Python TkInter: Eine Einführung

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte BeschreibungErstellt von Tobias Kisling

Python TkInter: Eine Einführung

TkInter ist eine einfache Python-Library, um grafische Benutzeroberflächen zu erstellen, die mit einer Python-Standart-Installation mitgeliefert wird. Sie ermöglicht die einfache Visualisierung von Python-Code, die gut für Objekt-Orientiertes-Programmieren geeignet ist.

Ein großer Vorteil von TkInter ist, dass sie plattformübergreifend funktioniert und mit Python mitgeliefert wird, dadurch kann man sie einfach in .exe- oder .app-Dateien konvertieren. Die Software ist jedoch im Vergleich zu 3rd-Party-Software in der Funktion limitiert, eignet sich jedoch gut für kleinere Apps. Im Verlauf dieser Einführung erkläre ich zunächst, was die einzelnen Elemente können und wie man sie nutzt, dann zeige ich TkInter in der Praxis.

Voraussetzungen:

* Kenntnis über Variablen (insbesondere global)
* Grundlegendes Wissen über Funktionen
* Grundlegendes Wissen über den Aufbau von Fenstern

*Wichtig: Bei TkInter, genau wie bei Python generell, ist Groß/kleinschreibung sehr wichtig. Es empfiehlt sich für TkInter sehr, eine IDE wie Visual Studio Code zu verwenden, weil damit der Code sehr schnell ausführbar ist.*

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

# Die Library importieren

TkInter kommt mit seinen Funktionen standardmäßig bei Python 3 mit. Geraten ist, immer die neueste Version zu nutzen.   
Zunächst müssen wir TkInter importieren. Dafür geben wir in eine neue Python-Datei (dateiname.py) ein:



Damit haben wir schon einmal die Basis importiert. Abhängig davon, was man damit jetzt machen will, kann man eine Vielzahl an Zusätzen importieren, ebenfalls inkludiert in Python. Einige häufig benutzte stelle ich hier vor:



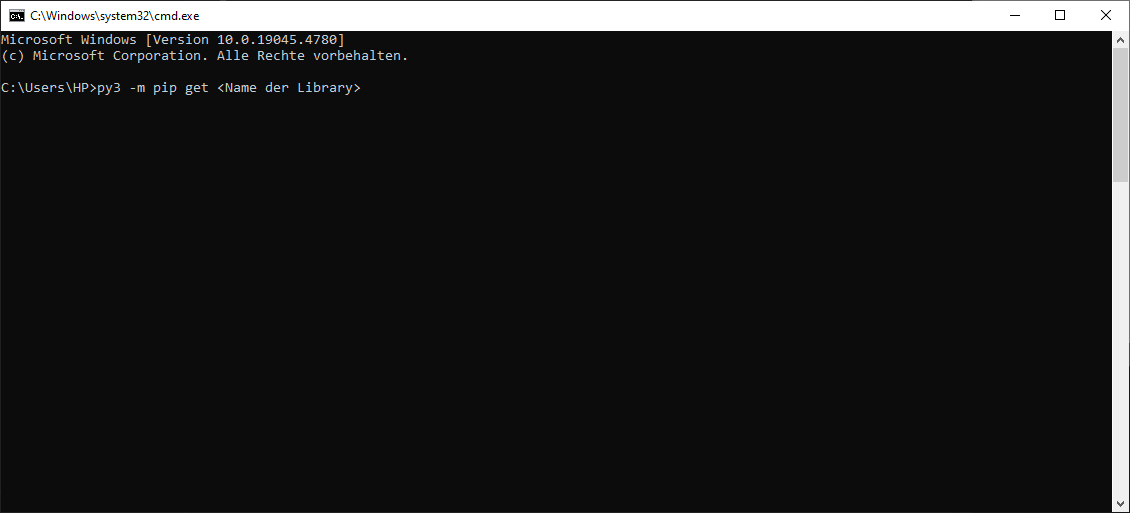
Font – Ermöglicht, unterschiedliche Schriftarten zu verwenden.

