

בד"כ בתוכנית פייתון תראו את פקודת הפעלת התוכנית הבאה (במקום רק (main():

```
if __name__ == '__main__':  
    main()
```

הסיבה היא שמודול (או קובץ) פייתון יכול גם להיות מבוצע כתוכנית וגם מוכלל בתוך תוכנית אחרת ע"י פקודת ה-import.

כלומר אם כתבנו תוכנית שימושית שבה יש פונקציות שאנו רוצים לעשות בהן שימוש בתוכנית חדשה (כמו פונקצית סיפריה), אפשר לקרוא לה עם import (כמו using ב-C#)

לדוגמה, נניח שכתבנו את התוכנית השימושית הבאה:

```
def square(x):  
    return x*x  
def main():  
    print(square(42))  
main()
```

ואנו רוצים למחזר את square במקום לכותבה מחדש. ניכתוב תוכנית חדשה בשם: mygame.py

```
#mygame.py  
  
import mymath # Brings in square function  
def main():  
    print("This is mygame")  
    print(mymath.square(17))  
main()
```

אם נריץ את mygame.py מה נקבל?

```
1764  
This is mygame  
289
```

מדוע קיבלנו את מספר 1764?

על ידי פקודת ה-import לא רק שהכללנו את mymath.py בתוך mygame.py, אלא גם ביצענו אותו (בעזרת ה: main())

כדי למנוע זאת היינו יכולים בתוך mymath.py להפעיל את main() בצורה הבאה:

```
if __name__ == '__main__':  
    main()
```

זה גם יאפשר לנו להריץ את mymath.py כתוכנית עצמאית, וגם להכלילו למשל ב-mygame.py

לכן מעתה והלאה נפעיל את ה-main() בצורה זו בכל תוכנית.

כאשר המודול מיובא ע"י import, ישנו משתנה מערכת של פייתון, בשם: __name__, אשר מקבל את שם המודול המיובא (בלי ה.py), לעומת זאת בתוך התוכנית המתבצעת, ערכו של המשתנה הזה הוא: '__main__' (אם תנסו להדפיס את בתוך mymath.py כך: print(mymath.__name__) תקבלו: mymath.

כדי לוודא שפיית'ון אינו מבצע פקודות שאינו מגיע אליהם במהלך הביצוע אפשר לנסות ולהריץ את הקוד הבא (עם חלק מהקוד שאינו קיים):

```
def main():
    if 1 == 1:
        print("one equals one, duh!!!")
    else:
        do_other_thing() # This function does not exist
main()
```