גבהים
תכנית הסייבר הלאומית

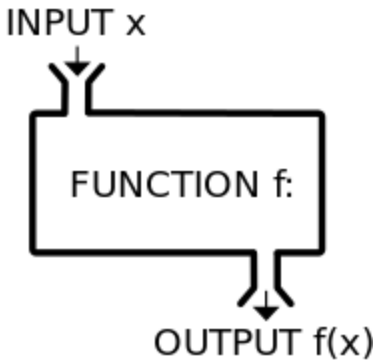
פייתון

שיעור 5: פונקציות

ברק גונן

פונקציה

- פונקציה היא קטע קוד שקוראים לו בשם
- אפשר להעביר לקטע הקוד מידע, שעליו יבוצעו פעולות
- אפשר גם לקבל חזרה ערכים מהפונקציה
- קריאה לפונקציה בפייתון: שם הפונקציה וסוגריים עגולים
  - בלי פרמטרים: my_func()
  - עם פרמטר אחד: my_func(param)



דוגמה לפונקציה ללא פרמטרים

```
def hello():
    """ Print Hello """
    print 'Hello'
```

- דגשים:
  - שם בעל משמעות
  - תיעוד בתחילת הפונקציה
  - אם נכתוב help(hello) תודפס מחרוזת התיעוד

```
Help on function hello in module __main__:

hello()
    Print Hello
```

דוגמאות לפונקציות עם פרמטרים

```
def print_message(message):
    """ Print a message
    Args:
        message - a string
    """
    print message


def print_messages(msg1, msg2):
    """ Print two messages
    Args:
        msg1 - a string
        msg2 - a string
    """
    print msg1, msg2
```

...כמובן שאין הגיון לכתוב
פונקציה שרק קוראת לפונקציה
print- הדוגמאות הללו הן רק
לצורך המחשת נושא הפרמטרים

קריאה לפונקציות בלי / עם פרמטרים

```
def main():
    message = 'I am a function which receives one parameter'
    print_message(message)
    mes = 'I am a function'
    sage = 'which receives two parameters'
    print_messages(mes, sage)
```

- תוצאת הקריאה ל-()main:

```
I am a function which receives one parameter
I am a function which receives two parameters
```

Return

- פונקציה מחזירה ערך, או ערכים, ע"י return
- לדוגמה:

```
def seven():
    x = 7
    return x


def main():
    a = seven()
    print a
```

Return- המשך

- אפשר להחזיר יותר מערך אחד:

```
def seven_eleven():
    x = 7
    y = 11
    return x, y


def main():
    var1, var2 = seven_eleven()
    print var1, var2
```

None

- הגדרנו את הפונקציה הבאה:

```
def stam():
    return None
```

- מה יהיה ערכו של k אם נכתוב k=stam()?

- המילה None היא מילה שמורה, ערך ריק
- משתנה יכול להיות שווה None.

```
k==None
True
```

None - המשך

- יש מצבים נוספים בהם יוחזר ערך None:
  - לפונקציה אין return כלל
  - לפונקציה יש return בלי ערך / משתנה
  - לפונקציה יש return None

Scope של משתנים

- משתנה גלובלי- מוגדר מחוץ לפונקציות, מוכר לכולם
- משתנה לוקלי – מוגדר בתוך פונקציה, מוכר רק לפונקציה

THINK Global.
ACT Local.

Scope של משתנים- משתנה גלובלי

- מה ידפיס הקוד הבא?
- המשתנה name הוא גלובלי ומוכר לפונקציה

```
name = 'Shooki'


def speak():
    word = 'hi'
    print word, name


def main():
    speak()
```

Scope של משתנים – משתנה לוקלי

```
def speak():
    word = 'hi'
    print word


def main():
    speak()
    print word
```

- מה ידפיס הקוד הבא?
- ה-print שב-main יעלה שגיאה. המשתנה word אינו מוכר מחוץ לפונקציה.

Scope של משתנים- קדימויות

```
word = 'bye'


def speak():
    word = 'hi'
    print word


def main():
    speak()
```

- מה ידפיס הקוד הבא?
- בתוך הפונקציה,
  המשתנה הלוקלי מקבל
  "קדימות"

Scope של משתנים- קדימות (המשך)

```
word = 'bye'


def speak():
    word = 'hi'
    print word


def main():
    speak()
    print word
```

- מה ידפיס הקוד הבא?
- מחוץ לפונקציה, רק המשתנה הגלובלי מוכר

Scope של משתנים

```
word = 'love'


def speak():
    word += ' you'
    print word


def main():
    speak()
    print word
```

- מה ידפיס הקוד הבא?
- שגיאת הרצה, word אינו מוכר בתוך הפונקציה
- מדוע? כאשר משנים את ערכו של משתנה בפונקציה, פייתון מניח שיש לו עותק מקומי

Scope של משתנים

- יש שתי שיטות לתקן את השגיאה:

העברת המשתנה כפרמטר:

```
word = 'love'

def speak(word):
    word += ' you'
    print word

def main():
    speak(word)
    print word
```

שימוש במילה global:

```
word = 'love'

def speak():
    global word
    word += ' you'
    print word

def main():
    speak()
    print word
```

- מה תהיה התוצאה בכל מקרה?

