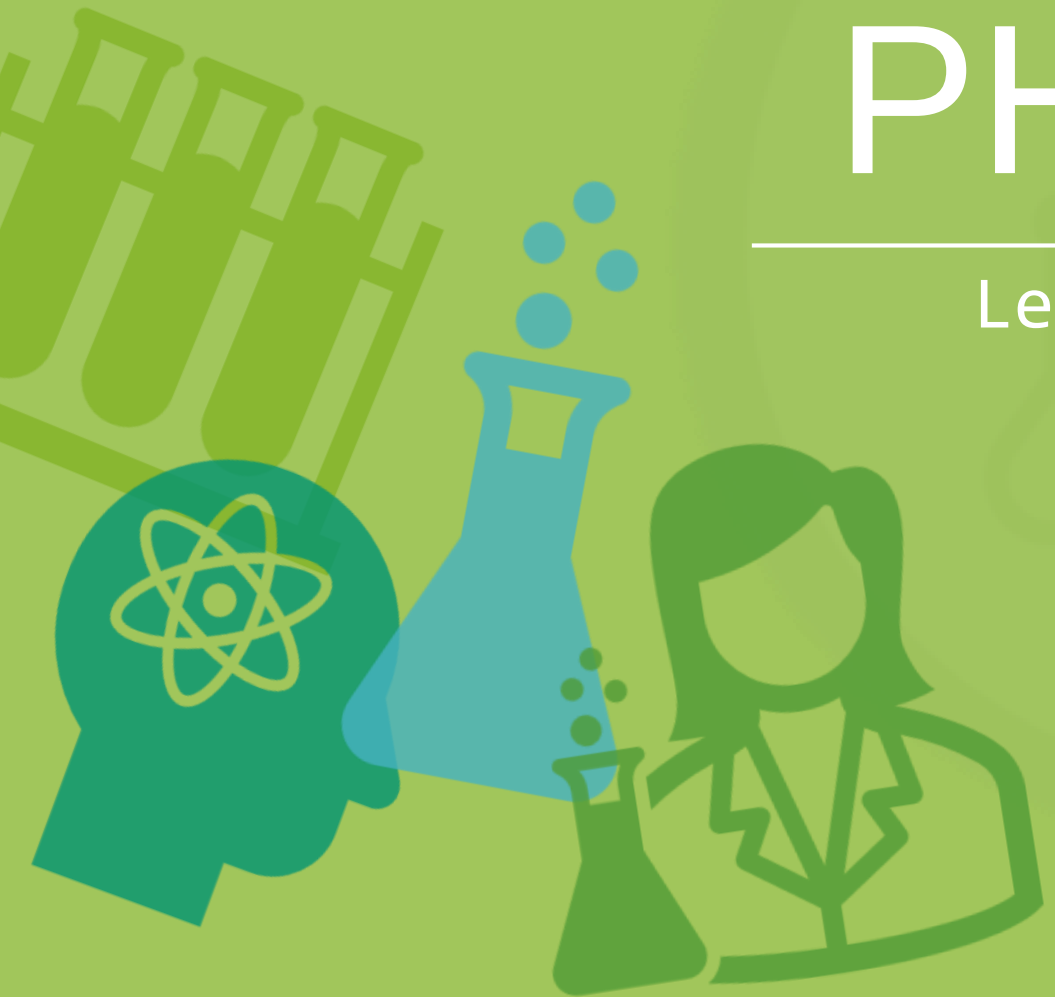
# Python & Java {4} Teachers

# PH Scale

Level 1- Python



Funded by the Erasmus+ Programme of the European Union

# Introduction

In science, chemistry especially, scientists tend to work with many different chemicals. In order to understand how acidic a substance is, these chemicals are measured with something known as a pH scale.

The pH of a substance can either be measured using a pH meter which gives a numerical value, or it can be measured using an indicator and comparing the colour with a pH scale.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# The pH Scale

- The pH scale is a number scale usually from 0 to 14. It tells us how acidic or alkaline an aqueous solution is.
- An aqueous solution is a solution that is made by dissolving a substance into water.
- You can either classify a solution as acidic, neutral or alkaline depending on which part of the scale it ends up on.
  - Solutions that are neutral have a pH of 7,
  - Solutions that have pH values less than 7 are known as acidic. The smaller the value is, the more acidic it is,
  - Solutions that have pH values more than 7 are known as alkaline. The larger the value is, the more alkaline it is.

