Python & Java {4} Teachers

Custom Wardrobe

Level 2 - Python

Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

Introduction

You probably think that art and programming couldn’t be more unalike, but actually they might be more similar than you think!

There are many modern artists whose main medium is code. For example: Mark Dorf, Josh Davis and Kyle McDonald.

Here are a few websites that merge together are and programming:

- Silk – Interactive Generative Art (weavesilk.com)
- Dream by WOMBO

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Task

- For this project you are going to use you fashion skills to create a wardrobe, customizable to however you would like to dress!

Extension:

- Try finding your own photos to use in place of the clothing used already.

Funded by the Erasmus+ Programme of the European Union

Process

This program should:

- ✓Import the Tkinter graphics library,
- ✓Use subroutines to create processes for the buttons,
- ✓Successfully import photos to use for the wardrobe.

Funded by the Erasmus+ Programme of the European Union

Python Libraries

Funded by the Erasmus+ Programme of the European Union

Python Libraries are a set of useful functions that eliminate the need for writing codes from scratch.

```
1 from random import *
2
3 import Tkinter
4
5 import statistics
6
```

They can be brought into the program using the ‘import’ keyword and can save valuable time when writing complex programs.

One common example of a python library is the ‘random’ module, used often for generating pseudo-random numbers

Step 1

Importing the Tkinter library

Tkinter is a Python graphics library that is most commonly used for creating GUIs – graphical user interfaces – allowing us to make interactive menus.

```
1 import tkinter
2 from tkinter import *
3
4
```

We will be importing it to create buttons and images that are displayed on the screen.

Funded by the Erasmus+ Programme of the European Union

Subroutines

Funded by the Erasmus+ Programme of the European Union

Subroutines are sequences of instructions that perform a specific task.

- It may be easier to think of them as mini-programs within a large program.
- Subroutines consist of modules of code that perform different tasks.
- If these tasks are repeated throughout the program, they can be written as subroutines.
- Each subroutine is given a unique name so that it can be called and executed quickly throughout the program, without having to write the code again.
- This reduces the size of the code, making the program more logical and easier to read and maintain.

```
#Defining a subroutine:
def nameOfSubroutine():
        code goes here
#Calling a subroutine:
nameOfSubroutine()
```

```
1 def nameOfSubroutine(): #declaring
2     print("hello")
3 nameOfSubroutine()      #calling
```

Step 2

Defining the subroutine

This is thee first subroutine that will be bound to the button to the left of the t-shirt. The variable 'currentShirt' is then made global so that we can access the shirt – this is the variable that controls which the image is displayed – from within the subroutine.

```
5  def leftShirt():
6      global currentShirt
7
8      position = shirts.index(currentShirt)
9      position -= 1
10     if position < 0:
11         position = len(shirts) - 1
12
13     currentShirt = shirts[position]
14     labelShirt.configure(image=currentShirt)
15     labelShirt.photo = currentShirt
16     print ("updated")
```

Lines 8-11 finds the position of the shirt within the list of possible shirts. When you press the button it will move backwards through the list, however, if the position is less than zero, the position will be reset to the end of the list. Lines 13-18 updates the image of the 'labelShirt' object so that it is now a different colour shirt. This can also help with the debugging of the program.

Funded by the Erasmus+ Programme of the European Union

Step 3

Defining the second subroutine

The next subroutine is coded in then same process as the previous subroutine, but should go through the list in the opposite direction.

```
18 def rightShirt():
19     global currentShirt
20
21     position = shirts.index(currentShirt)
22     position += 1
23
24     if position >= len(shirts):
25         position = 0
26
27     currentShirt = shirts[position]
28     labelShirt.configure(image=currentShirt)
29     labelShirt.photo = currentShirt
30     print ("updated")
```

Again when the button is pressed, it will move forwards through the list. However, if the position is greater than the length of the 'shirts' array, and is therefore not representative of an item within the list, the position resets to zero.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Step 4

