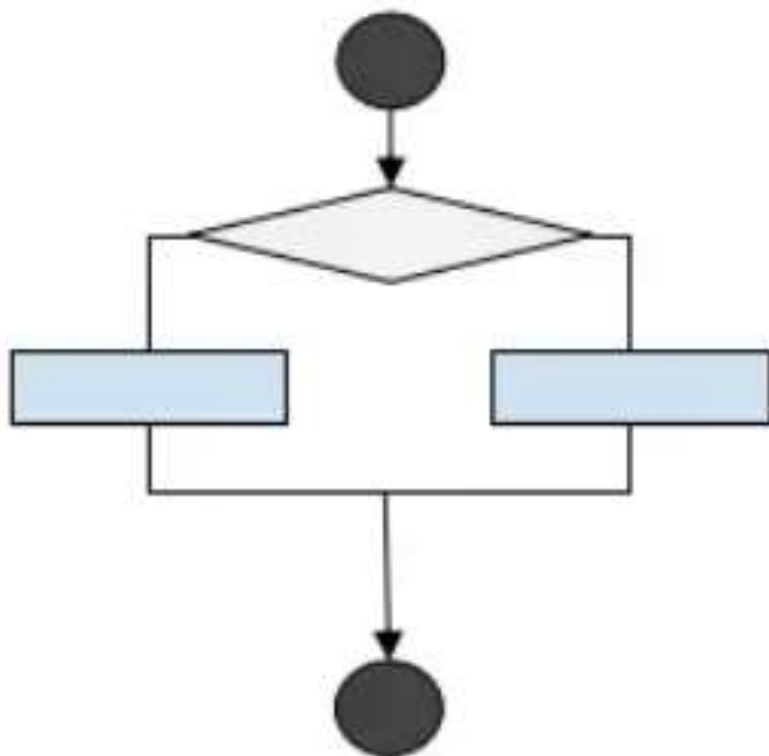
Repetition (Looping)

חזרה (לולאה) - משמש לחזרה על קטע קוד מספר פעמים.

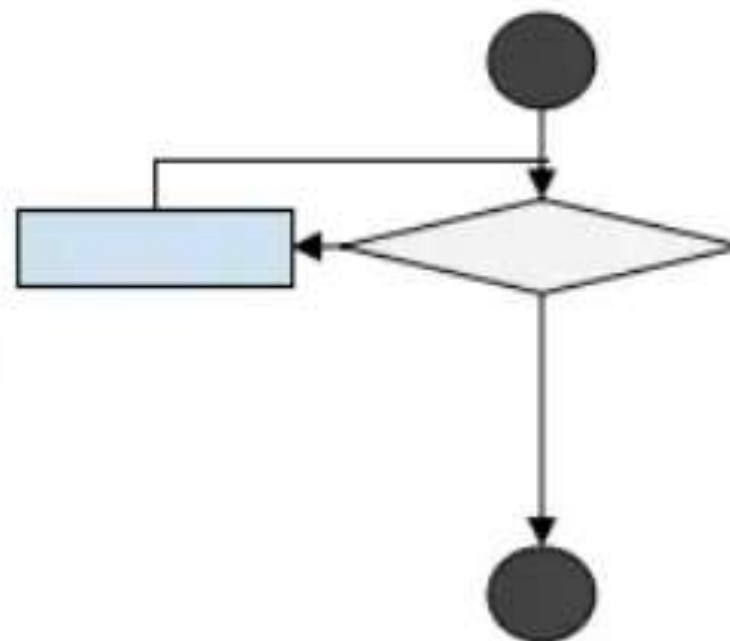
CONTROL FLOW



Sequential



Conditional



Looping

SEQUENTIAL STATEMENTS

הפקודות הרציפות מבוצעות שורה אחרי שורה.

```
print("one")  
print("two")  
print("three")
```

Output::

```
one  
two  
three
```

CONDITIONAL STATEMENTS

הצהרות תנאים נקראות גם הצהרות קבלת החלטות.

