רשומה - Tuple

1. כללי

הגדרה - tuple משמש לאחסון פריטים מרובים במשתנה יחיד.

ההגדרה של **tuple** דומה ל list רק שההבדל הוא ש **tuple** כותבים בין סוגריים עגולים קטנים ולא בין סוגריים מרובעים.

דוגמה : myTuple = ("red", "blue", "green")

tuple הוא אחד מתוך 4 סוגי נתונים מוכללים ב- Python המשמשים לאחסון אוספי נתונים, 3 האחרים הם **list,**
set ו **dictionary** . כולם בעלי תכונות ושימוש שונים.

tuple הוא אוסף שיש לו סדר **והוא בלתי ניתן לשינוי.**

האינדקס של הפריט הראשון הוא [0] של השני [1] וכו'.

tuples אינם ניתנים לשינוי, כלומר אין באפשרותנו לשנות, להוסיף או להסיר פריטים לאחר יצירת ה tuple.

ל **tuple** יכולים להיות פריטים בעלי ערך זהה.

גם כאן כדי לקבל את טיפוס הנתון רושמים את הפונקציה **type( )** . מנקודת המבט של פייתון, tuples מוגדרים כאובייקטים עם סוג הנתונים 'tuple' :

דוגמה :

```
colors = ("red", "blue", "green", "blue")
print(myTuple)
print (type(myTuple))
```

ההדפסה שנקבל :

```
('red', 'blue', 'green', 'blue')
<class 'tuple'>
```

1.א אורך של tuple

כדי לדעת מהו אורך ה tuple (כמה פריטים יש) נשתמש בפונקציה len( )

דוגמה :

```
myTuple = ("red", "blue", "green")
print(len(myTuple))
```

ההדפסה שנקבל : 3

1.ב יצירת tuple עם פריט 1

כדי ליצור tuple עם פריט אחד בלבד יש להוסיף פסיק (,) אחרי הפריט אחרת פייתון לא תזהה שמדובר ב tuple ותזהה אותו כ str .

דוגמה : (יש לשים לב לפסיק אחרי הפריט הראשון).

```
myTuple = ("red",)
print(myTuple)
print(type(myTuple))
```

ההדפסה שנקבל :

```
('red',)
<class 'tuple'>
```

קיבלנו הדפסה ש myTuple הוא tuple .

דוגמה : אם לא נשים פסיק אחרי הפריט הראשון :

```
myTuple = ("red")
print(myTuple)
print(type(myTuple))
```

ההדפסה שנקבל :

```
red
<class 'str'>
```

קיבלנו הדפסה שהמחלקה היא מחרוזת ולא tuple .

1.ג הפריטים ב tuple

הפריטים ב tuple יכולים להיות מכל טיפוס נתונים.

דוגמה : ניצור 3 דוגמאות של tuples מהטיפוסים מחרוזת – str , שלם - int ובוליאני :

```
tuple1 = ("red", "blue", "green")
tuple2 = (10, -5, -17, 19, 35)
tuple3 = (True, False, False)
```

ב tuple יכולים להיות משתנים מטיפוסי נתונים שונים.

דוגמה ל tuple עם טיפוסי נתונים שונים :

```
tuple1 = ("red", -17, 35, "blue", True, 100, False)
```

1.ד הבנאי constructor

ניתן ליצור tuple בעזרת שימוש בבנאי tuple .

דוגמה : יש לשים לב לפעמיים סוגריים קטנים בשורה הראשונה !

```
myTuple = tuple (("red", "blue", "green"))
print(myTuple)
```

```
print(type(myTuple))
```

ההדפסה שנקבל :

```
('red', 'blue', 'green')
<class 'tuple'>
```

1.ה אוספים בפייתון (מערכים)

קיימים ארבעה טיפוסי נתונים של אוסף בשפת פייתון:

- **List** - אוסף מסודר שניתן לשינוי ומאפשר פריטים – חברים – כפולים.
- **Tuple** - אוסף מסודר ובלתי ניתן לשינוי. גם בו יש אפשרות לפריטים כפולים.
- **Set** - אוסף לא מסודר וללא אינדקסים. אין אפשרות לחברים כפולים.
- **Dictionary** - אוסף מסודר (ראה הערה בהמשך) שניתן לשינוי . אין חברים כפולים.

הערה : החל מפייתון 3.7 ה dictionaries מסודרים. בפייתון 3.6 וקודמים הם לא מסודרים.

כאשר בוחרים טיפוס של אוסף כדאי להבין את המאפיינים של הטיפוס . בחירת הטיפוס הנכון עבור מערכת של נתונים נותנת שמירה על המשמעות שלהם וזה מגדיל את היעילות והבטיחות.

2. גישה לפריטים ב tuple

2.א גישה בעזרת האינדקס שלהם

ניתן לגשת לפריט ב tuple בעזרת האינדקס שלו אותו נכתוב בתוך סוגריים מרובעים - [האינדקס].

לדוגמה : להדפיס את הפריט השני ב tuple הנקרא myTuple .

```
myTuple = ("red", "blue", "green")
print(myTuple[1])
```

ההדפסה שנקבל היא : blue (האיבר הראשון הוא עם אינדקס [0])

2.ב אינדקס שלילי

אינדקס שלילי פירושו להתחיל מהסוף. הפריט האחרון יהיה במיקום 1- הפריט לפני אחרון יהיה עם אינדקס 2- וכך הלאה.

דוגמה : נדפיס את האיבר האחרון של ה tuple .

```
myTuple = ("red", "blue", "green")
print(myTuple[-1])
```

ההדפסה שנקבל : green .

2.ג טווח של אינדקסים

ניתן לציין טווח של אינדקסים על ידי ציון איפה להתחיל ואיפה לסיים את הטווח.

כאשר מציינים טווח הערך המוחזר יהיה tuple חדש עם הפריטים שציינו.

דוגמה : הדפסה של הפריטים השני, השלישי והרביעי . הטווח שנרשום הוא [2:5] אבל זה לא כולל את החמישי !

```
myTuple = ("red", "blue", "green", "white", "blue")
print(myTuple[2:5])
```

ההדפסה שנקבל : ('green', 'white', 'blue')

אם בטווח לא נציין את האינדקס הראשון אז הכוונה היא החל מאינדקס 0 .

דוגמה :

```
myTuple = ("red", "blue", "green", "white", "blue")
print(myTuple[:5])
```

ההדפסה שנקבל : ('red', 'blue', 'green', 'white', 'blue')

אם בטווח לא נרשום את האינדקס הסופי אז נקבל הדפסה עד פריט האחרון.

דוגמה :

```
myTuple = ("red", "blue", "green", "white", "blue")
print(myTuple[2:])
```

ההדפסה שנקבל : ('green', 'white', 'blue')

2.ד טווח של אינדקסים שליליים

יש לציין אינדקסים שליליים אם רוצים להתחיל את החיפוש מסוף ה tuple .

דוגמה : הדפסת הפריטים מ 4- עד 1- (לא כולל 1-) :

```
myTuple = ("red", "blue", "green", "white", "blue")
print(myTuple[-4:-1])
```

ההדפסה שנקבל : ('blue', 'green', 'white')

2.ה בדיקה האם פריט נמצא ?

כדי לדעת האם פריט מסוים נמצא ב tuple נשתמש במילת המפתח **in** .

תרגיל : נבדוק האם הצבע green נמצא ב tuple :

```
myTuple = ("red", "blue", "green", "white", "blue")
if "green" in myTuple:
    print("yes, green is in the tuple")
```

ההדפסה שנקבל : yes, green is in the tuple

הערה : גם אם נרשום במקום "green" את המילה 'green' התוכנית תעבוד באופן זהה.

3. עדכון של tuple

אין אפשרות לשנות tuple , כלומר לא ניתן להוסיף או להסיר פריטים אחרי יצירת ה tuple , אבל ישנן כמה דרכים לעקיפת הבעיה.

3.א שינוי הערכים של tuple

יש אפשרות להמיר את ה tuple ל list , לשנות את ה list ולהמיר בחזרה את ה list ל tuple

דוגמה : נשנה את הצבע green ל yellow ב tuple ששמו myTuple .

```
myTuple = ("red", "blue", "green", "white", "blue")
myList=list(myTuple)          # הפיכה לרשימה
myList[2]="yellow"            #  [2] החלפת הפריט במיקום
myTuple=tuple(myList)         # tuple המרה של הרשימה ל
print(myTuple)
```

ההדפסה שנקבל : ('red', 'blue', 'yellow', 'white', 'blue')

3.ב הוספת פריט

היות ו tuple איננה ניתנת לשינוי אין לה מתודה כמו append() אבל יש דרכים אחרות להוספה של פריטים ל tuple .

3.ב.1 המרה לרשימה

בדיוק כמו הדרך לעקיפת הבעיה לשינוי tuple , יש אפשרות להמיר אותה לרשימה, להוסיף את הפריטים הרצויים ולהמיר אותם בחזרה ל tuple .

דוגמה: המרה של myTuple לרשימה list הוספה של פריט בסוף הרשימה והחזרת הרשימה ל tuple .

```
myTuple = ("red", "blue", "green")
myList=list(myTuple)          # הפיכה לרשימה
myList.append("yellow")                  [הוספת הפריט בסוף הרשימה #
myTuple=tuple(myList)# tuple המרה של הרשימה ל
print(myTuple)
```

ההדפסה שנקבל : ('red', 'blue', 'green', 'yellow')

2.ב.3 הוספה של tuple ל tuple

ניתן להוסיף tuple ל tuple כך שאם ברצוננו להוסיף פריט אחד (או כמה פריטים) יוצרים tuple נוסף עם הפריט או הפריטים שרוצים להוסיף ואז מוסיפים את tuple הנוסף ל tuple הרצוי.

דוגמה : הוספה של tuple חדש של 3 פריטים בסוף tuple רצוי.

```
myTuple = ("red", "blue", "green")
newTuple = ("white", "black", "brown" )        # יצירת tuple חדש
myTuple += newTuple              # הוספת ה tuple החדש לרצוי
print(myTuple)
```

ההדפסה שנקבל : ('red', 'blue', 'green', 'white', 'black', 'brown')

הערה : אם יוצרים tuple עם פריט אחד לא לשכוח לשים פסיק לאחר הפריט אחרת הפריט לא יזוהה כ tuple !!

4. הסרה של פריט מ tuple

לא ניתן להסיר פריט מ tuple אבל ניתן לעקוף את הבעיה כמו שעשינו בסעיפים הקודמים ששינינו או הוספנו פריט ל tuple .

דוגמה : הסרה של פריט על ידי הפיכת tuple ל list , הסרת הפריט ב list והמרה חזרה ל tuple . בדוגמה כאן נסיר את blue

.

```
myTuple = ("red", "blue", "green")
myList=list(myTuple)          # הפיכה לרשימה
myList.remove("blue")         # הסרת הפריט blue
myTuple=tuple(myList)         # המרה של הרשימה ל tuple
print(myTuple)
```

ההדפסה שנקבל : ('red', 'green')

4.1 מחיקה של tuple

ניתן למחוק את כל ה tuple בעזרת מילת המפתח **del** .

דוגמה :

```
myTuple = ("red", "blue", "green")
del myTuple
print(myTuple)
```

ההדפסה שנקבל :

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
NameError: name 'myTuple' is not defined
```

קיבלנו שגיאה בשורה 3 כי מחקנו את myTuple ולכן היא איננה מוגדרת ולא ניתן להדפיס אותה.

5. פרוק/ריווח של tuple - Unpack Tuples

כאשר יוצרים tuple אנחנו מקצים להם ערכים. הדבר נקרא "packing" (אריזה או צפיפות).

בפייתון ניתן לחלץ את הערכים בחזרה למשתנים. הפעולה נקראת "unpacking" (פירוק או ריווח בעברית).

דוגמה: הכנסת הערכים של myTuple למשתנים:

```
myTuple = ("red", "blue", "green")
(x, y, z) = myTuple       # העברה של הערך red למשתנה x הערך blue למשתנה y ושל הערך green למשתנה z
print(x)
print(y)
print(z)
```

ההדפסה שנקבל :

```
red
blue
green
```

כמות המשתנים צריכה להיות שווה לכמות הערכים שב tuple . אם הכמות לא שווה נשתמש בכוכבית * ההסבר בפסקה הבאה.

אם בשורה 2 היינו רושמים כמות גדולה יותר של משתנים כמו (x, y, z) = myTuple היינו מקבלים את ההדפסה :

ValueError: not enough values to unpack (expected 4, got 3)

שאומרת שאין מספיק ערכים לפירוק (ציפינו ל 4 ערכים וקיבלנו 3).

אם בשורה 2 היינו שמים כמות קטנה יותר של משתנים כמו (x, y) = myTuple היינו מקבלים את ההדפסה :

ValueError: too many values to unpack (expected 2)

שאומרת שיש ערכים רבים מידי לפירוק (ציפינו רק ל 2 ערכים ויש 3).

5.1 השימוש בכוכבית * Using Asterisk

אם מספר המשתנים קטן ממספר הערכים ניתן להוסיף * לשם המשתנה והערכים יוקצו למשתנה כרשימה - list :

דוגמה : נעביר ערכים ל x ו y ואת שאר הערכים כרשימה ל z

```
myTuple = ("red", "blue", "green", "black", "white")
(x, y, *z) = myTuple     # העברה של הערך red למשתנה x הערך blue למשתנה y ושאר הערכים למשתנה z
print(x)
print(y)
print(z)
```

ההדפסה שנקבל :

```
red
```

```
blue
['green', 'black', 'white']
```

אם נוסיף כוכבית לשם של משתנה לפני האחרון אז יוקצו ערכים למשתנה הזה עד שמספר הערכים שנותרו יתאים למספר המשתנים שנותרו.

דוגמה :

```
myTuple = ("red", "blue", "green", "black", "white")
(x, *y, z, w) = myTuple
print(x)
print(y)
print(z)
print (w)
```

ההדפסה שנקבל :

```
red
['blue', 'green']
black
white
```

הערך שהוקצע ל x הוא red . ל y הוקצתה רשימה של 2 ערכים blue ו green . הוקצו 2 ערכים כי אחרי y יש לנו עוד 2 משתנים והם קיבלו את 2 הערכים האחרונים של ה tuple .

6. לולאות ב tuple - Python - Loop Tuples

6.1 לולאת for

ניתן לבצע לולאה ב tuple בעזרת לולאת for .

דוגמה : מעבר בלולאה על הפריטים והדפסה שלהם.

```
myTuple = ("red", "blue", "green")
for x in myTuple :
   print(x)
```

ההדפסה שנקבל :

```
red
blue
green
```

נרחיב על לולאות for בפרק על לולאות for בפייתון.

6.2 לולאה בעזרת מספר האינדקס

ניתן לעבור בלולאה בין הפריטים של tuple על ידי התייחסות למספר האינדקס שלהם. לשם כך ניעזר בפונקציות range( ) ו len( )

דוגמה : נדפיס את הפריטים לפי האינדקס שלהם בעזרת הפונקציות range ( ) ו len( ) :

```
myTuple = ("red", "blue", "green")
for i in range (len(myTuple)) :
   print(myTuple[i])
```

ההדפסה שנקבל :

```
red
blue
green
```

6.3 שימוש בלולאת while

ניתן לעבור על הפריטים ב tuple בעזרת לולאת while . נשתמש בפונקציה len( ) כדי לקבוע את אורך הלולאה . נתחיל מ 0 ונעבור בלולאה בין הפריטים של ה tuple בעזרת התייחסות לאינדקס שלהם. יש לזכור להגדיל את האינדקס ב 1 בסיום כל איטרציה (לולאה , מעבר על סדרת פקודות).

דוגמה : הדפסת הפריטים בעזרת לולאת while

```
myTuple = ("red", "blue", "green")
j=0
while j < (len(myTuple)) :
   print(myTuple[j])
   j=j+1
```

ההדפסה שנקבל :

```
red
blue
green
```

נרחיב על לולאות while בפרק על לולאות while בפייתון.

7. צירוף tuples

7.1 צירוף של 2 tuples

כדי לצרף 2 tuples או יותר נשתמש **באופרטור** + .

דוגמה : צירוף של tuple בסיום tuple **:**

```
myTuple1 = ("red", "blue", "green")
myTuple2 = (1, 2, 3)
myTuple3 = ("white", "black")
```

```
myTuple4 = myTuple3 + myTuple2 + myTuple1
print(myTuple4)
```

ההדפסה שנקבל : ('white', 'black', 1, 2, 3, 'red', 'blue', 'green')

7.2 הכפלה של tuples

אם רוצים להכפיל את התוכן של tuple מספר פעמים נשתמש **באופרטור** * .

דוגמה : נכפיל את myTuple 3 פעמים :

```
myTuple1 = ("red", "blue", "green")
myTuple2 = myTuple1*3
print(myTuple2)
```

ההדפסה שנקבל : ('red', 'blue', 'green', 'red', 'blue', 'green', 'red', 'blue', 'green')

8. מתודות של tuple

לפייתון יש 2 מתודות מוכללות בשימוש עם tuples .

- מתודת ( )count המחזירה את כמות הפעמים שמופיע ערך רצוי ב tuple.
- מתודת ( )index המחזירה את האינדקס ב tuple של הפריט עם הערך אותו מחפשים .

8.1 המתודה ( )count

המתודה ( )count מחזירה את מספר הפעמים שערך שצוין מופיע ב tuple .

התחביר :

tuple.count(*value*)

value הוא הערך (הפריט) שאותו מחפשים.

דוגמה : נבדוק כמה פעמים מופיע הערך "red" ב tuple הבא :

```
myTuple = ('red', 'blue', 'green', 'red', 'green', 'red', 'blue')
print(myTuple.count("red"))
```

ההדפסה שנקבל : 3

8.2 המתודה ( )index

המתודה ( )index מוצאת את האינדקס של הפריט הראשון ב tuple של הערך שצוין.

המתודה מבצעת תעופה (חריג) אם הערך לא נמצא.

תחביר :

***tuple*.index(*value*)**

value הוא הערך (פריט) אותו מחפשים.

דוגמה : מציאת האינדקס של הערך "blue" :

```
myTuple = ('red', 'blue', 'green', 'red', 'green', 'red', 'blue')
print(myTuple.index("blue"))
```

ההדפסה שנקבל : 1

9. תרגול tuples

בתרגול נשתמש ב 2 tuples שעליהן נבצע את התרגול:

```
myTuple1 = ("red", "blue", "green")
myTuple2 = (1, -7,10)
```

9.1 להדפיס את הפריט הראשון ב tuple1 ואת הפריט השני ב tuple2

9.2 להדפיס את כמות הפריטים בכל tuple .

9.3 להשתמש באינדקס שלילי כדי להדפיס את הפריט האחרון ב myTuple1 ואת הפריט לפני האחרון ב myTuple2 .

9.4 להדפיס את הפריטים השני והשלישי ב myTuple1 ואת הראשון והשני ב myTuple2 .

9.5 לבדוק ולהדפיס האם "blue" נמצא ב myTuple1 והאם 5 נמצא ב myTuple2 .

9.6 להוסיף "black" ל myTuple1 ואת הערכים 100, 200, 1000 ל myTuple2 .

9.7 לרשום פקודה שתכפיל את myTuple1 בכמות הפריטים שיש ב myTuple2.

פתרונות

9.1

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
print(myTuple1[0])
print(myTuple2[1])
```

9.2

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
print("The length of myTuple1 is : ",len(myTuple1))
print("The length of myTuple2 is : ",len(myTuple2))
```

9.3

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
print(myTuple1[-1])
print(myTuple2[-2])
```

9.4

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
print(myTuple1[1:3])
print(myTuple2[:2])
```

9.5

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
if "blue" in myTuple1:
        print("Yes, blue is in mytuple1")
else :
        print("No, blue is not in mytuple1")
if 5 in myTuple1:
        print("Yes, 5 is in mytuple2")
else :
        print("No, 5 is not in mytuple2")
```

9.6

```
myTuple1 = ('red', 'blue', 'green')
myTuple2 = (1, -7, 10)
x=list(myTuple1)
x.append("black")
myTuple1 = tuple(x)

myTuple2+=(100,200,1000)
print(myTuple1)
print(myTuple2)
```

9.7

```
myTuple1 = ('red', 'blue', 'green' )
myTuple2 = (1, -7, 10)
myTuple1 *= len(myTuple2)
print(myTuple1)
```