פרק 7 פונקציות

תוכן העניינים :

פרק 7 : פונקציות

בפרק 7 של תוכנית הלימודים להנדסאים במקצוע שפת פייתון מופיע:

פרק 7 : פונקציות

יעדים

היכרות עם עולם הפונקציות שבו מצרפים יחד מספר הוראות, במטרה לבצע אלגוריתם מוגדר.

תכנים

1. תחום קיום של משתנים (Scope)
2. משמעות הפקודה "def"
3. כיצד ניגשים למשתנה גלובלי ?
4. קינון (Nested) פונקציות
5. סוגי משתנים וערכים מוחזרים
6. מתי יש להשתמש בכל אחד מהם
7. קלט מהמקלדת
8. עבודה עם עיבוד מידע : map , reduce , lambda , filter , zip
9. עבודה עם מחרוזות מפורמטות
10. פונקציית מחוללת אקראיות – random
11. העברת פרמטרים לפונקציה, תחביר הפונקציות :

- func(name)
- func(name = value)
- func(*name)
- func(**name)

פרק 7 : פונקציות

פונקציה היא בלוק של קוד (בלוק של פקודות) הפועל רק כאשר קוראים לו . הפונקציה יכולה לקבל נתונים, לעבד אותם ולהחזיר את תוצאות העיבוד (נקרא לזה "ערך מוחזר") לשורה שקראה לפונקציה.

בעבודה עם פונקציה יש את השלבים הבאים :

* יצירה/הגדרה של הפונקציה – כתיבת בלוק הקוד של הפונקציה כולל הכנה לקבלת נתונים הנשלחים אליה ועיבוד הנתונים.

הנתונים המתקבלים בפונקציה נקראים **פרמטרים**.

* קריאה לפונקציה משורה מסוימת בתוכנית עם אפשרות לשלוח לה נתונים הנקראים **ארגומנטים**.

* בשורה הקוראת יש לעשות "הכנה" לקבלת ערך חוזר מהפונקציה שהוא תוצאה של עיבוד הנתונים שנשלח אל הפונקציה.

בהמשך נסביר את השימוש בפונקציות והייתרון של המודולריות והאחזקה הקלה יותר של תוכניות עם פונקציות.

7.1 יצירה/הגדרה של פונקציה

פונקציה בפייתון מוגדרת בעזרת מילת המפתח def

תחביר:

def שם הפונקציה (פרמטרים שהפונקציה מקבלת (אופציונלי)) :

משפטי הפונקציה

return ((ערך מוחזר (אופציונלי))

def – מילה שמורה המתארת הגדרה של פונקציה.

שם הפונקציה – שם כלשהו אבל בדומה לחוקיות כמו למשתנה. כדאי לתת שם המסביר מה הפונקציה עושה.

פרמטרים שהפונקציה מקבלת – אופציה. אם נשלחים לפונקציה ארגומנטים הם נכנסים לפרמטרים שרושמים בין הסוגריים .

משפטי הפונקציה – המשפטים של הפונקציה כאשר הם באיידנטציה (הזחה) יחסית לשורה הראשונה של ההגדרה.

Ret – אופצונלי . ערך חוזר אל השורה הקוראת.

דוגמה 1 : הגדרת פונקציה שלא מקבלת ערכים ולא מחזירה ערכים.

```
def func1( )
    print("a print in func1")
```

הקריאה לפונקציה : השורה הקוראת לפונקציה יכולה להיות בתכנית ה main או בכל פונקציה שהיא והיא תהיה עם שם הפונקציה ולאחריה סוגריים קטנים. בדוגמה כאן : func1( ) .

7.2 ארגומנטים

ניתן להעביר נתונים לפונקציה והם נקראים ארגומנטים. הארגומנטים מצוינים לאחר שם הפונקציה, בתוך הסוגריים.

יש אפשרות להוסיף ארגומנטים רבים ככל שנרצה ונפריד ביניהם באמצעות פסיק.

ארגומנטים נקראים לפעמים בקיצור בשם args .