ישנם שלושה סוגים של הצהרות תנאים ב-Python:

- if statement
- if-else statement
- if elif else

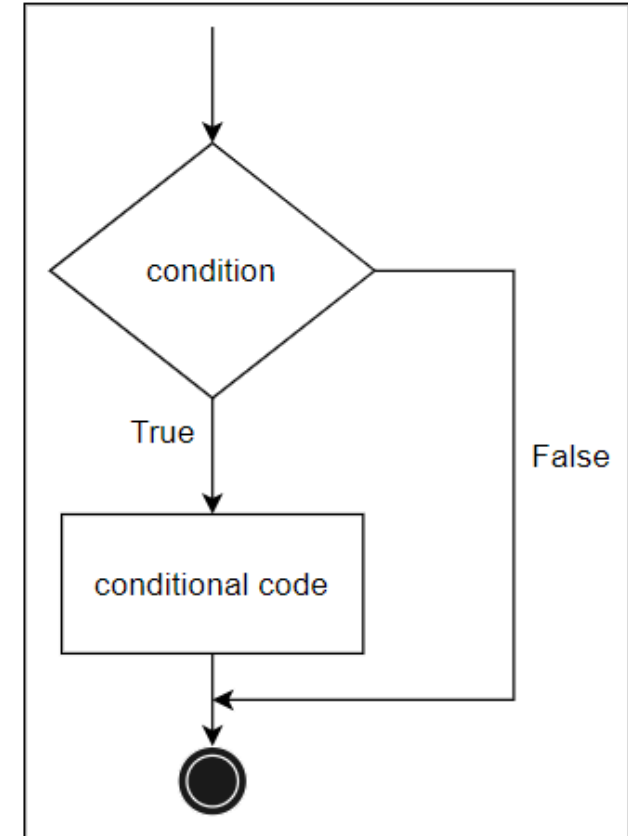
CONDITIONAL STATEMENTS

IF STATEMENT

הצהרת if עוזרת להריץ קוד מסוים, אבל רק כאשר תנאי מסוים מתקיים או מתמלא ל-if יש רק תנאי אחד לבדוק
הצהרת if מכילה ביטוי/תנאי, כאשר סימן הנקודתיים (:) הוא חובה, אחרת יתקבל שגיאת תחביר

התוצאה של התנאי היא מסוג bool, כלומר True או False
אם התוצאה של התנאי היא True, הפקודות שבבלוק של if יבוצעו
אם התוצאה היא False, הפקודות בבלוק לא יבוצעו

```
n = 10  
if n / 2 == 5:  
    print("n / 2 = 5")
```

