**פייתון - הרצאה מספר 2**

**מבני בקרה – תנאים**

התנאי הבסיסי ביותר הוא ,if לאחר if יבוא ביטוי בוליאני כלשהו . אם ערך הביטוי הוא True אז יבוצע הקוד שלאחר ה - if

**מבנה משפט תנאי:**

**(ה- else אופציונאלי).**

**: תנאי if**

**הוראה/הוראות לביצוע**

הפקודה / פקודות שלאחר - if נמצאת / נמצאות בהזחה אינדנטציה יחסית לתנאי ה if - .

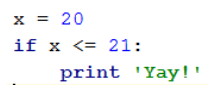
**דוגמאות:**

x = 20

if x != 20:

print ("???")

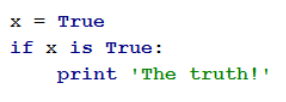
לא יודפס דבר



יודפס Yay!

**בדיקת שוויון מול משתנה בוליאני**:

כאשר מתבצעת השוואה למשתנה בוליאני נהוג להשתמש בביטוי is ולא להשוות ע"י ==.



**תנאי בוליאני מקוצר:**

צורת שימוש מקובלת נוספת, היא פשוט לרשום if ואז את שם המשתנה:

x = True

if x: // if x is True

print (" the truth !")

שתי הדוגמאות האחרונות בעצם זהות, הפלט שיתקבל : the truth !

**שימוש ב – is**

התנאי == בודק אם שני הצדדים של התנאי מכילים את אותם ערכים.

היות וכל משתנה בפייתון הוא עצם, הוא מצביע על מקום בזיכרון.

התנאי is בודק אם שני הצדדים של התנאי מצביעים על אותה כתובת בזיכרון.

דוגמה:

x = 5;

y = x;

if (x == y):

print ('x & y are equal')

if (x is y):

print (' x & y point to the same address ')

print ("id(x)",id(x))

print ("id(y)",id(y))

הפלט שיתקבל:

x & y are equal

x & y point to the same address

id(x) 1811967136

id(y) 1811967136

**הערה:** ההוראה id(x) - מדפיסה את כתובת המשתנה

**אם נרצה להגדיר בלוק של פקודות** שיתבצע במידה והתנאי מתקיים, אז ההוראות בבלוק יקבלו את **אותה אינדנטציה (הזחה)** יחסית לתנאי. הן יבוצעו כולן אם התנאי יתקיים.

**דוגמה:**



שימו לב להבדל בין הקוד לעיל לבין הקוד הבא :



השורה האחרונה, שמדפיסה ,21 תרוץ בכל מקרה בין שהתנאי מתקיים ובין שהוא אינו מתקיים. זאת מכיוון –שהיא נמצאת מחוץ לבלוק של תנאי ה. if

נושא ההזחה הוא קריטי בפייתון, מכיוון שבניגוד לשפות אחרות בהן יש סימונים שונים של "סוף בלוק ," לדוגמה: סוגריים מסולסלים ,בפייתון סוף הבלוק נקבע רק לפי ההזחה. על כן הכרחי שההזחה תהיה עקבית. אנחנו לא יכולים לעשות לפעמים הזחה של שני רווחים ולפעמים של ארבעה רווחים. חשוב לזכור גם שתווי רווח וטאב שונים זה מזה . כלומר, גם אם לנו נראה ששורה שיש בה טאב נמצאת בהזחה זהה לשורה אחרת שיש בה הזחה של ארבעה תווי רווח, מבחינת פייתון אלו תווים שונים לגמרי . לכן, אם התחלנו לעשות הזחה עם טאבים, נמשיך רק עם טאבים .

לא רק מה שנמצא בהזחה של טאב אחד שייך לבלוק. בתוך בלוק יכולים להיות עוד טאבים, לדוגמה עקב שימוש בתנאי if נוספים. כל הפקודות שנמצאות בהזחה של לפחות טאב אחד מתנאי ה- if שלנו שייכות לאותו בלוק.

**לדוגמה :**



מה יודפס כתוצאה מריצת הבלוק?

יורצו כל השורות חוץ מאשר אלו שנמצאות בבלוק של if x < 21 , כיוון שתנאי זה אינו

מתקיים. פלט התוכנית יהיה :

lower than 23

lower than 22

love

python

yes :)

**תנאים מורכבים**

אם יש לבצע משהו רק אם מתקיים תנאי א' וגם תנאי ב', או שמתקיים רק אחד מכמה תנאים אפשריים, או שמתקיים תנאי א' אך תנאי ב' אינו מתקיים. במקרים כאלו, נשתמש בתנאים המורכבים מ- and, or, not .

תנאי and משמעו "וגם ."לדוגמה, התנאי הבא יתקיים רק אם מתקיים גם x גדול מ - 20 וגם x קטן מ 22 :



ואפשר גם לכתוב את אותו ביטוי עם סוגריים, לעיתים יותר קל לקרוא אותו כך :



תנאי or משמעו "או ." לדוגמה, כדי שהתנאי הבא יתקיים, מספיק ש x- יהיה שווה ל 21- או כל מספר מעל :30



תנאי not מבצע היפוך. מקובל להשתמש בו יחד עם is לדוגמה:



כאשר מדובר בערך בוליאני, הדרך המקובלת היא לרשום not לפני שם המשתנה :



יודפס Yes!