דוגמה : נגדיר פונקציה עם 2 ארגומנט אחד הוא fname – שם פרטי והשני הוא lname – שם המשפחה. כאשר קוראים לפונקציה נעביר לה את השם הפרטי ושם המשפחה . הפונקציה תדפיס את השם המלא.

```
def func2(fname,lname) :                    # הגדרה של פונקציה המקבלת 2 ערכים ולא מחזירה ערך
    print (fname + ' ' + lname)
```

השורה הקוראת :

```
func2("Avraham","Avinoo")
```

ההדפסה שנקבל : Avraham Avinoo

7.3 ארגומנטים ופרמטרים

המושגים ארגומנטים ופרמטרים משמשים עבור המידע שמועבר אל הפונקציה. ההבדל ביניהם הוא :

הפרמטרים הם המשתנים הרשומים בסוגריים בהגדרת הפונקציה.

הארגומנטים הם הערכים הנשלחים מהשורה הקוראת אל הפונקציה.

בדוגמה הקודמת fname ו lname הם הפרמטרים של הפונקציה. "Avraham" ו "Avinoo" הם הארגומנטים הנשלחים אל הפונקציה.

כמות הארגומנטים הנשלחים אל הפונקציה חייבת להיות שווה לכמות הפרמטרים שהפונקציה מקבלת !! אם כמות הארגומנטים איננה שווה לכמות הפרמטרים נקבל הודעת שגיאה.

דוגמה : הודעת שגיאה שנקבל אם שולחים ארגומנט אחד אבל בפונקציה הגדרנו 2 פרמטרים :

```
def func2(fname,lname) :
    print (fname + ' ' + lname)
```

הקריאה לפונקציה :

```
func2("Avraham")
```

הודעת השגיאה שנקבל :

```
Traceback (most recent call last):
  File "./prog.py", line 4, in <module>
TypeError: func2() missing 1 required positional argument: 'lname'
```

השגיאה היא שפונקציה 2 חסרה ארגומנט 1 הנדרש במיקום lname .

7.4 הגדרת פונקציה כאשר לא יודעים כמה ארגומנטים מועברים אל הפונקציה (*args)

אם אנחנו לא יודעים כמה ארגומנטים נעביר אל הפונקציה , ניתן להוסיף כוכבית (*) לפני שם הפרמטר בהגדרת הפונקציה.

בדרך זו הפונקציה תקבל tuple של ארגומנטים, ויכולה לגשת לפריטים בהתאמה:

דוגמה : קליטת כמות ארגומנטים שלא ידועה והדפסת האיבר האחרון .

```
def func3(*brothers):
    print("The youngest child is " + brothers[-1])
```

הקריאה לפונקציה :

```
func3("Reuven", "Shimon", "Levi", "Josef", "Benjamin")
```

ההדפסה שקיבלנו :

```
The youngest child is Benjamin
```

7.5 ארגומנטים של מילות מפתח

ניתן לשלוח ארגומנטים עם התחביר *key = value* syntax . בצורה זו סדר הארגומנטים לא משנה.

דוגמה :

```
def func4(child3, child2, child1,child5, child4):  # סדר הפרמטרים הוא לא סדר הארגומנטים בשורה הקוראת
  print("The youngest child is " + child5)
```

השורה הקוראת :

```
func4(child1 = "Reuven", child2 = "Shimon", child3 = "Levi", child4="Josef", child5="Benjamin")
```

ההדפסה שנקבל : The youngest child is Benjamin

הערה : צירוף הארגומנטים של מילות המפתח נרשם בקצרה kwargs .

7.6 ארגומנטים שרירותיים של מילות מפתח, **kwargs

אם אנחנו לא יודעים כמה ארגומנטים של מילות מפתח יועברו לפונקציה ניתן להוסיף שתי כוכביות ** לפני שם הפרמטר בהגדרת הפונקציה. בדרך זו הפונקציה תקבל מילון - dictionary - של ארגומנטים, ויכולה לגשת לפריטים בהתאמה . ארגומנטים שרירותיים של מילות מפתח נקראות לפעמים בקיצור **kwargs** .

דוגמה : הוספת 2 כוכביות ** לפרמטר כאשר מספר הארגומנטים של מילות המפתח אינו ידוע.

```
def func5(**kid):
  print("His last name is " + kid["lname"])
```

הקריאה לפונקציה:

```
func5(fname = "Avraham", lname = "Avinoo")
```

ההדפסה : His last name is Avinoo

7.7 ברירת מחדל של ערך פרמטר

אם נקרא לפונקציה ולא נשלח לו ארגומנט אז הפונקציה תשתמש בערך ברירת המחדל.