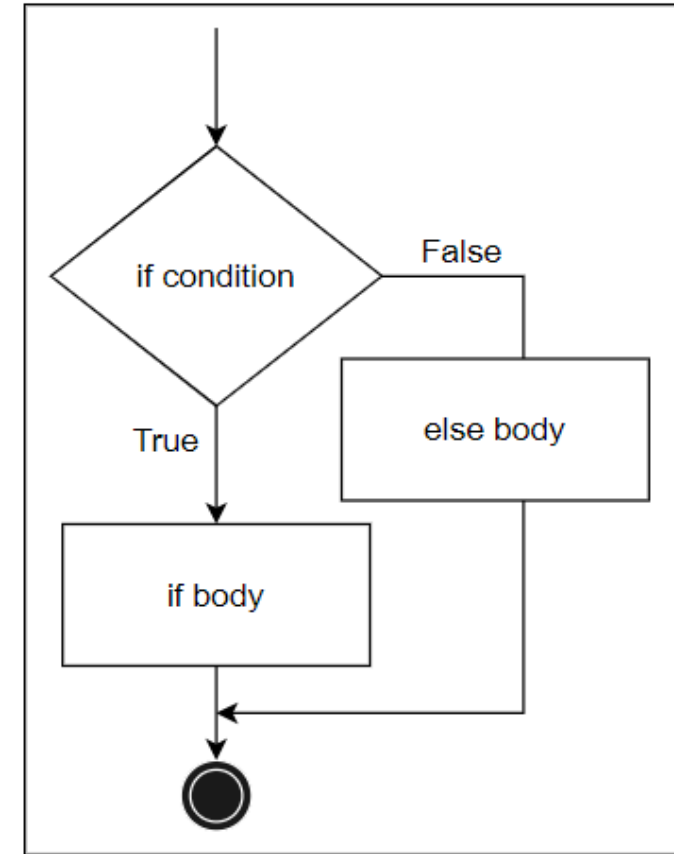


CONDITIONAL STATEMENTS

IF-ELSE STATEMENT

הצהרת if-else בודקת את התנאי ותבצע את הבלוק של if אם התנאי הוא True, אך אם התנאי הוא False, יבוצע הבלוק של else. הצהרת if מכילה ביטוי/תנאי כאשר סימן הנקודתיים (:) הוא חובה אחרי התנאי ואחרי else, אחרת תתקבל שגיאת תחביר. התוצאה של התנאי היא מסוג bool, כלומר True או False. אם התוצאה של התנאי היא True, הפקודות שבבלוק של if יבוצעו. אם התוצאה היא False, הפקודות שבבלוק של else יבוצעו.

```
1 n = 10
2 if n / 2 == 5:
3     print("n / 2 = 5")
4 else:
5     print("it's not")
```

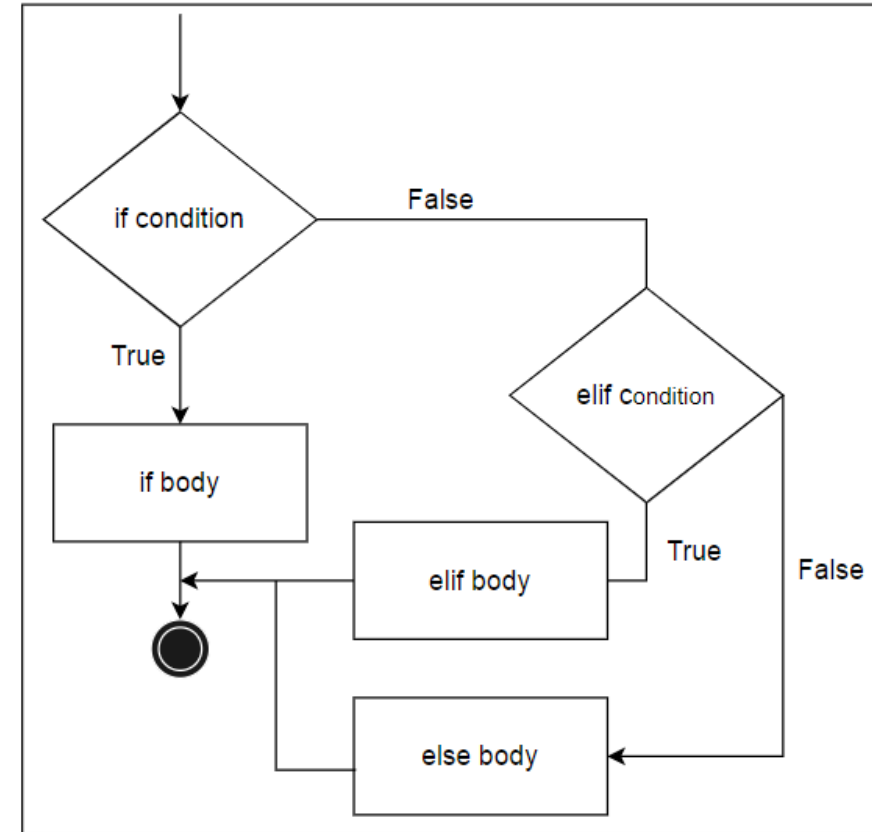


CONDITIONAL STATEMENTS

IF-ELIF-ELSE STATEMENT

הצהרת if-elif-else משמשת להרצה מותנית של פקודה או בלוק של פקודות
הצהרת if מכילה ביטוי/תנאי כאשר סימן הנקודתיים (:) הוא חובה אחרי כל
תנאי ואחרי else, אחרת תתקבל שגיאת תחביר
התוצאה של התנאי היא מסוג bool, כלומר True או False.