Messagebox – Damit können Warnung, Errors und Infos am Display angezeigt werden.

PhotoImage – Dies bringt die Möglichkeit, Grafiken anzuzeigen

Diese Zusätze werden wir uns im Laufe dieser Einführung genauer ansehen.

Andere Zusätze findet man in einer Vielzahl im Internet, zum Beispiel auf PyPi.com, der offiziellen Library-Seite, deren Inhalte man mittels *py3 –m pip get <name der library>* aus dem Terminal herunterladen kann.



# Das erste Fenster

Haben wir nun TkInter erfolgreich importiert, geht es nun daran, das erste

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungFenster damit zu erstellen.

„neuesFenster“ wir hier zur Fenstervariable, mit der wir alle Widgets definieren.

Darunter kommt „neuesFenster.mainloop()“, was Python erklärt, dass ab hier der „Loop“ endet. Zwischen die beiden Elemente kommen nun unsere Widgets.

Jetzt haben wir, wenn wir den Code ausführen, so ein Fenster:

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungNun sollten wir Python erklären, wie unser Fenster aussehen soll. Das tun wir wie folgt:

Wir konfigurieren unser Fenster mit der Variable („neuesFenster“) und einem Punkt („.“). Dann definieren wir, was wir konfigurieren wollen-dahinter. Hier am Anfang einmal:

* .title(“”) – hier kommt der Name des Fensters hinein
* .geometry(“”) – hier können wir die Fenstergröße ändern (Breite x Höhe)
* configure() – hier können wir alles mögliche einstellen, dazu mehr später. Hier stellen wir einmal den Background (bg=“Farbcode“) ein.

Das lässt unser Fenster dann so aussehen: Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Als nächstes fügen wir ein Menü hinzu. Menüs sehen unter Windows 10 so aus:

Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

Um ein Menü hinzuzufügen, nutzen wir den folgenden Code:



Damit ist das Menü initiert. Jetzt fügen wir eine Kaskade („Datei“, „Bearbeiten“) und eine Schaltfläche hinzu:

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

Das erzeugt ein Menü, das so aussieht:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Unsere Schaltfläche können wir anklicken, allerdings bringt das jetzt natürlich nichts, weil kein Command hinterlegt ist, der eine Funktion anruft. Diesem Thema werden wir uns später widmen, wir haben ja noch keine Funktionen.

Jetzt wird es Zeit, ein paar „Widgets“ hinzuzufügen. So heißen Elemente, die auf unserem Fenster angezeigt werden. Als Erstes fügen wir einmal einen Text hinzu, bei TkInter genannt Label.



Dieser Code fügt ein Label hinzu. Der Code genau enthält folgende Elemente:

* Das Keyword Label – Definiert die Art des Widgets.
* neuesFenster – Dies ist höchst wichtig, weil es Python sagt, wo das Widget hinzugefügt werden soll.
* font – Gibt die Schriftart an, wenn keine spezifische Schriftart benötigt wird (Font muss importiert sein), zusätzlich kann die Schriftgröße geändert werden:

*(neuesFenster, font=(“TkDefaultFont“, 15) ,text=“Text“)*

15 ist hierbei die Schriftgröße.

* Text enthält den Text, der angezeigt wird

Wenn wir das Programm jetzt ausführen, ist allerdings nichts erkennbar. Das liegt daran, das Python nicht weiß, wo im Fenster das Label platziert werden soll.

# Platzieren mit Python

Um dies Python zu erklären, haben wir drei Optionen:

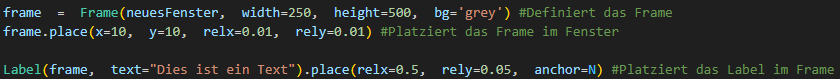
**Place, Pack und Grid**

So heißen die Widget-Manager, die uns TkInter zur Verfügung stellt. Alle haben ihre Vor- und Nachteile. Wir werden im nachfolgenden Tutorial zunächst Pack nutzen, ich stelle trotzdem einmal alle vor.

**Place**

Dies ist die komplizierterste Methode, die wir nur selten anwenden sollten. Sie funktioniert, in dem wir die Bildschirmkoordinaten angeben. Dafür müssen wir zuerst ein *frame* angeben, dann geben wir für jedes Widget an, wo es im *frame* sein soll.

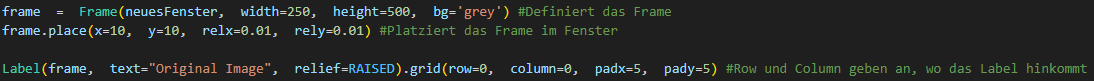
Beispiel:



**Grid**

Grid funktioniert ein wenig wie eine Excel-Datei. Man erstellt ein Grid, dass ähnlich mit Zeilen und Spalten arbeitet und gibt dann den Platz darin an, wo das Widget sein soll.

Beispiel:

**

**Pack**

Pack ist der einfachste Manager in TkInter. Wir werden ihn deshalb auch im weiteren Tutorial verwenden. Bei Pack platziert TkInter das Widget einfach im Fenster und übernimmt den Rest. Er ist einfach, aber auch limitiert, eignet sich daher gut für kleinere Projekte.  
Beispiel:



Hurra, wenn wir das Programm jetzt ausführen, dann erscheint unser Text im Fenster:

Ein Bild, das Text, Screenshot, Schrift, weiß enthält.

Automatisch generierte Beschreibung

Jetzt haben wir gelernt, wie wir Fenster erstellen, Menüs hinzufügen und Labels zu platzieren. Im nächsten Kapitel werden wir lernen, Funktionen hinzuzufügen, damit unser Code auch etwas macht. Allerdings sollten Sie dafür wissen, wie man Funktionen definiert und anruft. Falls Sie dies noch nicht wissen, erkläre ich dies hier kurz:

So definiert man eine Funktion.



So ruft man eine Funktion an und gibt eine Variable aus.

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

So erklärt man eine Variable global, um sie in einer Funktion zu nutzen.

Ein Bild, das Text, Schrift, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung

So ruft man eine Funktion generell an:

Ein Bild, das Text, Schrift, Screenshot, Design enthält.

Automatisch generierte Beschreibung

Wie man Funktionen im Kontext von TkInter nutzt, wird später erklärt.

# Widgets und funktionen

Unser Fenster soll nun etwas tun. Aus diesem Grund werden wir nun erstmal einen Knopf („Button“) hinzufügen. Dieser Knopf soll das Label, dass wir vorhin erstellt haben, ändern, sodass es einen anderen Text anzeigt. Dafür müssen wir erstmal etwas am Code ändern:



Wir müssen dem Label eine Variable zuweisen, damit wir sie ändern können. Außerdem darf .pack() nicht mehr am Ende des Labels stehen, sondern muss in einer neuen Zeile der Label-Variable zugewiesen werden.

Nun definieren wir eine Funktion:



Der Code bewirkt folgendes:

* def LabelTextÄndern(): - Definiert eine neue Funktion mit diesem Namen.
* Label.config(text=“Hallo Welt!“) – Ändert die Konfiguration des Labels und setzt den Text auf „Hallo Welt!“.

*Wichtig: Die Funktionsdefinition muss IMMER über dem Anruf sein, sonst funktioniert die Funktion nicht! Generell sollten Funktionen immer über den Widgets platziert sein!*

Jetzt haben wir eine Funktion, die den Text ändern kann. Jetzt benötigen wir allerdings noch etwas, dass die Funktion anruft und damit ausführt. Dafür werden wir einen Button nutzen:



Der Code bewirkt folgendes:

* Button1 = Button(neuesFenster, text=“Klick mich an!“, command=LabelTextÄndern) – Erstellt einen an eine Variable gebundenen Button in ‚neuesFenster‘ mit Text und setzt den command (Trigger) auf unsere vorher erstellte Funktion.
* Button1.pack() – Platziert den Button im Fenster.