Defining the third subroutine

This is the third subroutine, and it will be used for the left button for the trousers. It follows the same principle as step 1 but all of the variables are now trouser related rather than shirt related.

```
32 def leftTrouser():
33     global currentTrousers
34
35     position = trousers.index(currentTrousers)
36     position -= 1
37     if position < 0:
38         position = len(trousers) - 1
39
40     currentTrousers = trousers[position]
41     labelTrouser.configure(image=currentTrousers)
42     labelTrouser.photo = currentTrousers
43     print ("updated")
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Step 5

Defining the fourth subroutine

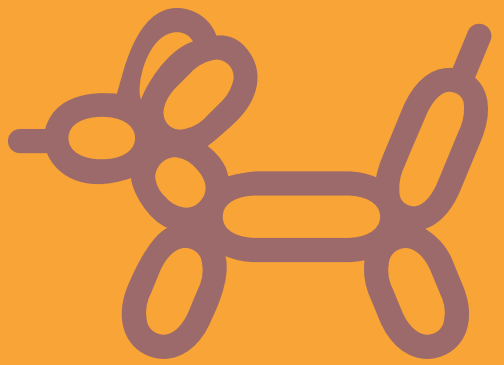
The last subroutine is created for the right button in our wardrobe program. The process is again the same as step 3 but with trouser related variables, and it also moves up the list instead of negatively.

```
45 def rightTrouser():
46     global currentTrousers
47
48     position = trousers.index(currentTrousers)
49     position += 1
50
51     if position >= len(trousers):
52         position = 0
53
54     currentTrousers = trousers[position]
55     labelTrouser.configure(image=currentTrousers)
56     labelTrouser.photo = currentTrousers
57     print ("updated")
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers



Step 6

Initializing the Tkinter window

This is the code that initialises the Tkinter window. We store the window (represented by Tk()) in the master variable.

```
60 master = tkinter.Tk()
61 master.title("Python Wardrobe")
62 master.geometry("500x500")
```

The title is then changed and the geometric properties of that window to customise its appearance and size.

Funded by the Erasmus+ Programme of the European Union

Step 7

Storing the Images

Lines 64-67 creates variables to store the shirts, put the relative path to the image. The same is then done for the trousers.

```
64 shirt1 = PhotoImage(file='images/shirt1.gif')
65 shirt2 = PhotoImage(file='images/shirt2.gif')
66 shirt3 = PhotoImage(file='images/shirt3.gif')
67
68 trouser1 = PhotoImage(file='images/Blue Jeans.png')
69 trouser2 = PhotoImage(file='images/Grey Jeans.png')
70 trouser3 = PhotoImage(file='images/Orange Jeans.png')
71
72 imgLeft = PhotoImage(file='images/left.png')
73 imgRight = PhotoImage(file='images/right.png')
```

Lines 72 and 73 import the images for the left and right arrow buttons.

Funded by the Erasmus+ Programme of the European Union

Step 8

Storing and displaying the images

These next lines make lists to store the shirt images and trouser images. This then sets up the initial shirt and trouser that will be displayed when the program first runs.

```
75 shirts = [shirt1, shirt2, shirt3]
76 currentShirt = shirts[1]
77 trousers = [trouser1, trouser2, trouser3]
78 currentTrousers = trousers[1]
79
```

Funded by the Erasmus+ Programme of the European Union

Step 9

Displaying the grid layout

Now we are using a grid layout to put the button on the screen. A button is created in the master window, with the width, height and command corresponding subroutine.

```
80 buttonLeftShirt=tkinter.Button(master, image=imgLeft, width=100, height=100, command=leftShirt) #left
81 buttonLeftShirt.grid(row=1,column=0)
82
83 buttonRightShirt=tkinter.Button(master,  image=imgRight, width=100, height=100, command=rightShirt) #right
84 buttonRightShirt.grid(row=1,column=3)
85
86 labelShirt = tkinter.Label(master, image=currentShirt, width=250, height=250)
87 labelShirt.grid(row=1,column=2)
```

The left and right arrows are set first, then the image for the correct shirt.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Step 10

Displaying the grid layout pt2

The buttons and label for trousers follow the same process as previously, the only difference is that the trouser related subroutines have been called.