דוגמה : כיצד להשתמש בערך פרמטר המהווה ברירת מחדל :

```
def func6(brother = "Levi"):                אם הפונקציה לא תקבל ארגומנט אז יירשם "Levi" #
  print("My brother is " + brother)
```

הקריאה לפונקציה :

```
func6("Reuven")
func6("Shimon")
func6()
func6("Josef")
```

ההדפסה שנקבל :

```
My brother is Reuven
My brother is Shimon
My brother is Levi
My brother is Josef
```

7.8 העברה של list כארגומנט

ניתן לשלוח כארגומנט לפונקציה את כל סוגי הנתונים (מחרוזת, מספר, רשימה, מילון וכו ') והארגומנט יטופל כסוג נתונים זהה בתוך הפונקציה. למשל, אם נשלח ארגומנט כמו רשימה הוא יהיה רשימה ויטופל כרשימה בפונקציה.

דוגמה :

```
def func7(food):
  for x in food:
    print(x)
fruits = ["apple", "cherry", "lemon", "banana"]   # הגדרה של רשימה
```

הקריאה לפונקציה :

```
func7(fruits)
```

ההדפסה שנקבל :

```
apple
cherry
lemon
banana
```

7.9 ערכים מוחזרים

כדי להחזיר ערך מפונקציה משתמשים בהצהרה return .

דוגמה : רשום שורה שתקרא לפונקציה המחזירה את הריבוע של המספר ששולחים אליה.

```
def func8(x):
  return x * x
```

השורות הקוראות לפונקציה

```
print(func8(3))
print(func8(5))
print(func8(9))
```

ההדפסה שנקבל :

```
9
25
81
```

7.10 ההצהרה pass

הגדרות פונקציה אינן יכולות להיות ריקות אבל אם מסיבה כלשהי אנחנו מגדירים פונקציה ללא תוכן יש לרשום הכנס את משפט ה pass כדי להימנע מקבלת שגיאה.

דוגמה :

```
def func9( ):
  pass
```

אם היינו מגדירים פונקציה ללא משפטים בפונקציה נקבל הודעת שגיאה.

דוגמה : קבלת הודעת שגיאה בפונקציה ללא משפטים וללא pass .

```
def func10():
```

הודעות השגיאה שנקבל :

```
Traceback (most recent call last):
  File "/usr/lib/python3.8/py_compile.py", line 144, in compile
    code = loader.source_to_code(source_bytes, dfile or file,
  File "<frozen importlib._bootstrap_external>", line 846, in source_to_code
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "./prog.py", line 2

    ^
SyntaxError: unexpected EOF while parsing

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/usr/lib/python3.8/py_compile.py", line 150, in compile
    raise py_exc
py_compile.PyCompileError:   File "./prog.py", line 2

    ^
SyntaxError: unexpected EOF while parsing
```

7.11 רקורסיה

רקורסיה היא מושג מתמטי ותכנותי נפוצים.

רקורסיה קורית כאשר מגדירים פונקציה ובאחד ממשפטי הפונקציה קוראים לפונקציה עצמה .

היתרון של רקורסיה הוא שניתן לעבור בלולאה על נתונים כדי להגיע לתוצאה.

יש לעבוד בזהירות רבה מאוד עם רקורסיה כי טעות קטנה יכולה לגרום לכניסה ללולאה אין סופית או שהמחשב ישתמש בכמות זיכרון עודפת וכוח מעבד מיותר. עם זאת, כאשר נכתוב נכון רקורסיה נוכל לקבל גישה יעילה מאוד מתמטית אלגנטית לתכנות.

בדוגמה שנראה מיד נגדיר פונקציה בשם my_recursion( ) . באחד ממשפטי הפונקציה נקרא ל my_recursion( ) פעם נוספת. נשתמש במשתנה בשם k כנתון אשר מורידים ממנו (1-) בכל לולאה של רקורסיה. הרקורסיה מסתיימת כאשר k מגיע ל 0 .

דוגמה 1: נרשום רקורסיה המחברת את המספרים מ 1 עד 5 . הרקורסיה תתבצע 5 פעמים כאשר בכל פעם נחבר את הערך ב k עם הערך של חיבור המספרים הקודמים שהיו ב k. נכנסים לרקורסיה כאשר k=5 ויוצאים ממנה רק כאשר k לא גדול מ 0 (כלומר k=0).

```
def my_recursion(k):      #  k הגדרה של פונקציה המקבלת את הערך 5 למשתנה
  if(k > 0):              #  k>0 כניסה רק כאשר
    result = k + my_recursion(k - 1)  # k-1 התוצאה result שווה לערך של k פלוס הערך החוזר מהפונקציה my_recursion ושולחים לה את
    print(result)
  else:
    result = 0
  return result
```

השורות שבו מדפיסים את תוצאות הרקורסיה

```
print("\n\nRecursion Example Results")
my_recursion(5)           #  קריאה לפונקציה ושליחת הערך 5 אליה
```

ההדפסות שנקבל :

```
Recursion Example Results
1
3
6
10
15
```

דוגמה 2 : מציאת העצרת factorial של מספר :

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
print(factorial(5))
```

ההדפסה שנקבל : 120

7.12 תרגילים

1 . כתוב תוכנית המחשבת שכר נטו של 10 עובדים. התוכנית תהיה מורכבת מתוכנית ראשית הקולטת שכר של עובד וקוראת לפונקציה המחזירה את שכר הנטו על פי מס פרוגרסיבי לפי המפתח הבא :

עד 2000 ₪ - אין הורדת מס. עד 4000 ₪ - 10% מס. עד 5000 ₪ – 20% מס. עד 6000 ₪ - 30% מס, עד 7000 ₪ - 40% . מעל זה 50% מס .

2. כתוב תוכנית הקולטת מספר שלם וקוראת לפונקציה המדפיסה את כל המספרים הראשוניים עד למספר הנקלט. מספר ראשוני הוא מספר המתחלק רק ב 1 ובעצמו (לדוגמא : 1,2,3,5,7,11)

3. כתוב תוכנית הקולטת 3 מספרים מהמשתמש. התוכנית תקרא לפונקציה הבודקת האם 3 המספרים שווים ותחזיר ערך 0 אם השלושה לא שווים. התוכנית תוציא הדפסה מתאימה.

4. כתוב תוכנית שתקלוט 3 מספרים ותקרא לפונקציה שתבדוק מי המספר הגדול ומי הקטן.
הפונקציה תחזיר את הערכים הבאים :
123 אם c>b>a או 132 אם b>c>a או 231 אם a>c>b או 312 אם b>a>c או 213 אם c>a>b או 321 אם a>b>c .

5. רשום תוכנית המבצעת את הדברים הבאים: א. מבקשת מהמשתמש להכניס שני מספרים שלמים. ב. מוציאה הדפסה של מסך תפריט הכולל חיבור, חיסור, כפל וחילוק. ג. עבור כל בחירה שהמשתמש יקיש יש לבצע את הפעולה המתבקשת ולהוציא הדפסה של התוצאה. התוכנית תכיל את הפונקציות הבאות: 1. פונקציית קלט 2 מספרים. 2. פונקציית הדפסת תפריט. 3. פונקציית קליטת מקש לבחירת הפעולה הרצויה. 4. פונקציית חיבור, פונקציית חיסור, פונקציית כפל ופונקציית חילוק.

6. כתוב תוכנית המקבלת כקלט מטריצה של 10*10 . התוכנית תקרא לפונקציה אשר תאתחל את המערך במספרים רנדומליים מ 0 עד 10. לאחר מכן תקרא לפונקציה שתציג את מסך התפריט הבא :

1. איפוס שורה במטריצה
2. חישוב ממוצע של שורה במטריצה // המשתמש יכניס מאיזו שורה עד איזו שורה
3. חישוב ממוצע של שורות במטריצה
4. איפוס עמודה במטריצה
5. חישוב ממוצע של עמודה במטריצה
6. חישוב ממוצע של עמודות
7. הדפסת המטריצה

יש לחלק את התוכנית לפונקציות המבצעות את הנאמר בתפריט

הערה : העזר בפונקציה random (11) השייכת לספריה stdlib.h והמכניסה מספרים אקראיים בין 0 ל 10 (המספר בסוגריים אומר עד איזה מספר אקראי ייכנס)

7. כתוב פונקציה המקבלת את מספר היום בשנה ומחזירה את החודש והיום המתאימים. הנח שכמות הימים בחודש היא : 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 החל מינואר ועד דצמבר בהתאמה.