Dieser Code erstellt dann so ein Fenster:

Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

Automatisch generierte Beschreibung

Und wenn wir den Button anklicken, sieht das Fenster so aus:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Als nächstes programmieren wir eine Texteingabe und lernen, mit Inputs umzugehen. Außerdem werden wir mit der MessageBox-Funktion arbeiten.

Unser Code soll es dem User nun möglich machen, seinen Namen im Fenster einzugeben, den Namen nach einem Klick auf einen Button in einer Variable abspeichern und dann diese über eine MessageBox ausgeben. Dafür erstellen wir erstmal einen Input:



Der Code bewirkt folgendes:

* Wir erstellen einen an die Variable InputName gebunden TextInput im Fenster ‚neuesFenster‘ mit der Höhe 1 (eine Zeile) und der Breite 20 (Zwanzig Zeichen) und platzieren diesen.

Das sieht so aus:

Ein Bild, das Screenshot, Text enthält.

Automatisch generierte Beschreibung

Jetzt können wir noch ein Label erstellen, um dem User zu erklären was er tun soll:



Das sieht dann so aus:

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Tipp: Um eine Leerzeile einzufügen können Sie den folgenden Code nutzen:

*Label(neuesFenster, font=“TkDefaultFont“, text=“\n“).pack()*

Jetzt haben wir einen Input, um den Text aufzunehmen. Um den Input aufzunehmen und anzuzeigen, benötigen wir jedoch eine weitere Funktion und einen Button, um diese auszuführen. Dafür fügen wir diesen Code hinzu:

Der Code bewirkt folgendes:

* Wir definieren eine neue Funktion
* Dann setzen wir die Variable ‚input1‘ auf ‚InputName.get‘ – Die .get-Funktion holt Daten aus Inputs in TkInter – und entfernen Leerzeilen mit .strip() – Die Daten in der .get()-Klammer definieren, dass wir Text von Anfang bis zum Ende des Inputs
* Wir zeigen eine MessageBox im Fenster an, deren Text „Hallo, [Eingabe des Users]“ ist.

Jetzt brauchen wir noch einen Button, um die Funktion zu aktivieren:

Das sieht dann so aus:

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Und wenn Sie etwas eingeben und denn Knopf klicken, kommt eine solche Nachricht:

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

# funktionen im Menü verlinken

So, nun wissen wir, wie man Funktionen erstellt. Nun werden wir lernen, eine Funktion im Menü zu verlinken. Außerdem werden wir ein zusätzliches Fenster erscheinen lassen.

Wir wollen nun eine Funktion erstellen, die ein Fenster öffnet, in dem Text steht. Dafür erstellen wir diesen Code:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Der Code bewirkt folgendes:

* Wir erstellen eine neue Funktion.
* Wir erstellen ein an die Variable ‚Fenster2‘ gebundenes Fenster auf Basis des Hauptfensters (Toplevel auf neuesFenster).
* Wir legen die Maße fest.
* Wir fügen ein Label hinzu.

Jetzt verlinken wir die Funktion im Menü:

Erinnern Sie sich noch an die Menüleiste, die wir weiter oben erstellt haben?

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

Wir werden zur „Schaltfläche 1“ jetzt einen Command hinzufügen, der unsere Funktion startet. Das geht so:



Wir fügen hinter dem Label der Schaltfläche „ ‚command=WeiteresFensterÖffnen)“ hinzu.

Damit, wenn wir das Fenster öffnen und ‚Schaltfläche1‘ im Menü anklicken:

Ein Bild, das Text, Screenshot, Software, Webseite enthält.

Automatisch generierte Beschreibung