

From the tkinter reference

To conclude, let's summarize some of the key steps involved in designing an application:

Depending on what you want to design, choose a suitable data structure to represent your needs logically.

תלוי בעיצוב שתמצה לתוכנה, בחר את מבנה הנתונים ההגיוני לצרכיך.

If required, combine primitive data structures to form complex structures like, say, a list of dictionaries or a tuple of dictionaries.

אם יש צורך במבנה נתונים מורכב, צרף מספר מבנים בסיסיים, למשל רשימת מילונים

Create classes for objects that constitute your application. Add attributes that need to be manipulated and methods to manipulate those attributes.

צור מחלקות עבור עצמים שמרכיבים את היישום. הוסף מאפיינים נדרשים לעיבוד ומתודות לעבד אותם.

Manipulate attributes using different API provided by a rich set of Python standard and external libraries.

עבד מאפיינים בעזרת ממשקי תוכנה של פייתון ומודולים שקיימים בשפה.

We tried to build several partly-functional applications in this book. And then we put up an explanation for the code. However, when you try to explain a software development process in a sequential text, you sometimes mislead your readers to imply that development of software programs is a linear process. This is hardly true.

ניסינו לבנות מספר ממשקים עבור יישומים שעובדים בחלקם בספר זה, ואז להסביר את הקוד. אולם, כאשר מנסים להסביר פיתוח תוכנה בתיאור מילולי רציף, לעיתים הקורא טועה לחשוב שגם פיתוח התוכנה הוא רציף. מחשבה זו אינה נכונה ככלל.

Actual programming doesn't usually work this way. In fact, small-to-medium-sized programs are normally written in an incremental trial and error process where assumptions get changed and structures modified throughout the course of application development.

תכנות בד"כ אינו עובד כך. תוכניות קטנות או בינוניות נבנות על ידי ניסוי וטעיה, נדבך אחרי נדבך, כאשר הנחות מסוימות משתנות ומבנים מוחלפים תוך כדי תהליך התכנות.

Here is how you would develop a small to medium application:

1. Start with a simple script.
2. Set a small achievable goal, implement it, and then think of adding the next feature to your program in an incremental fashion.
3. You may or may not introduce a class structure initially. If you are clear about the problem domain, you may introduce the class structure right from the very beginning.
4. If you are not sure about the class structure initially, start with simple procedural code. As your program starts to grow, you will probably start getting lot of global variables. It is here that you will start getting a glimpse of the structural dimensions of your program.

It is now time to refactor and restructure your program to introduce a class structure.

It is also important not to be unnecessarily bogged down by ever evolving jargons in the technical world. Programming is less about knowing a particular API or even a particular programming language. You can literally get to know the basic constructs of a programming language in a small sitting. Programming is rather a tool for finding solution to your immediate problems.

That brings us to the end of the book. I hope this book has taught you something about GUI programming with Python and Tkinter.

Beyond reading books, there is really no substitute for doing some original GUI programming. So, take up an original programming challenge and execute it for the fun of it.

How you implement it is a matter of individual experiences and taste. Do what feels comfortable to you, but keep yourself open to the idea of continuous refactoring at every stage of development.

Light bulb

If you are writing a small program, the evolutionary trial and error strategy works well.

If, however, you get into developing medium to large-scale applications, it is better to do some serious upfront planning before you sit down to write your code, because the cost of failure of a large program is way higher than what we can generally afford.

An analogy would explain this better. You can build a small shed on a trial and error basis, but you would not attempt to build a skyscraper without some serious planning.

אם אתם כותבים תוכנית קטנה, האסטרטגיה של ניסוי וטעיה בפיתוח עובדת טוב.

אולם, אם ברצונכם לפתח אפליקציה בגודל בינוני או גדול (פרוייקט), עדיף לבצע תיכנון מקדים רציני לפני כתיבת הקוד. זאת מהסיבה שמחיר הכישלון של תוכנית גדולה הינו גדול בהרבה ממה שתוכלו להרשות לעצמכם.

האנלוגיה הבאה יכולה להמחיש זאת. בנית אוהל על ידי ניסוי וטעיה יכול להצליח, אבל לא סביר שתנסו לבנות גורד שחקים ללא תיכנון רציני לפני הבניה.