דוגמה נוספת:

x = 7

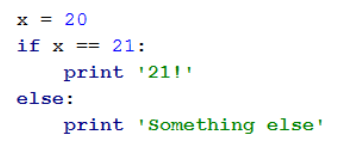
if not (x == 5):

print ("5!=7")

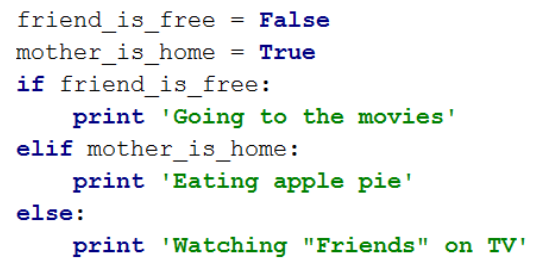
יודפס: 7 = ! 5

**שימוש ב- elif, else**

אחרי else יש לרשום נקודותיים וכמו אחרי הוראת if יש לרשום הוראה/ הוראות לביצוע תוך שימוש בהזחה.



כאשר יש מספר תנאים התלויים זה בזה, זאת אומרת שיש תנאי נוסף שאנחנו רוצים לבדוק, לדוגמה, אנחנו מתקשרים לחבר כדי שיבוא אתנו לסרט. אם הוא לא יכול, מתקשרים לאמא ושואלים מה יש לאכול בבית. אם אמא לא עונה – אנחנו נשארים בבית וצופים בשידור חוזר של הסדרה Friends.



בדוגמה זו נאכל עוגת תפוחים כיוון שתנאי ה- elif התקיים. התכנית לא נכנסה ל- else. נצפה בטלוויזיה רק אם תנאי ה- if וגם תנאי ה- elif לא יתקיימו.

**חישובים על מספרים**

ראינו כיצד ניתן לבצע את הפעולות החשבוניות הבסיסיות ביותר: חיבור, חיסור, כפל, וחילוק. בנוסף ראינו שאפשר לחשב חזקות באמצעות זוג כוכביות (\*\*), אבל אפשר לבצע את אותו החישוב גם באמצעות הפונקציהpow המקבלת זוג מספרים כפרמטרים ומחזירה לנו את התוצאה של המספר הראשון בחזקת המספר שני.

print(pow(10, 2))

ההוראה תדפיס 100

בנוסף, הפונקציה pow יכולה לקבל פרמטר שלישי אשר מגדיר מודולו על תוצאת החזקה.

print(pow(4, 3, 5))

ההוראה תדפיס: 4

הדוגמה הזאת מקבילה לחישוב 5 % (3\*\*4) (השארית של 64 חלקי 5 היא 4).

**פונקציות נוספות:**

**חישוב ערך מוחלט** (מרחק מ-0) ניתן לקבל באמצעות הפונקציה **abs**

print(abs(-123))

ההוראה תדפיס: 123

**הפונקציה round** עוזרת לנו לעגל מספרים עשרוניים

print(round(5.4321))

ההוראה תדפיס: 5.0 (התוצאה חזרה בפורמט עשרוני)

במידה ונרצה לעגל עד למקום מסוים לאחר הנקודה העשרונית, ניתן להוסיף לפונקציית **round** פרמטר המגדיר את מידת הדיוק שנרצה לקבל בתוצאה. כלומר, כמה ספרות יש להציג אחרי הנקודה העשרונית.

print(round(5.4321, 2))

ההוראה תדפיס: 5.43

הפונקציה **round** מבצעת עיגול משתנה, זאת אומרת שאת הספרות 0-4 היא תעגל למטה בעוד שאת הספרות 5-9 תעגל למעלה.

**כדי לחשב שורש ריבועי, לעומת זאת כבר נאלץ לייבא ספרייה נוספת**

import math

print(math.sqrt(100))

ההוראה תדפיס: 10.0

הספרייה math מכילה עשרות פונקציות לחישובים מתקדמים המתאימים לתחומי המתמטיקה, פיזיקה, סטטיסטיקה, הנדסה וכו'. בדוגמה האחרונה היינו צריכים לרשום קודם את שם הספרייה (math) כדי שנוכל לגשת לכל הפונקציות שהיא מכילה. צורת הקריאה הזאת לפונקציות מתוך ספריות ייעודיות נפוצה מאוד בפייתון ונראה אותה עוד הרבה בהמשך.

אם אנחנו רוצים לוודא עיגול בכיוון מסוים אנחנו יכולים להשתמש בפונקציות  **ceil תקרה** – תמיד תעגל למעלה ו**- floor רצפה** – תמיד תעגל למטה. הפונקציות האלה לא יכולות לקבל פרמטר נוסף לקביעת דיוק ולכן תמיד יעגלו למספר שלם. שתי הפונקציות האלה הן חלק מספריית  **math** שראינו קודם ולכן גם כדי להשתמש בהן נצטרך לייבא את הספרייה.

import math

print(math.floor(10.99999))

print(math.ceil(10.00001))

ההוראה תדפיס: 10

11