התנאים נבדקים לפי הסדר שנכתבו. אם תנאי אחד נכון, קבוצת הפקודות
הקשורה אליו תבוצע והביצוע ייצא מהבלוק מבלי לבדוק את שאר התנאים



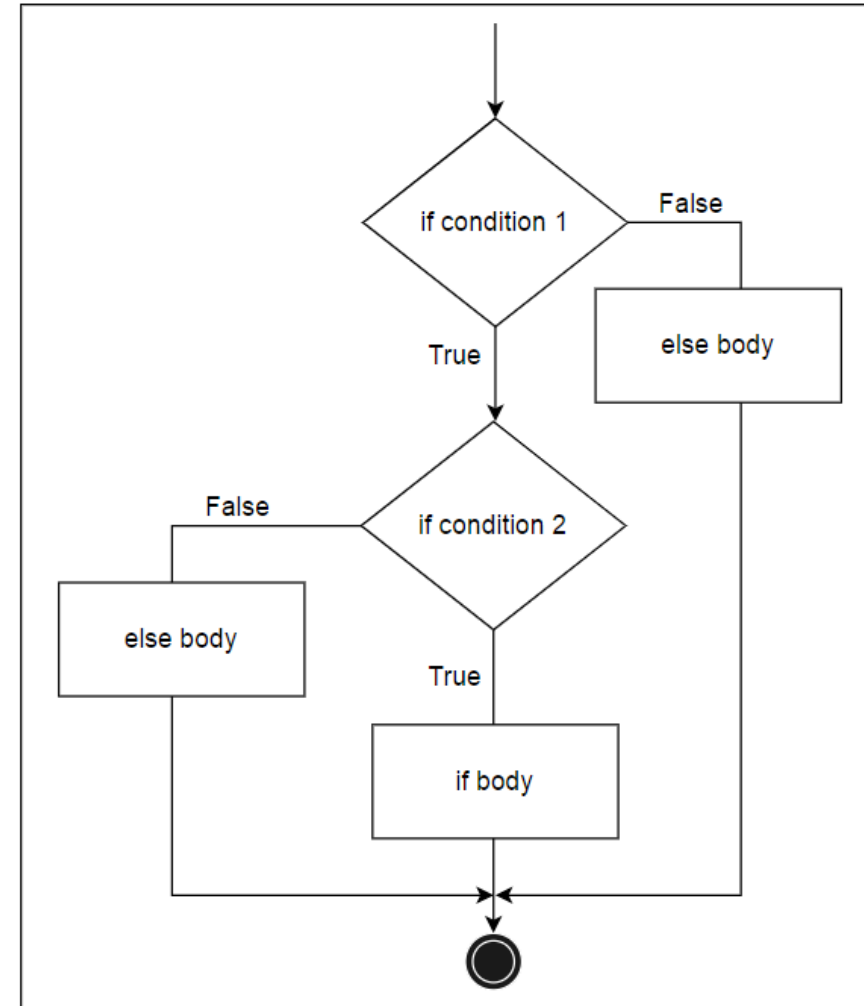
```
1 x = 15
2 y = 12
3 if x == y:
4     print("Both are Equal")
5 elif x > y:
6     print("x is greater than y")
7 else:
8     print("x is smaller than y")
```

CONDITIONAL STATEMENTS

NESTED IF

הצהרות if מקוננות הן הצהרת if שנמצאת בתוך הצהרת if אחרת.

```
1 a = 20
2 b = 10
3 c = 15
4 if a > b:
5     if a > c:
6         print("a value is big")
7     else:
8         print("c value is big")
9 elif b > c:
10    print("b value is big")
11 else:
12    print("c is big")
```



REPETITION STATEMENTS

הצהרת חזרה משמשת לחזרה על קבוצת (בלוק) של הוראות תכנות.

ב- Python יש לנו בדרך כלל שני סוגי לולאות/הצהרות חזרה:

for loop
while loop

REPETITION STATEMENTS – FOR LOOP

FOR LOOP

לולאת for משמשת לחזרה על רצף כמו רשימה list קבוצה set מילון dictionary או זוג tuple ניתן להריץ קבוצה של פקודות פעם אחת עבור כל פריט ברשימה, זוג או מילון.

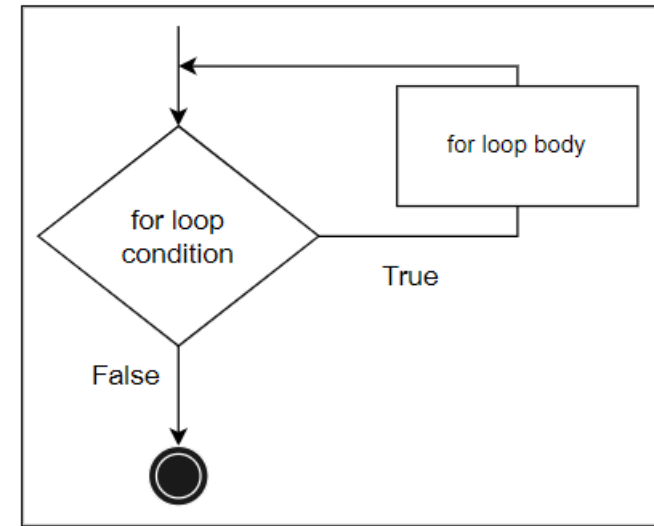
We can use indexing too : `for x in fruits[0:2]:`

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    print(x)
```

```
apple  
banana  
cherry
```

```
for x in "banana":  
    print(x)
```

```
b  
a  
n  
a  
n  
a
```



REPETITION STATEMENTS – FOR LOOP

FOR LOOP - ELSE

המילה השמורה else בלולאת for מציינת בלוק של קוד שיבוצע כאשר הלולאה מסתיימת.

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

```
for x in range(6):  
    print(x)  
else:  
    print("Finally finished!")
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP - ELSE

בלוק ה-else לא יבוצע אם הלולאה נעצרת על ידי הצהרת break.

Example

Break the loop when x is 3, and see what happens with the else block:

```
for x in range(6):  
    if x == 3: break  
    print(x)  
else:  
    print("Finally finished!")
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP – NESTED LOOPS

לולאה מקוננת היא לולאה בתוך לולאה

ה"לולאה הפנימית" תבוצע פעם אחת עבור כל איטרציה של ה"לולאה החיצונית"

Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]  
fruits = ["apple", "banana", "cherry"]
```

```
for x in adj:  
    for y in fruits:  
        print(x, y)
```


REPETITION STATEMENTS – FOR LOOP

FOR LOOP

ניתן להשתמש בלולאת for עם התבנית הבאה:

```
myfruits = ['apples', 'cherries', 'pears']  
  
for fruit in myfruits:  
    print(f"I like {fruit}")
```

```
I like apples  
I like cherries  
I like pears
```

```
fruitbasket = ['Banana', 'Mango', 'Apple', "Strawberry", "Lemon"]  
  
for fruit in fruitbasket:  
    if fruit == "Apple":  
        print(f"This is a beautiful {fruit}")  
    else:  
        print(fruit)
```

```
Banana  
Mango  
This is a beautiful Apple  
Strawberry  
Lemon
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP - LIST

ניתן להשתמש בלולאת for כדי להוסיף פריטים לרשימה.

```
list = []  
for i in range(5):  
    list.append("hello")  
print(list)
```

```
['hello', 'hello', 'hello', 'hello', 'hello']
```

REPETITION STATEMENTS – FOR LOOP

LOOP – BREAK STATEMENT

עם הצהרת `break` ניתן לעצור את הלולאה לפני שהיא סיימה לעבור על כל הפריטים.

Exit the loop when x is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

```
apple
banana
```

Exit the loop when x is "banana", but this time the break comes before the print :

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

```
apple
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP – CONDITIONS

We can use conditions words, as "AND" , "OR" :

```
numlist = [10, 22, 44, 30]

for i in numlist:
    if numlist[0] > 6 and numlist[3] > 40:
        print("both are bigger")
    elif numlist[1] == 22 or numlist[2] == 60:
        print("one is equal")
        break
    else:
        print("None of this is true")
```

REPETITION STATEMENTS – FOR LOOP

LOOP – CONTINUE STATEMENT

עם הצהרת continue ניתן לעצור את האיטרציה הנוכחית של הלולאה ולהמשיך לאיטרציה הבאה.

Do not print banana :

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    if x == "banana":  
        continue  
    print(x)
```

```
apple  
cherry
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP - THE RANGE() FUNCTION

כדי להריץ לולאה על קטע קוד מספר פעמים מוגדר, ניתן להשתמש בפונקציה `range()`. פונקציית `range()` מחזירה רצף של מספרים שמתחיל מ-0 כברירת מחדל, מעלה את המספר באחד (ברירת מחדל), ומסתיים במספר מוגדר.

Example

```
for x in range(6):  
    print(x)
```

שימו לב כי `range(6)` לא מחזירה את הערכים 0 עד 6, אלא את הערכים 0 עד 5.

REPETITION STATEMENTS – FOR LOOP

FOR LOOP - THE RANGE() FUNCTION

פונקציית `range()` מתחילה כברירת מחדל בערך 0, אך ניתן להגדיר ערך התחלתי על ידי הוספת פרמטר:
`range(2, 6)` כלומר הערכים מ-2 עד 6 (אך לא כולל את 6).

Example

Using the start parameter:

```
for x in range(2, 6):  
    print(x)
```

REPETITION STATEMENTS – FOR LOOP

FOR LOOP - THE RANGE() FUNCTION

פונקציית `range()` מעלה את הרצף בערך של 1 כברירת מחדל, אך ניתן להגדיר את ערך ההעלאה על ידי הוספת פרמטר שלישי: `range(2, 30, 3)`.

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):  
    print(x)
```

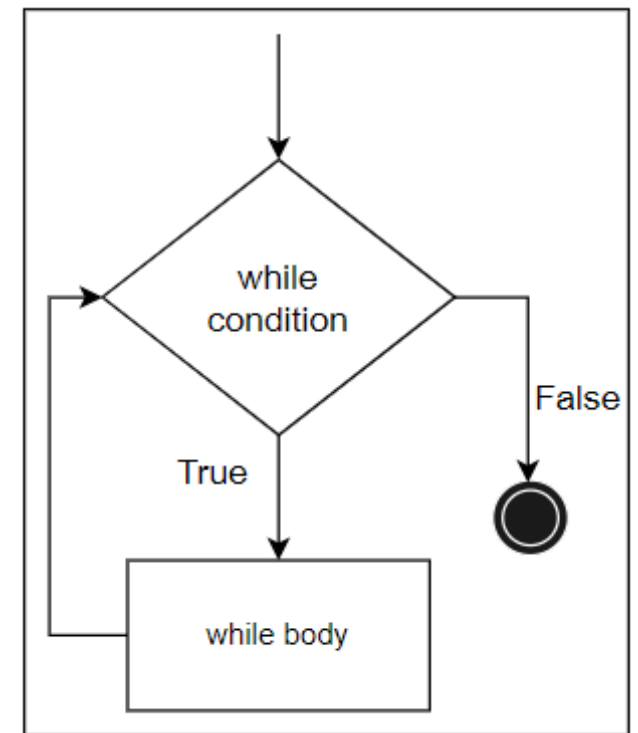

REPETITION STATEMENTS – WHILE LOOP

WHILE LOOP

ב-, Python לולאות while משמשות להרצת בלוק של פקודות שוב ושוב עד שתנאי מסוים מתקיים. לאחר מכן הביטוי נבדק שוב, ואם הוא עדיין נכון, הגוף מבוצע שוב. זה ממשיך עד שהביטוי הופך להיות שקרי.

```
i = 1
while i < 6:
    print(i)
    i += 1
```

```
1
2
3
4
5
```



REPETITION STATEMENTS – WHILE LOOP

WHILE LOOP – BREAK STATEMENT

עם הצהרת break ניתן לעצור את הלולאה אפילו אם התנאי של while הוא נכון.

Exit the loop when i is 3:

```
i = 1
while i < 6:
    print(i)
    if (i == 3):
        break
    i += 1
```

```
1
2
3
```

REPETITION STATEMENTS – WHILE LOOP


WHILE LOOP – CONTINUE STATEMENT

עם הצהרת `continue` ניתן לעצור את האיטרציה הנוכחית ולהמשיך לאיטרציה הבאה.

Continue to the next iteration if `i` is 3:

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)

# Note that number 3 is missing in the
result
```



```
1
2
4
5
6
```

