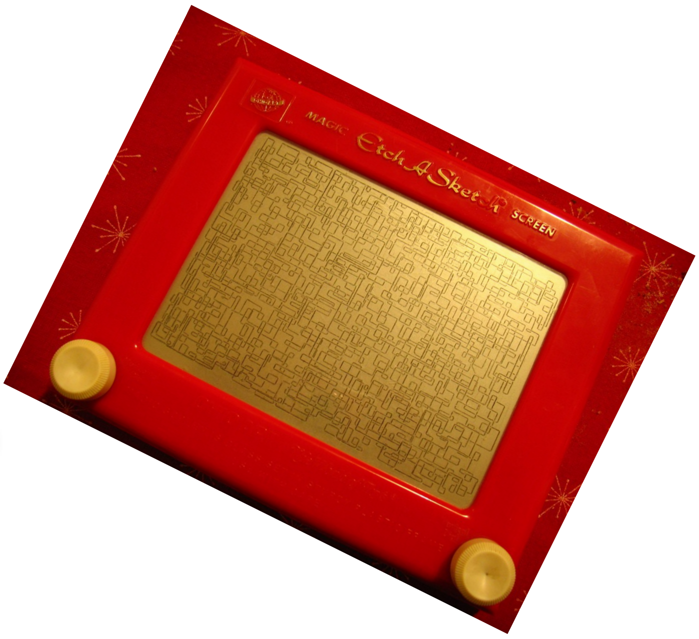
Python-A-Sketch



Name:______________________________

TLundComputing

Table of Contents

Setting Up

Before we do any work today, we need to do the following:

1. Open IDLE

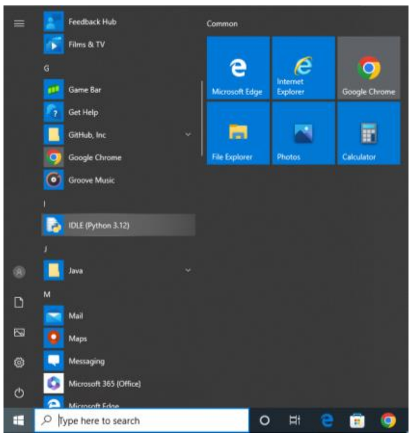


Figure 1 - Finding IDLE

2. Click File ⟶ New File

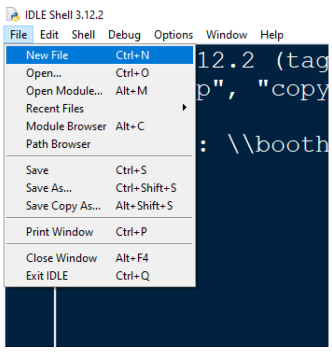


Figure 2 - Creating a new file.

3. Click File ⟶ Save as...

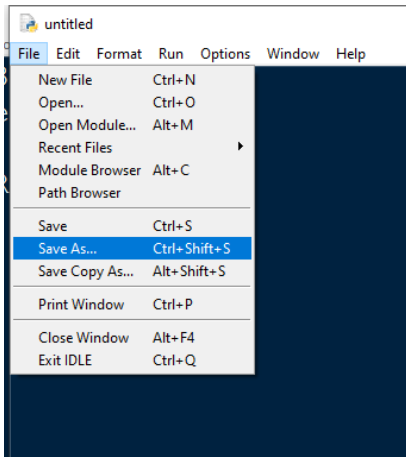
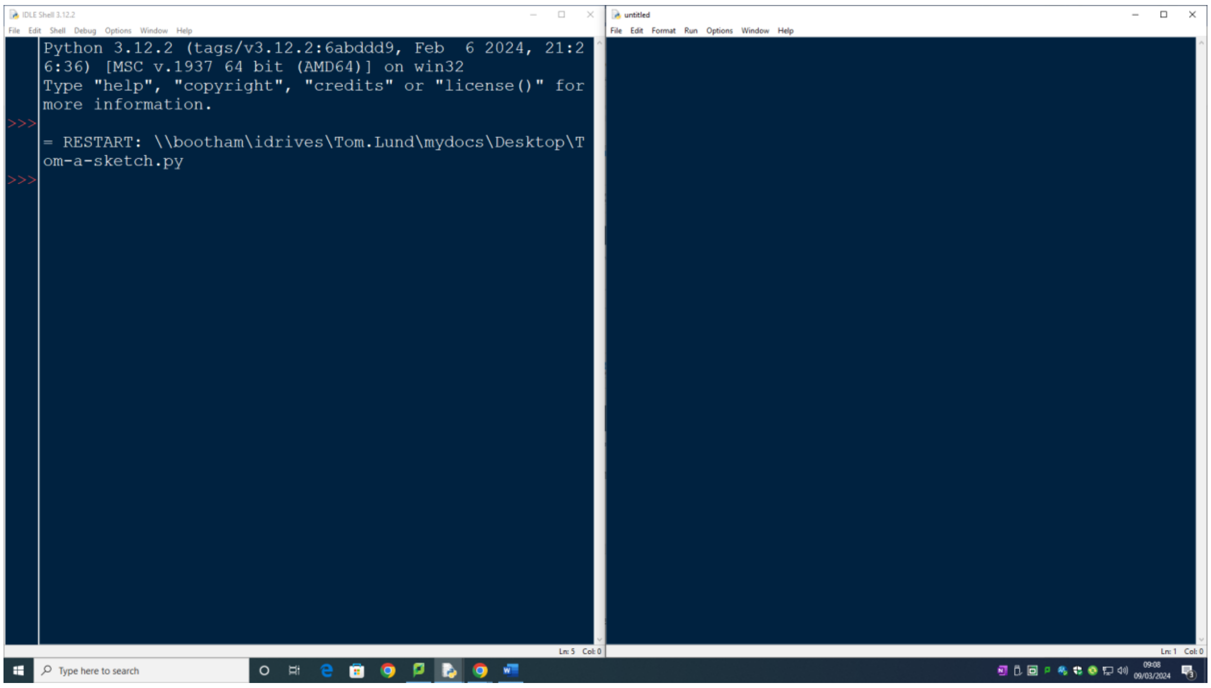