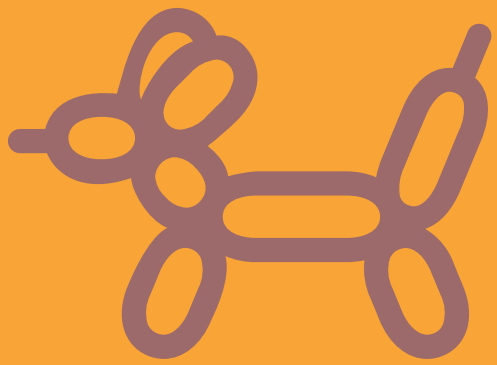
```
90
91 buttonLeftTrouser=tkinter.Button(master, image=imgLeft, width=100, height=100, command=leftTrouser) #left
92 buttonLeftTrouser.grid(row=2,column=0)
93
94 buttonRightTrouser=tkinter.Button(master,  image=imgRight, width=100, height=100, command=rightTrouser) #right
95 buttonRightTrouser.grid(row=2,column=3)
96
97 labelTrouser = tkinter.Label(master, image=currentTrousers, width=250, height=250)
98 labelTrouser.grid(row=2,column=2)
99
```

Here the pen up() function is used to make sure we don't draw any excessive lines when giving the turtle its initial placement. The turtle needs to be moved down so that when the face is drawn it doesn't cut off the ends of the screen.

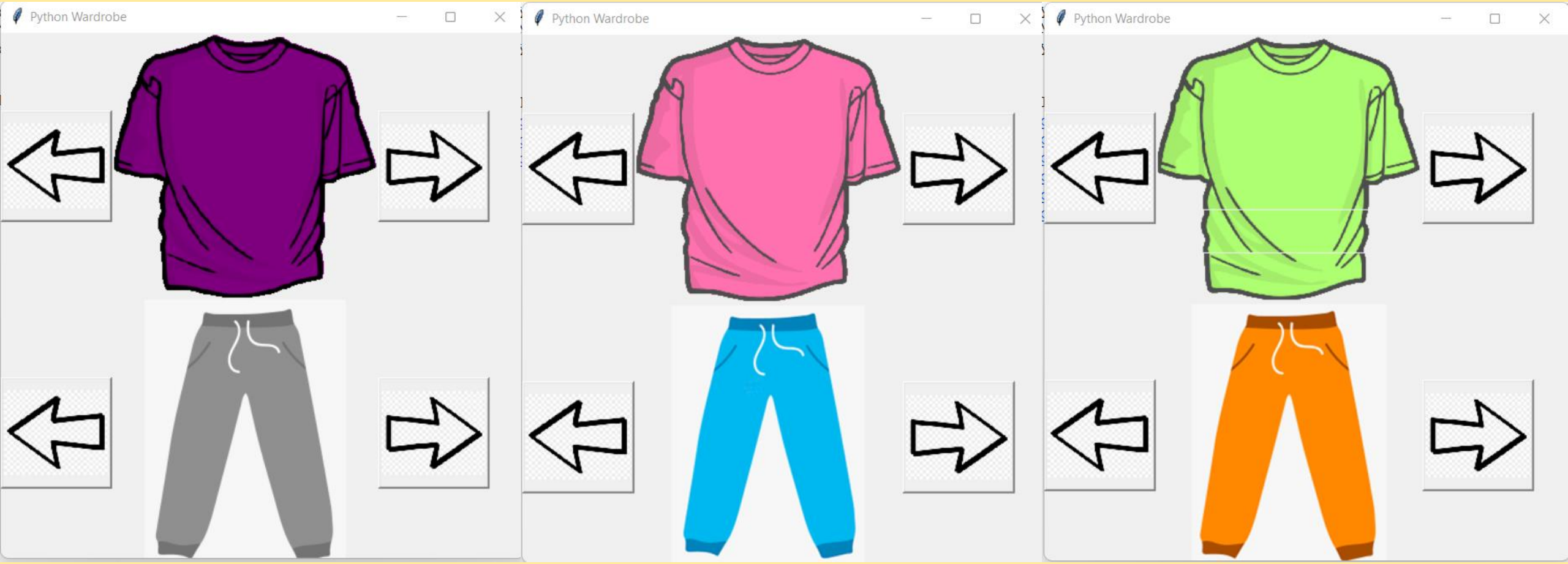
Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers



What the code will look like…



Conclusion

This program should:

- ✓You should confidently be able to import the Tkinter graphics library,
- ✓You should be able to create subroutines with processes for the buttons,
- ✓You should successfully be able to import photos from the folder ‘images’.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

Congratulations!

You have created your own custom wardrobe

Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers