python

COLLECTION IN PYTHON

אוספים בפייתון הם סוגי נתוני מכולה, כגון רשימות, קבוצות, טאפלים ומילונים. לכל אחד מהם מאפיינים שונים בהתאם לאופן ההכרזה והשימוש.

A list

רשימה מוכרזת בסוגריים מרובעים, היא ניתנת לשינוי mutable שומרת ערכים כפולים, וניתן לגשת לאיברים באמצעות אינדקסים.

A tuple

טאפלים מסודרים ואינם ניתנים לשינוי immutable למרות שניתן שיהיו בהם ערכים כפולים.

A set

קבוצה אינה מסודרת ומוכרזת בסוגריים מרובעים. אין לה אינדקסים ואין בה ערכים כפולים.

A dictionary

מילון מכיל זוגות של מפתחות וערכים והוא ניתן לשינוי. הוא מוכרז בסוגריים מרובעים.

הפונקציה type מאפשרת לנו להדפיס את סוג האוסף:
print(type(name)).

LISTS

רשימות משמשות לאחסון מספר פריטים במשתנה אחד. הן אחד מ-4 סוגי נתונים מובנים בפייתון המשמשים לאחסון אוספים של נתונים, כאשר שלושת הסוגים האחרים הם טאפלים, קבוצות ומילונים, שלכל אחד מהם תכונות ושימושים שונים.

Lists are created using square brackets []:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

[]
Ordered
Changeable
Allow duplicates

הפריטים ברשימה מסודרים, ניתנים לשינוי ומאפשרים ערכים כפולים.
הפריטים ברשימה מאונדקסים, הפריט הראשון מקבל אינדקס [0], הפריט השני מקבל אינדקס [1] וכן הלאה.

לפריטים ברשימה יש סדר מוגדר, והסדר הזה לא ישתנה.
כאשר מוסיפים פריטים חדשים לרשימה, הם יתווספו בסוף הרשימה.
הרשימה ניתנת לשינוי, כלומר ניתן לשנות, להוסיף ולהסיר פריטים מהרשימה לאחר שנוצרה.
מכיוון שהרשימות מאונדקסות, הן יכולות לכלול פריטים עם אותו ערך.

LISTS -Access Items

הפריטים ברשימה מאונדקסים, וניתן לגשת אליהם על ידי התייחסות למספר האינדקס:
חשוב לציין שהפריט הראשון ברשימה מקבל את האינדקס 0.

Print the second item of the list:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

אינדקס שלילי אומר להתחיל מהסוף.
-1 מתייחס לפריט האחרון, -2 מתייחס לפריט הלפני אחרון, וכך הלאה.

Print the last item of the list:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

LISTS -Access Items

Range of Indexes

ניתן לציין טווח של אינדקסים על ידי קביעת נקודת ההתחלה ונקודת הסיום של הטווח.
כאשר מציינים טווח, הערך המוחזר יהיה רשימה חדשה עם הפריטים שהוגדרו.

Return the third, fourth, and fifth item:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

הערה: החיפוש יתחיל באינדקס 2 (כולל) ויסתיים באינדקס 5 (לא כולל).

NOT including, "kiwi":

```
print(thislist[:4])
```

LISTS -Change List Items

Change Item Value

כדי לשנות את הערך של פריט מסוים, יש להתייחס למספר האינדקס:

Change the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

Change a Range of Item Values

כדי לשנות את הערכים של פריטים בטווח מסוים, יש להגדיר רשימה עם הערכים החדשים ולהתייחס לטווח של מספרי האינדקס שבו רוצים להכניס את הערכים החדשים.

Change the values "banana" and "cherry" with the values "blackcurrant" and "watermelon":

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

LISTS -Change List Items

אם תכניס יותר פריטים ממה שאתה מחליף, הפריטים החדשים יתווספו במיקום שציינת, ושאר הפריטים יזוזו בהתאם:

Change the second value by replacing it with two new values:

```
thislist = ["apple", "banana", "cherry"]
thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

הערה: אורך הרשימה ישתנה כאשר מספר הפריטים שהוכנסו לא תואם למספר הפריטים שהוחלפו.

אם תכניס פחות פריטים ממה שאתה מחליף, הפריטים החדשים יוכנסו במקום שציינת, ושאר הפריטים יזוזו בהתאם.

Change the second and third value by replacing it with one value:

```
thislist = ["apple", "banana", "cherry"]
thislist[1:3] = ["watermelon"]
print(thislist)
```

LISTS -Change List Items

Insert Items

כדי להכניס פריט חדש לרשימה מבלי להחליף אף אחד מהערכים הקיימים, ניתן להשתמש בשיטת .insert()
שיטת insert() מכניסה פריט במיקום האינדקס שצוין:

Insert "watermelon" as the third item:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

Add Items

כדי להוסיף פריט לסוף הרשימה, יש להשתמש בשיטת append()

Using the append() method to append an item:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

LISTS -Extend List

Extend List

כדי להוסיף אלמנטים מרשימה אחרת לרשימה הנוכחית, יש להשתמש בשיטת extend()

Add the elements of tropical to thislist:

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

LISTS -Remove Items

Remove Specified Item

שיטת remove() מסירה את הפריט שצוין.

Remove "banana":

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Remove Specified Index

שיטת pop() מסירה את האינדקס שצוין.

Remove the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

אם לא צוין אינדקס, שיטת pop() תסיר את הפריט האחרון.

LISTS -Remove Items

מילת המפתח del גם מסירה את האינדקס שצוין:

Remove the first item:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

מילת המפתח del יכולה גם למחוק את הרשימה לחלוטין.

Delete the entire list:

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Clear the List

שיטת clear() מרוקנת את הרשימה. הרשימה נשארת, אך אין בה תוכן.

```
thislist.clear()
```

TUPLES

טאפלים משמשים לאחסון מספר פריטים במשתנה אחד.
הטאפל הוא אחד מארבעת סוגי הנתונים המובנים בפייתון לאחסון אוספי נתונים.
טאפל הוא אוסף שמסודר ואינו ניתן לשינוי.
טאפלים נכתבים עם סוגריים עגולים ().

Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

()
Ordered
Unchangeable
Allow duplicates

הפריטים בטאפל מסודרים, לא ניתנים לשינוי, ומאפשרים ערכים כפולים.
הפריטים בטאפל מאונדקסים, הפריט הראשון מקבל אינדקס [0], הפריט השני מקבל אינדקס [1], וכן הלאה.

כשאומרים שטאפלים מסודרים, הכוונה היא שלפריטים יש סדר מוגדר, וסדר זה לא ישתנה.
טאפלים אינם ניתנים לשינוי, כלומר לא ניתן לשנות, להוסיף או להסיר פריטים לאחר יצירת הטאפל.

TUPLE ITEMS

כדי לקבוע כמה פריטים יש בטאפל, יש להשתמש בפונקציה :len()

Print the number of items in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Create Tuple With One Item

כדי ליצור טאפל עם פריט אחד בלבד, יש להוסיף פסיק אחרי הפריט, אחרת פייתון לא תזהה אותו כטאפל.

One item tuple, remember the comma:

```
thistuple = ("apple",)
print(type(thistuple))
```

TUPLE ITEMS

הפריטים בטאפל יכולים להיות מכל סוג נתונים:

```
String, int and boolean data types:
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)
```

טאפל יכול להכיל סוגי נתונים שונים:

```
A tuple with strings, integers and boolean values:
tuple1 = ("abc", 34, True, 40, "male")
```

ניתן לחבר שני טאפלים יחד.

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

Unpacking a Tuple

כאשר אנו יוצרים טאפל, בדרך כלל אנו מקצים לו ערכים. זה נקרא "אריזת" טאפל:

Packing a tuple:

```
fruits = ("apple", "banana", "cherry")
```

אבל, בפייתון, ניתן גם לחלץ את הערכים בחזרה למשתנים. זה נקרא "פריקת" טאפל.

Unpacking a tuple:

```
fruits = ("apple", "banana", "cherry")
(green, yellow, *red) = fruits
print(green)
print(yellow)
print(red)
```

Using Asterisk*

אם מספר המשתנים קטן ממספר הערכים, ניתן להוסיף * לשם המשתנה, והערכים יוקצו למשתנה כרשימה.

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
```

Dictionaries

מילונים משמשים לאחסון ערכי נתונים בזוגות של מפתח:ערך.

מילון הוא אוסף שמסודר, ניתן לשינוי, ואינו מאפשר ערכים כפולים.

מילונים נכתבים עם סוגריים מסולסלים {} וכוללים מפתחות וערכים.

Create and print a dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

{}
Ordered
Changeable
Not allow duplicates

The dict() Constructor

אפשר גם להשתמש ב dict() כדי ליצור מילון.

Using the dict() method to make a dictionary:

```
thisdict = dict(name = "John", age = 36, country = "Norway")
print(thisdict)
```

Dictionaries - Adding Items

הוספת פריט למילון מתבצעת על ידי שימוש במפתח אינדקס חדש והקצאת ערך אליו.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

thisdict["color"] = "red"
print(thisdict)
```

Dictionaries - Update Dictionary

שיטת ()update תעדכן את המילון עם הפריטים מהארגומנט שניתן. אם הפריט לא קיים, הוא יתווסף.

הארגומנט חייב להיות מילון או אובייקט עם זוגות של מפתח:ערך.

Add a color item to the dictionary by using the update() method:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
```

Dictionaries - Change Values

ניתן לשנות את הערך של פריט מסוים על ידי התייחסות לשם המפתח שלו.

Change the "year" to 2018:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

Dictionaries - Accessing Items

ניתן לגשת לפריטים במילון על ידי התייחסות לשם המפתח שלהם בתוך סוגריים מרובעים:

Get the value of the "model" key:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
print(x)
```

ישנה גם שיטה בשם get() שתיתן את אותה תוצאה.

Get the value of the "model" key:

```
x = thisdict.get("model")
print(x)
```

Dictionaries – Keys and Values

Get Keys

שיטת keys() תחזיר רשימה של כל המפתחות במילון.

Get a list of the keys:

```
x = thisdict.keys()
print(x)
```

Get Items

שיטת items() תחזיר כל פריט במילון כטאפלים ברשימה.

Get a list of the key:value pairs

```
x = thisdict.items()
print(x)
```

Get Values

שיטת values() תחזיר רשימה של כל הערכים במילון.

Get a list of the values:

```
x = thisdict.values()
print(x)
```

Dictionaries - Removing Items

ישנן מספר שיטות להסרת פריטים ממילון:

שיטת pop() מסירה את הפריט עם שם המפתח שצוין.

```
thisdict.pop("model")
```

שיטת popitem() מסירה את הפריט האחרון שהוכנס.

```
thisdict.popitem()
```

מילת המפתח del מסירה את הפריט עם שם המפתח שצוין או מוחקת את המילון לחלוטין.

```
del thisdict["model"]

del thisdict
```

שיטת clear() מרוקנת את המילון.

```
thisdict.clear()
```

Sets

קבוצות Sets משמשות לאחסון מספר פריטים במשתנה אחד.

קבוצה היא אוסף שאינו מסודר, אינו ניתן לשינוי*, ואינו מאונדקס.
*הערה: הפריטים בקבוצה אינם ניתנים לשינוי, אך ניתן להסיר פריטים ולהוסיף פריטים חדשים.

קבוצות נכתבות עם סוגריים מסולסלים.
הפריטים בקבוצה יכולים להיות מכל סוג נתונים, וקבוצה יכולה להכיל סוגי נתונים שונים.

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

{}
Unordered
Unchangeable
Not allow duplicates

Sets - Add Set Items

ברגע שקבוצה נוצרת, לא ניתן לשנות את הפריטים בה, אך ניתן להוסיף פריטים חדשים.

To add one item to a set use the add() method.

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

To add items from another set into the current set, use the update() method.

האובייקט בשיטת ()update לא חייב להיות קבוצה, הוא יכול להיות כל אובייקט ניתן לחזרה (טאפלים, רשימות, מילונים וכו').

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)
```

Sets – Remove Items

כדי להסיר פריט מקבוצה, ניתן להשתמש בשיטות ()remove או ()discard
קבוצות אינן מסודרות, לכן בעת שימוש בשיטת ,()pop אינך יודע איזה פריט יוסר.

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

<u>הערה:</u> אם הפריט שברצונך להסיר לא קיים, שיטת ()remove תזרוק שגיאה.

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

שיטת ()clear מרוקנת את הקבוצה, ומילת המפתח del תמחק את הקבוצה לחלוטין.

format() Method

שיטת מעצבת את הערך/הערכים שצוינו ומכניסה אותם למקומות השמורים במחרוזת.
format()
המקום השמור מוגדר באמצעות סוגריים מסולסלים: {}.
שיטת format() מחזירה את המחרוזת המעוצבת.

ניתן לזהות את המקומות השמורים באמצעות אינדקסים עם שמות {price } אינדקסים ממוספרים {0}, או אפילו מקומות שמורים ריקים {}.

Using different placeholder values:

```
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)

age = 44
name = "bob"
txt4 = f"Hi my name is {name} and I am {age} years old"
```

Exercises

Exercises

Exercise 1 :

Create a program that asks the user for his name, his age and his occupation.
Print a resume of informations you collected using f format.

Exercise 2 :

Multiply the fruits tuple by 2 and display it
fruits = ("apple", "banana", "cherry")

Exercise 3 :

Create a program with a predefined list of people. Ask the user for his name, add it to the end of the list and print the updated list.

Exercises

Exercise 4 :

Create a program with a predefined dictionary for a person. Include the following information : name, gender, age, address and phone.
Ask the user what information he wants to know about the person (example : name)
Print the value associated to that key
Display a message in case the key is not found

Exercise 5 : Extra

Create a program that asks the user for his birthday in the format “DD-MM-YYYY”.
Then print :
“You were born in [month]” (In letters)


Q&A