Figure 3 - Saving the Python file.

4. Navigate to your documents.
5. Name the file NAME-a-Sketch (e.g. Tom-a-Sketch)

6. Put your windows side by side so that you can see your code and the result.



```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:2
6:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
= RESTART: \\bootham\idrives\Tom.Lund\mydocs\Desktop\T
om-a-sketch.py
>>>
```

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
untitled
File Edit Format Run Options Window Help
Ln: 5 Col: 0
Ln: 1 Col: 0
Type here to search

Figure 4 - Windows side by side.

Plans for today

Today we are going to make a computerised version of an Etch-a-Sketch. Wait… What even is an Etch-a-Sketch?

The Etch-a-Sketch is a classic drawing toy from 1960 where people would use two dials (one for left and right, and one for up and down) to draw different pictures and other pieces of art. Once you finished your drawing you would give it a shake and the picture would erase for you to start again! (Spin Master, 2023)

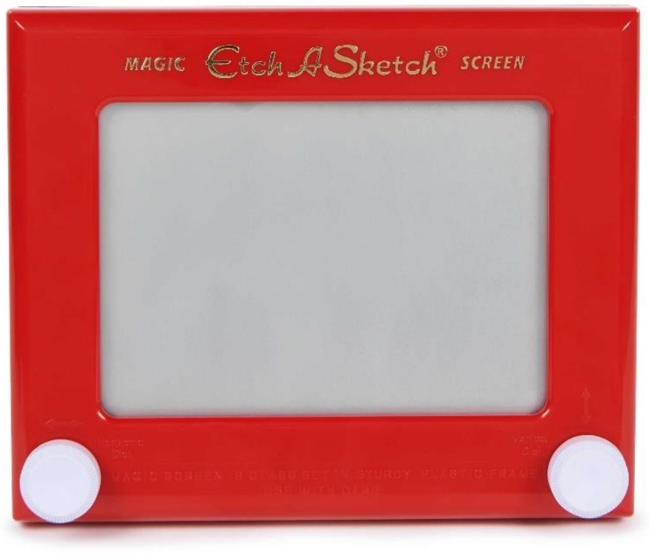


Figure 5 - Etch-a-Sketch (Spin Master, 2023)

We are going to code a simpler version of this using Python. Ours will look something like this:

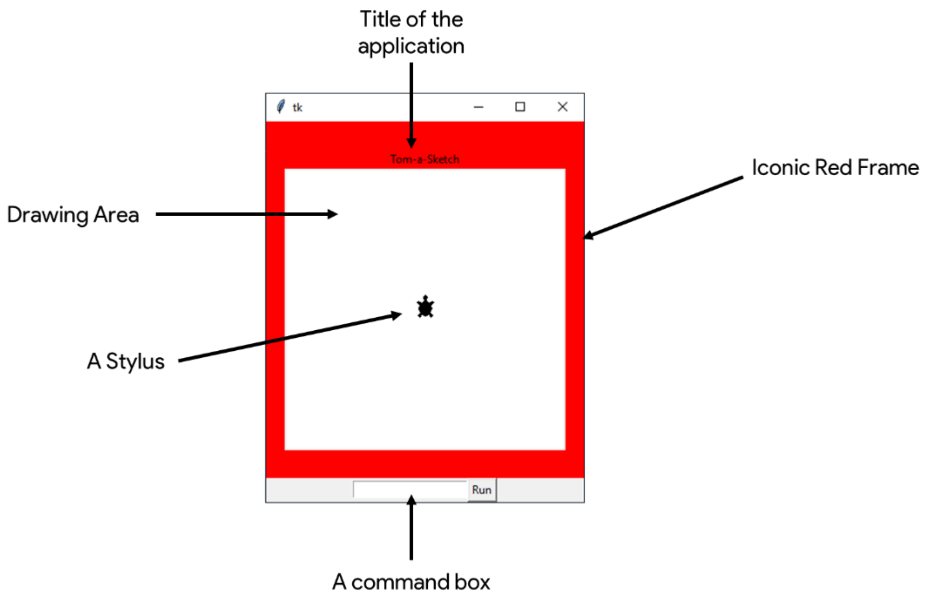


Figure 6 - Python Etch-a-Sketch.

Success Criteria

Before we begin, we need to make sure we know what we are aiming for. Let's make a list of success criteria that we can use to make sure our solution works.

Task

I have put in 3, can you think of anymore? (Hint: Look back at Figure 6 of the solution we are aiming for)

- ☐ Must have a title.
- ☐ Needs to have a box to enter commands.
- ☐ Stylus must move around the screen as directed.
- ☐
- ☐
- ☐
- ☐
- ☐

Making the Window

To make the window we are going to use a library called `tkinter`. This is a Python library used to create graphical user interfaces so that people can interact with buttons etc.

In IDLE write the following code:

```
from tkinter import *
from tkinter import messagebox as mbox
import turtle

root = Tk()

root.mainloop()
```

Press F5 on your keyboard or Run ⟶ Run Module. You will be prompted to save your work and then the program should work.

Expected Result

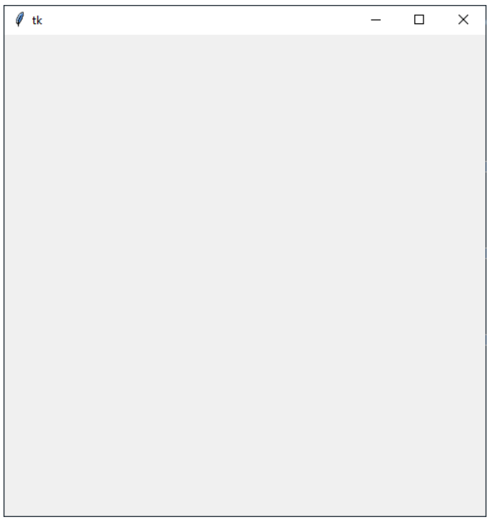


Figure 7 - Expected Blank Window

What does the code do?

```
from tkinter import *
from tkinter import messagebox as mbox
import turtle
```

The above lines import all the functionality of `tkinter` and `turtle` for Python to use within our code.

```
root = Tk()

root.mainloop()
```

The following lines do some pretty simple stuff even through they look complicated. The first line creates a window object that we can add widgets (Buttons, Entries, Frames) to. The second line puts our window on the screen for users to see and interact with.

Creating Widgets

What is a Widget?

A widget is a component that we can put on our screen. These are objects like buttons, frames, entries, and more.

Widgets We Need

We need the following widgets:

- ✓ Frame (one to make the red frame, one to hold our canvas, and one to hold our command box)
- ✓ Button (To execute our commands)
- ✓ Entry (So the user can enter their commands)
- ✓ Canvas (A place for the user to draw)

The code

```
red_frame = Frame(root, bg=”red”, width=500, height=450,
     padx=10, pady=10)
inner_frame = Frame(red_frame, width=400, height=350)
command_frame = Frame(root)

canvas = Canvas(inner_frame, width=400, height=350)
command_entry = Entry(command_frame)
command_btn = Button(command_frame, text="Run!")
```

The above code needs to go between `root.geometry()` and `root.mainloop()`

If you run your code you will see that nothing will show up on the screen yet and that is because we have not placed any of our widgets on the screen yet.

Challenge

```
inner_frame = Frame(red_frame, width=400, height=350)
```

The above code shows how we made the `inner_frame`, what do you think the code inside the brackets tells Python?

Placing Widgets

Now we have made the widgets, let's add them to the screen! We are going to use `.pack()` to put these on the screen.

The Code

This should go below the previous code you have just written.

```
red_frame.pack()
inner_frame.pack()
command_frame.pack()

canvas.pack()
command_entry.grid(column=0, row=0)
command_btn.grid(column=1, row=0)
```

Running your program should give you the following result:

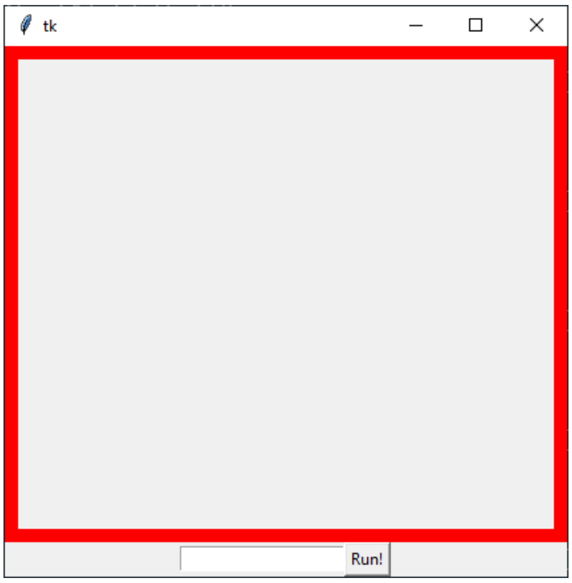


Figure 8 - Widgets Placed

Adding a Stylus in the Form of a Turtle

Our little Etch-a-Sketch is coming along brilliantly! However, we need to actually add a stylus for us to draw! Add the following bits of code between the widget creation code, and your packing code you've just written.

```
stylus = turtle.RawTurtle(canvas, shape="turtle")
stylus.left(90)
```

Challenge

What does the above code do?

Expected Result

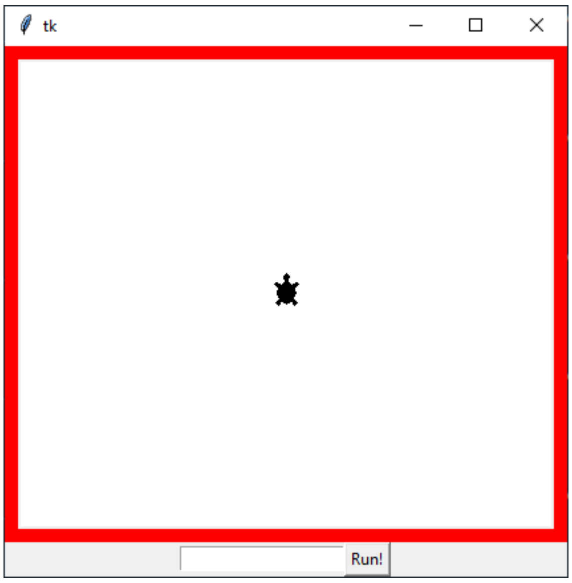


Figure 9 - Etch-A-Sketch with Stylus

Making the stylus move!

We have nearly completed our application, we just need to make our stylus move. We are going to expect that our users will enter commands that are two words (e.g. up 100 or right 300)

To do this we need to make a procedure. A procedure is a chunk of code that can run over and over again without it having to be rewritten.

Why does using procedures benefit programmers?

The Code

Write this code above the `root.mainloop()` line but below all the others.

```
def draw():
    raw_command = command_entry.get()
    command = raw_command.split(" ")
    if len(command) == 2:
        if command[0] == "up":
            stylus.setheading(90)
            stylus.forward(int(command[1]))
    else:
        mbox.showerror("Invalid command",
                       "You did not enter a valid command!")
    command_entry.delete(0, "end")

command_btn.config(command=draw)
```

There is a lot of code here for us to break up. Let's start with:

```
def draw():
```

This line of code is used to start our procedure. Anytime draw is used this code will run.

```
raw_command = command_entry.get()
command = raw_command.split(" ")
```

This chunk of code gets the text that is currently in the Entry box and then splits it up to try and get the two worded command.

```
if len(command) == 2:
    # Code is here
else:
    mbox.showerror("Invalid command",
                   "You did not enter a valid command!")
```

This if statement is what helps us check that the command entered is a valid command!

The code in the if statement looks like this:

```
if command[0] == "up":
    stylus.setheading(90)
    stylus.forward(int(command[1]))
```

This if statement sees if the first word is up. If it is then it will point the turtle in the correct direction and then move it forward as many spaces as given in the second "word".

```
command_btn.config(command=draw)
```

This line of code tells our button what procedure should run when the button is clicked.

Challenge

Try adding if statements that will then do the following directions:

- ✓ Left
- ✓ Right
- ✓ Down

What does the following line of code do?

```
command_entry.delete(0, "end")
```

Super Challenges

Try these super challenges!

1. Go to this website: https://tkdocs.com/tutorial/widgets.html#label and try and create a label in your red_frame that has the name of your program in it.

2. Add some code to the draw procedure so that when the user enters the word shake the screen clears.

3. Add some code to the draw function so that when the user enters reset the stylus goes back to the centre of the screen.

Table of Figures

References

Spin Master. (2023, January 1). *Etch a Sketch.* Retrieved from Spin Master.

Spin Master. (2023, January 1). *Etch a Sketch | Spin Master*. Retrieved from Spin Master: https://www.spinmaster.com/en-US/brands/etch-a-sketch/

License