Funded by the Erasmus+ Programme of the European Union



# Universal indicator

- The universal indicator can show us how strongly acidic or alkaline a solution is accurately using a pH scale.
- Universal indicator has many different colour changes ranging from red for acids, green for neutral and blue/purple for alkaline.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Task

- ✓Import the python library turtle to draw the graphics for the pH scale.
- ✓Choose the correct colours for the pH scale.
- ✓Create the scale and identify which everyday items would go on each colour.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# 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 turtle library

The Python library for turtle has been imported into the program. This means that we can now create graphics using python.

```
1 #PH scale drawing
2 #Importing Turtle library
3 from turtle import *
4 #Changes the shape of the cursor
5 shape("turtle")
6 #Changes the speed of the turtle
7 delay(0)
8 #Changes the thickness of the pen
9 pensize(3)
```

The shape of the cursor has also been changed into the shape of a turtle, and the delay has been set to zero. This means that when the program is run the turtle draws any lines at a much faster pace than normal with the pen at the size 3.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# 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
```

# Loops

Funded by the Erasmus+ Programme of the European Union

A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

In Python there are two main loops: ‘WHILE Loops’ and ‘FOR Loops’

While Loops are condition controlled and will repeat until their condition is false.

```
1
2  condition = True
3  while condition:
4      print("Repeating...")
5
6      print("Finish loop?")
7      finished = input()
8
9
10     if finished == "Y":
11         condition = False
12
```

```
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
Y
```

For loops are count controlled and will repeat a set number of times.

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

# Step 2

## Defining a subroutine

This is the first subroutine that is defined in the pH scale program. This subroutine creates the scale that each colour will sit inside.

```
 8 def scale():
 9     for loop in range(2):
10         fd(1200)
11         rt(90)
12         fd(200)
13         rt(90)
```

