מה זה פייתון?

רקע –

פייתון היא שפת תכנות עילית, המשתמשת במודל ביצוע מפורש (interpreter) והיא שפה רב תכליתית.
הפילוסופיה של עיצוב השפה היא שהקוד יהיה קריא כמה שאפשר.
כלומר שהקוד יהיה קרוב יותר לשפת בני האדם: נקי, לוגי ומצומצם יותר עבור פרויקטים גדולים וקטנים כאחד.
הפילוסופיה של השפה ,המסוכמת בקובץ "The zen of python" כולל מימרות (היגדים) כגון:

- יפה עדיף על מכוער.
- מפורש עדיף על מרומז (implicit)
- פשוט עדיף על מורכב.
- מורכב עדיף על מסובך.
- קריאות נחשב כמעלה.

יותר משהיא מכילה את הפונקציונליות שלה בהתבסס על הפילוסופיה של השפה, פייתון גם עוצבה להיות שפה הניתנת להתרחבות (extensible) מה שהופך אותה לפופולרית במיוחד כאמצעי להוספת ממשקים ניתנים לתכנות לממשקים קיימים.

פייתון היא שפה מרובת פרדיגמות המאפשרת תכנות מונחה עצמים, תכנות פרוצדורלי, ואפשר גם לומר שתכנות פונקציונלי.

את השפה יזם חידו ואן רוסו בשלהי שנות השמונים, והתחיל לממשה בתחילת שנות התשעים כממשיכת דרכה של השפה ABC.
שם השפה הוא מחווה לתוכנית הטלוויזיה "הקרקס המעופף של מונטי פייתון".

מהדר ומפרש-

תכנות דינאמי ותכנות סטטי-

פייתון היא שפת תכנות דינאמית.
ההבדל בין שפות תכנות דינאמיות לשפות תכנות סטאטיות הוא כזה:
שפות תכנות סטאטיות הכוונה שהטיפוסים של המשתנים מוגדרים מראש, כך שהם נבדקים עוד לפני הרצת התוכנית, פעולה שנעשית בד"כ ע"י הקומפיילר(המהדר).
דוגמא לשפות תכנות סטאטיות הן כל אותן שפות שיורשות מ-C למשל ג'אווה, C++ C# וכו'.
הסינטקס שלהן בד"כ נראה כך:

```
public void foo() {
    int x = 5;
    boolean b = true;
}
```

כאן לפני שנותנים ערך ל-x או b מגדירים את הטיפוס שלהם (int ,bool,float...).
היתרון המשמעותי ביותר שיש לשפות כאלה הוא שהרבה יותר קל לזהות שגיאות תכנות לפני שהן קוראות בפועל, כלומר

לזהות אותן עוד בזמן קומפילציה ולא בזמן ריצת התוכנית.
זה מקל גם על ה-IDE (integrated development environment) לסייע לנו לזהות שגיאות אף לפני שקימפלנו את הקוד, וככל שסביבת הפיתוח מכירה יותר את המשתנים יהיה לה קל יותר להגיד לנו אם השתמשנו במשתנה שלא הוגדר או שחייב להיות מוגדר עם יצירת אינסטנס שלו(למשל משתנים שהם const או מצביעים).
וכמובן זמן ריצה- מתי שהקומפיילר כבר מכיר את הקוד והטיפוסים השונים שבו הוא יכול להסיק מראש כמה זיכרון יש להקצות ואין צורך בחישובים מיותרים בזמן אמת, מה שמפחית את זמן ריצת התוכנית משמעותית.

ומהצד השני יש לנו שפות תכנות דינאמיות שבהן טיפוס המשתנים מוגדר בזמן ריצת התוכנית
למשל הקוד הבא בשפות דינאמיות הוא לגיטימי:

```
x = 1
x = 'one'
```

למרות שהגדרנו ל-x להיות משתנה מטיפוס int אנחנו משנים אותו אח"כ לטיפוס מחרוזת, על אף שהוא כבר הקצה משאבים לint.
שגיאות בשפות דינאמיות נבדקות רק כאשר התוכנית הגיעה לשורה שבה מתרחש הקוד עם השגיאה.
למשל בשפות סטטיות השגיאה הבאה תיתפס עוד בזמן קומפילציה אם לא ע"י ה-ide :

```
_zero = 1 - 'one'
```

בשפות דינאמיות נדע על השגיאה רק כאשר נגיע לאותה שורה בקוד וכנראה שהאופרטור '-' יזרוק אותה בזמן ריצה.
אך עם זאת יש גם הרבה יתרונות לשפות דינמיות על שפות סטאטיות:
השפות הדינמיות הרבה יותר תמציתיות- שפות סטטיות בדר"כ דורשות הרבה מאוד 'קוד מוקדם' עד שנוכל להשתמש במשתנה למשהו שימושי, למשל הקטע קוד הבא שנכתב בג'אווה לעומת אותה פונקציונליות רק בפייתון:

```
// Java Hello World
package helloworld;

import java.util.Scanner;


public class HelloWorld {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Please enter your name: ");
        String name = scanner.nextLine();
        System.out.println("Hello " + name);
    }
}


# Python Hello World
name = input("Please enter your name: ")
print(f"Hello {name}")
```

קל לראות כאן את ההבדל בין כמות השורות הדרושות בשביל לקבל קלט ולהדפיסו בשפת ג'אווה לעומת הקלות שבה ניתן לבצע את אותה הפעולה בדיוק רק בפייתון.
קוד קצר יותר הוא גם הרבה יותר קריא, לאו דווקא יותר מובן, אלא יותר קל לעקוב אחרי השתלשלות האירועים בקוד, מה שיכול להקל על מתכנתים חדשים "להיכנס" אליו.
גם יותר פשוט ללמוד שפות דינאמיות, השפה הרבה פחות עמוסה בחוקים והסינטקס הרבה יותר קרוב לצורת מחשבה אנושית.
עוד יתרון שיש לשפות דינאמיות שניתן להריץ את הקוד שלהן בחלקים ואין צורך לקמפל את כל התוכנית או ליצור קוד במיוחד כדי להריץ קטע קוד ספציפי.
וכמובן שימוש פשוט יותר בספריות חיצוניות (API וכו'). לשפות דינאמיות יש בדר"כ יותר ספריות חיצוניות שניתן לייבא לפרויקט ככל הנראה בשל הקלות שבה ניתן לתכנת בהן, אך זה גם יכול להוות חסרון במובן מסוים, שאומנם יש יותר מבחר, אך רמת המתכנתים שיש לספריות בשפות דינאמיות היא לא חד משמעית בשל הקלות ללמוד אותן

לעומת ספריות של שפות סטטיות שבדר"כ רמת המתכנתים בהן יותר גבוהה.