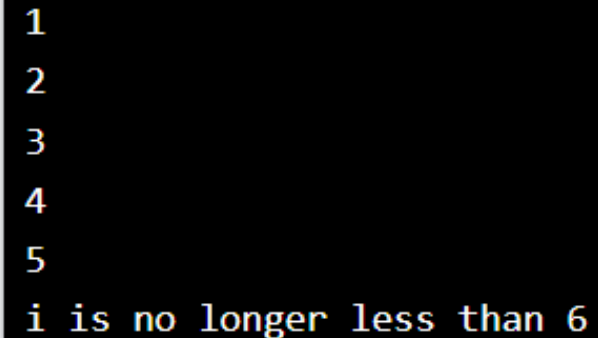
REPETITION STATEMENTS – WHILE LOOP

WHILE LOOP – ELSE STATEMENT

עם הצהרת `else` ניתן להריץ בלוק של קוד פעם אחת כאשר התנאי כבר לא נכון.

Print a message once the condition is false:

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

A terminal window with a black background and white text. It shows the output of the Python code: the numbers 1 through 5 on separate lines, followed by the message "i is no longer less than 6" on the final line.

```
1
2
3
4
5
i is no longer less than 6
```

RANDOM MODULE

ל-Python יש מודול מובנה שניתן להשתמש בו כדי לייצר מספרים אקראיים. יש לייבא אותו כדי להשתמש בו.

Pick a number between 0 and 5 :

```
import random  
x = random.randrange(5)  
print(x)
```

RANDOM MODULE

Method	Description
<code>seed()</code>	Initialize the random number generator
<code>getstate()</code>	Returns the current internal state of the random number generator
<code>setstate()</code>	Restores the internal state of the random number generator
<code>getrandbits()</code>	Returns a number representing the random bits
<code>randrange()</code>	Returns a random number between the given range
<code>randint()</code>	Returns a random number between the given range
<code>choice()</code>	Returns a random element from the given sequence
<code>choices()</code>	Returns a list with a random selection from the given sequence
<code>shuffle()</code>	Takes a sequence and returns the sequence in a random order
<code>sample()</code>	Returns a given sample of a sequence
<code>random()</code>	Returns a random float number between 0 and 1
<code>uniform()</code>	Returns a random float number between two given parameters
<code>triangular()</code>	Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters
<code>betavariate()</code>	Returns a random float number between 0 and 1 based on the Beta distribution (used in statistics)
<code>expovariate()</code>	Returns a random float number based on the Exponential distribution (used in statistics)
<code>gammavariate()</code>	Returns a random float number based on the Gamma distribution (used in statistics)
<code>gauss()</code>	Returns a random float number based on the Gaussian distribution (used in probability theories)
<code>lognormvariate()</code>	Returns a random float number based on a log-normal distribution (used in probability theories)
<code>normalvariate()</code>	Returns a random float number based on the normal distribution (used in probability theories)
<code>vonmisesvariate()</code>	Returns a random float number based on the von Mises distribution (used in directional statistics)
<code>paretovariate()</code>	Returns a random float number based on the Pareto distribution (used in probability theories)
<code>weibullvariate()</code>	Returns a random float number based on the Weibull distribution (used in statistics)

Exercises

Exercises

Exercise 1 :

Write a program to print multiplication table of a given number.

Exercise 2 :

Write a Python algorithm that asks the user to enter their age and display the message "you are major!" if the typed age is greater than or equal to 18 and the message "you are a minor!" if the typed age is less than 18

Exercise 3 :

Write an if statement that asks for the user's name via input() function. If the name is "Bond" make it print "Welcome on board 007." Otherwise make it print "Good morning NAME".
(Replace Name with user's name)

Exercises

Exercise 4 :

Create a program, using WHILE LOOP, that asks for the name, and add "ish" at the end, until the user put the word "EXIT".

Exercise 5 :

Create a program that asks the user for 2 numbers and adds it together.
If the user writes "EXIT", stop the program and print "Thanks, have a good day".

Exercise 6 :

Create a program with a counter. Print all the numbers until the counter arrives to 5, and print "We arrived at 5".

Exercises

Exercise 7 : EXTRA

Create a program that asks the user for 4 names of people and store them in a list.
When all the names have been given, pick a random one and print it.
Print also the new list with the names in it.

Exercise 8 : EXTRA

Create a guess game with the names of the colors.
At each round pick a random color from a list and let the user try to guess it.
When he does it, ask if he wants to play again.
Keep playing until the user types "no".

Q&A