Scope של משתנים

- טיפ לתכנות נכון:
  - אם רוצים שפונקציה תשנה משתנה, מומלץ להעבירו כפרמטר ולהחזיר אותו
  - שימוש ב-global מסוכן,

פונקציה משנה ערכים שיתר הקוד משתמש בהם

תרגילי פונקציות

- כיתבו פונקציה בשם factorial שמחזירה את התוצאה של 5! (5 עצרת)
- כיתבו פונקציה בשם beep שמקבלת מחרוזת ומחזירה את המחרוזת ועוד beep בסופה
- כיתבו פונקציה בשם kefel שמקבלת שני מספרים ומחזירה את המכפלה שלהם, או 0 אם התוצאה שלילית
  - אפשר לכתוב return יותר מפעם אחת בפונקציה

הרחבה: פייתון, מתחת למכסה המנוע

- ID
- כיצד פייתון מעבירה פרמטרים לפונקציות

המחשת עקרון התרגום לשפת מכונה בזמן ריצה

```
def check(num1):
    if True:
        print 'OK'
    else:
        bla(blabla)


def main():
    check(1)
```

```
OK

Process finished with exit code 0
```

- זוכרים שאמרנו שפייתון מתרגם פקודות לשפת מכונה רק כשהוא מגיע להריץ אותן?
- אפשר לשים פקודות שגורמות לשגיאה, כל עוד הן עונות לכללי הדקדוק של פייתון התוכנית תרוץ
  - בדוגמה הבאה הפונקציה bla אינה מוגדרת
  - למרות זאת התוכנית רצה ומדפיסה OK

id(), is

- הפונקציה id() מקבלת אובייקט (שם כללי למשתנה או פונקציה) ומחזירה את הכתובת שלו בזיכרון
- אם לשני אובייקטים יש את אותו id(), זה אומר שהם מצביעים על אותו אובייקט בזיכרון
  - is מבצע את השוואה הזו
- נסו למצוא את הכתובת בזיכרון של הפונקציה id

```
>>> hi = 2
>>> bye = 2
>>> id(hi)
30585724
>>> id(bye)
30585724
>>> hi is bye
True
```

Data:	‘T’	‘a’	‘l’
Address:	12345678	12345679	12345680

↑

name = ‘Tal’

איך מועברים הפרמטרים לפונקציה?

- כזכור קיימות שתי שיטות להעברת פרמטרים לפונקציה:
  - Pass by value
  - Pass by reference

Pass By Value -תזכורת

- בשיטה זו מועבר לפרוצדורה ערך הפרמטר
- על המחסנית נוצר העתק של הפרמטר
- לפרוצדורה אין גישה לפרמטר המקורי
  - אינה יכולה לשנות את ערכו, רק את ההעתק

תזכורת -Pass By Reference

- מעבירים לפרוצדורה את הכתובת של הפרמטר
- על המחסנית לא נוצר העתק של הפרמטר
- לפרוצדורה יש גישה לכתובת בזיכרון שמכילה את הפרמטר
  - יכולה לשנות את ערכו

גבהים
תכנית הסייבר הלאומית

מה קורה בשפת פייתון?

```
1   def func(x):
2       print 'id(x): {}  x: {}'.format(id(x), x)
3       x = 'Bye'
4       print 'id(x): {}  x: {}'.format(id(x), x)
5
6
7   def main():
8       x = 'Hi'
9       print id(x)
10      func(x)
11      print id(x)
```

```
37058640
id(x): 37058640  x: Hi
id(x): 37058000  x: Bye
37058640
```

לפני השינוי, אותו id()

אחרי השינוי, id() שונה

מחוץ לפונקציה, ה-id() והערך המקורי ללא שינוי

סיכום

- פייתון מעביר לפונקציות by reference
- **אבל**, אם מנסים לשנות אותו, נוצר אובייקט חדש*
  - אפקט דומה ל-by value

* רק אם הוא immutable- נלמד בהמשך