```
171 scale()
```

To define a subroutine we will use the def() function. This subroutine will only run when called later on in the program and by looking at the line numbers we can tell that there will be a lot more code written before the subroutine is called.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Colour Gallery

Funded by the Erasmus+ Programme of the European Union

| Red colors | | | |
|---|---|---|---|
| IndianRed | CD 5C 5C | 205 92 92 |
| LightCoral | F0 80 80 | 240 128 128 |
| Salmon | FA 80 72 | 250 128 114 |
| DarkSalmon | E9 96 7A | 233 150 122 |
| LightSalmon | FF A0 7A | 255 160 122 |
| Crimson | DC 14 3C | 220 20 60 |
| Red | FF 00 00 | 255 0 0 |
| FireBrick | B2 22 22 | 178 34 34 |
| DarkRed | 8B 00 00 | 139 0 0 |
| **Pink colors** | | |
| Pink | FF C0 CB | 255 192 203 |
| LightPink | FF B6 C1 | 255 182 193 |
| HotPink | FF 69 B4 | 255 105 180 |
| DeepPink | FF 14 93 | 255 20 147 |
| MediumVioletRed | C7 15 85 | 199 21 133 |
| PaleVioletRed | DB 70 93 | 219 112 147 |
| **Orange colors** | | |
| LightSalmon | FF A0 7A | 255 160 122 |
| Coral | FF 7F 50 | 255 127 80 |
| Tomato | FF 63 47 | 255 99 71 |
| OrangeRed | FF 45 00 | 255 69 0 |
| DarkOrange | FF 8C 00 | 255 140 0 |
| Orange | FF A5 00 | 255 165 0 |

| Yellow colors | | |
|---|---|---|
| Gold | FF D7 00 | 255 215 0 |
| Yellow | FF FF 00 | 255 255 0 |
| LightYellow | FF FF E0 | 255 255 224 |
| LemonChiffon | FF FA CD | 255 250 205 |
| LightGoldenrodYellow | FA FA D2 | 250 250 210 |
| PapayaWhip | FF EF D5 | 255 239 213 |
| Moccasin | FF E4 B5 | 255 228 181 |
| PeachPuff | FF DA B9 | 255 218 185 |
| PaleGoldenrod | EE E8 AA | 238 232 170 |
| Khaki | F0 E6 8C | 240 230 140 |
| DarkKhaki | BD B7 6B | 189 183 107 |
| **Purple colors** | | |
| Lavender | E6 E6 FA | 230 230 250 |
| Thistle | D8 BF D8 | 216 191 216 |
| Plum | DD A0 DD | 221 160 221 |
| Violet | EE 82 EE | 238 130 238 |
| Orchid | DA 70 D6 | 218 112 214 |
| Fuchsia | FF 00 FF | 255 0 255 |
| Magenta | FF 00 FF | 255 0 255 |
| MediumOrchid | BA 55 D3 | 186 85 211 |
| BlueViolet | 8A 2B E2 | 138 43 226 |
| DarkViolet | 94 00 D3 | 148 0 211 |
| DarkOrchid | 99 32 CC | 153 50 204 |
| DarkMagenta | 8B 00 8B | 139 0 139 |
| Purple | 80 00 80 | 128 0 128 |
| Indigo | 4B 00 82 | 75 0 130 |
| SlateBlue | 6A 5A CD | 106 90 205 |
| DarkSlateBlue | 48 3D 8B | 72 61 139 |
| MediumSlateBlue | 7B 68 EE | 123 104 238 |

Python has a wide selection of colours that many don't even realise are available.

| Brown colors | | |
|---|---|---|
| Cornsilk | FF F8 DC | 255 248 220 |
| BlanchedAlmond | FF EB CD | 255 235 205 |
| Bisque | FF E4 C4 | 255 228 196 |
| NavajoWhite | FF DE AD | 255 222 173 |
| Wheat | F5 DE B3 | 245 222 179 |
| BurlyWood | DE B8 87 | 222 184 135 |
| Tan | D2 B4 8C | 210 180 140 |
| RosyBrown | BC 8F 8F | 188 143 143 |
| SandyBrown | F4 A4 60 | 244 164 96 |
| Goldenrod | DA A5 20 | 218 165 32 |
| DarkGoldenrod | B8 86 0B | 184 134 11 |
| Peru | CD 85 3F | 205 133 63 |
| Chocolate | D2 69 1E | 210 105 30 |
| SaddleBrown | 8B 45 13 | 139 69 19 |
| Sienna | A0 52 2D | 160 82 45 |
| Brown | A5 2A 2A | 165 42 42 |
| Maroon | 80 00 00 | 128 0 0 |

Here is a list of potential colours Python allows you to use including the hex and RGB values.

# Colour Gallery

Funded by the Erasmus+ Programme of the European Union

| Green colors | | | |
|---|---|---|---|
| GreenYellow | AD FF 2F | 173 255 47 |
| Chartreuse | 7F FF 00 | 127 255 0 |
| LawnGreen | 7C FC 00 | 124 252 0 |
| Lime | 00 FF 00 | 0 255 0 |
| LimeGreen | 32 CD 32 | 50 205 50 |
| PaleGreen | 98 FB 98 | 152 251 152 |
| LightGreen | 90 EE 90 | 144 238 144 |
| MediumSpringGreen | 00 FA 9A | 0 250 154 |
| SpringGreen | 00 FF 7F | 0 255 127 |
| MediumSeaGreen | 3C B3 71 | 60 179 113 |
| SeaGreen | 2E 8B 57 | 46 139 87 |
| ForestGreen | 22 8B 22 | 34 139 34 |
| Green | 00 80 00 | 0 128 0 |
| DarkGreen | 00 64 00 | 0 100 0 |
| YellowGreen | 9A CD 32 | 154 205 50 |
| OliveDrab | 6B 8E 23 | 107 142 35 |
| Olive | 80 80 00 | 128 128 0 |
| DarkOliveGreen | 55 6B 2F | 85 107 47 |
| MediumAquamarine | 66 CD AA | 102 205 170 |
| DarkSeaGreen | 8F BC 8F | 143 188 143 |
| LightSeaGreen | 20 B2 AA | 32 178 170 |
| DarkCyan | 00 8B 8B | 0 139 139 |
| Teal | 00 80 80 | 0 128 128 |

| Blue/Cyan colors | | |
|---|---|---|
| Aqua | 00 FF FF | 0 255 255 |
| Cyan | 00 FF FF | 0 255 255 |
| LightCyan | E0 FF FF | 224 255 255 |
| PaleTurquoise | AF EE EE | 175 238 238 |
| Aquamarine | 7F FF D4 | 127 255 212 |
| Turquoise | 40 E0 D0 | 64 224 208 |
| MediumTurquoise | 48 D1 CC | 72 209 204 |
| DarkTurquoise | 00 CE D1 | 0 206 209 |
| CadetBlue | 5F 9E A0 | 95 158 160 |
| SteelBlue | 46 82 B4 | 70 130 180 |
| LightSteelBlue | B0 C4 DE | 176 196 222 |
| PowderBlue | B0 E0 E6 | 176 224 230 |
| LightBlue | AD D8 E6 | 173 216 230 |
| SkyBlue | 87 CE EB | 135 206 235 |
| LightSkyBlue | 87 CE FA | 135 206 250 |
| DeepSkyBlue | 00 BF FF | 0 191 255 |
| DodgerBlue | 1E 90 FF | 30 144 255 |
| CornflowerBlue | 64 95 ED | 100 149 237 |
| MediumSlateBlue | 7B 68 EE | 123 104 238 |
| RoyalBlue | 41 69 E1 | 65 105 225 |
| MediumBlue | 00 00 CD | 0 0 205 |
| DarkBlue | 00 00 8B | 0 0 139 |
| Navy | 00 00 80 | 0 0 128 |
| MidnightBlue | 19 19 70 | 25 25 112 |

| White colors | | |
|---|---|---|
| White | FF FF FF | 255 255 255 |
| Snow | FF FA FA | 255 250 250 |
| Honeydew | F0 FF F0 | 240 255 240 |
| MintCream | F5 FF FA | 245 255 250 |
| Azure | F0 FF FF | 240 255 255 |
| AliceBlue | F0 F8 FF | 240 248 255 |
| GhostWhite | F8 F8 FF | 248 248 255 |
| WhiteSmoke | F5 F5 F5 | 245 245 245 |
| Seashell | FF F5 EE | 255 245 238 |
| Beige | F5 F5 DC | 245 245 220 |
| OldLace | FD F5 E6 | 253 245 230 |
| FloralWhite | FF FA F0 | 255 250 240 |
| Ivory | FF FF F0 | 255 255 240 |
| AntiqueWhite | FA EB D7 | 250 235 215 |
| Linen | FA F0 E6 | 250 240 230 |
| LavenderBlush | FF F0 F5 | 255 240 245 |
| MistyRose | FF E4 E1 | 255 228 225 |

| Gray colors | | |
|---|---|---|
| Gainsboro | DC DC DC | 220 220 220 |
| LightGrey | D3 D3 D3 | 211 211 211 |
| Silver | C0 C0 C0 | 192 192 192 |
| DarkGray | A9 A9 A9 | 169 169 169 |
| Gray | 80 80 80 | 128 128 128 |
| DimGray | 69 69 69 | 105 105 105 |
| LightSlateGray | 77 88 99 | 119 136 153 |
| SlateGray | 70 80 90 | 112 128 144 |
| Black | 00 00 00 | 0 0 0 |

# Step 3

## Creating a subroutine

This is the code for the first colour block as a subroutine. As this is a pH scale we will start with the colour the universal indicator will go when in contact with a very strong acid.

```
14 def colour0():
15     pu()
16     fillcolor("Red")
17     begin_fill()
18     for loop in range(2):
19         fd(80)
20         rt(90)
21         fd(200)
22         rt(90)
23     end_fill()
```

This subroutine is defined under the name ‘colour1’. The pen goes up to avoid getting a border line and the fill colour is set to ‘Red’. The for loop is then set to create a rectangular block for the colour to sit in.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 4

## Creating a subroutine

For the pH scale the acids become weaker and weaker before being neutral. This is why the colour will become increasingly lighter as we move through the colour block subroutines.

```
24 def colour1():
25     pu()
26     fillcolor("OrangeRed")
27     begin_fill()
28     for loop in range(2):
29         fd(80)
30         rt(90)
31         fd(200)
32         rt(90)
33     end_fill()
```

***Remember:*** Subroutines only run when they are called in the program not just when they are defined. There will be some extra code between the definition of the subroutine and us calling it, that is why the block looks like it has moved perfectly without us actually coding the turtles movements yet!

The subroutine stays the same as the previous blocks code the only thing that needs to change here is the colour of the block itself to orange.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 5

## Creating a Subroutine

The subroutine has been adjusted again for the colour ‘orange’ to be drawn in the graphic.

```
34 def colour2():
35     pu()
36     fillcolor("Orange")
37     begin_fill()
38     for loop in range(2):
39         fd(80)
40         rt(90)
41         fd(200)
42         rt(90)
43     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 6

## Creating the colour block subroutine

The subroutine itself stays the same for each colour block that is coded. The only part that will change is the colour being used.

```
44 def colour3():
45     pu()
46     fillcolor("Gold")
47     begin_fill()
48     for loop in range(2):
49         fd(80)
50         rt(90)
51         fd(200)
52         rt(90)
53     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 7

## Creating the colour block subroutine

The colours are now reaching yellow which translates a weaker acid of about pH 3. An example for this pH would be Orange Juice.

```
54 def colour4():
55     pu()
56     fillcolor("Yellow")
57     begin_fill()
58     for loop in range(2):
59         fd(80)
60         rt(90)
61         fd(200)
62         rt(90)
63     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 8

## Creating the colour block subroutine

The colour has been adjusted for ‘GreenYellow’, which begins filling the colour just before the for loop is used. Line 73 then means that the shape is finished filling-in colour and the rectangular block will become one block colour.

```
64 def colour5():
65     pu()
66     fillcolor("GreenYellow")
67     begin_fill()
68     for loop in range(2):
69         fd(80)
70         rt(90)
71         fd(200)
72         rt(90)
73     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 9

## Creating the colour block subroutine

The pH scale is now reaching colours like lime green which would be approximately pH 6.

```
74 def colour6():
75     pu()
76     fillcolor("LimeGreen")
77     begin_fill()
78     for loop in range(2):
79         fd(80)
80         rt(90)
81         fd(200)
82         rt(90)
83     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 10

## Creating the colour block subroutine

Each colour block has the same dimensions so that when each subroutine is called the blocks are all even.

```
84 def colour7():
85     pu()
86     fillcolor("ForestGreen")
87     begin_fill()
88     for loop in range(2):
89         fd(80)
90         rt(90)
91         fd(200)
92         rt(90)
93     end_fill()
```

This subroutine is for the equivalent of pH 7 which is completely neutral, the best example of a neutral substance is pure water.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 11

## Creating the colour block subroutine

This subroutine has been altered for the colour ‘SeaGreen’.

```
 94 def colour8():
 95     pu()
 96     fillcolor("SeaGreen")
 97     begin_fill()
 98     for loop in range(2):
 99         fd(80)
100         rt(90)
101         fd(200)
102         rt(90)
103     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 12

## Creating the colour block subroutine

The pH scale has now moved n to the colour palette for the alkalis, with dark greens and lighter blues being weaker alkalis.

```
104 def colour9():
105     pu()
106     fillcolor("LightSeaGreen")
107     begin_fill()
108     for loop in range(2):
109         fd(80)
110         rt(90)
111         fd(200)
112         rt(90)
113     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 13

## Creating the colour block subroutine

This subroutine has been defined for the colour ‘SteelBlue’.

```
114 def colour10():
115     pu()
116     fillcolor("SteelBlue")
117     begin_fill()
118     for loop in range(2):
119         fd(80)
120         rt(90)
121         fd(200)
122         rt(90)
123     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 14

## Creating the colour block subroutine

The colours are starting to get darker, which means the alkalis will be getting stronger at this point on the scale.

```
124 def colour11():
125     pu()
126     fillcolor("RoyalBlue")
127     begin_fill()
128     for loop in range(2):
129         fd(80)
130         rt(90)
131         fd(200)
132         rt(90)
133     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 15

## Creating the colour block subroutine

This subroutine has been adapted for the colour block ‘DarkViolet’.

```
134 def colour12():
135     pu()
136     fillcolor("DarkViolet")
137     begin_fill()
138     for loop in range(2):
139         fd(80)
140         rt(90)
141         fd(200)
142         rt(90)
143     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 16

## Creating the colour block subroutine

The subroutine is defined using the def function followed by the rest of the code set in an idented block. For this subroutine thet colour has been changed to ‘DarkMagenta’.

```
144 def colour13():
145     pu()
146     fillcolor("DarkMagenta")
147     begin_fill()
148     for loop in range(2):
149         fd(80)
150         rt(90)
151         fd(200)
152         rt(90)
153     end_fill()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 17

## Creating the colour block subroutine

This is the last colour block subroutine that is needed for the pH scale. This ‘Indigo’ colour would represent the stronger alkalis such as drain cleaner.

```
154 def colour14():
155     pu()
156     fillcolor("Indigo")
157     begin_fill()
158     for loop in range(2):
159         fd(80)
160         rt(90)
161         fd(200)
162         rt(90)
163     end_fill()
```

Now that all of the subroutines have been defined we can now call them into the program so that the graphics can actually be drawn.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 18

## Setting the Turtle

If you try to run the program with just the subroutines that have been created, the program should show up blank. This is because we have yet to actually call these subroutines to draw the graphics we have coded.

```
164 #Main Code
165 #Moving the turtle so that the scale fits properly
166 pu()
167 lt(180)
168 fd(600)
169 rt(180)
170 pd()
```

The code here moves with the ‘pen up’, so that when we move the turtle initially no unnecessary lines are drawn. These instructions move the turtle to a point, so when we start drawing the pH scale it shouldn’t get cut off the corners of the screen.
The pen is then ‘put down’ so that when we call the next few lines of code, the lines show up.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Step 19

## Calling the subroutines

Here we call the subroutine to draw the outline of the pH scale and then the first colour block.

Between each colour block being called the turtle is moved forward by 80 pixels. This is so when the next colour is added it joins the last block not just colours over it instead.

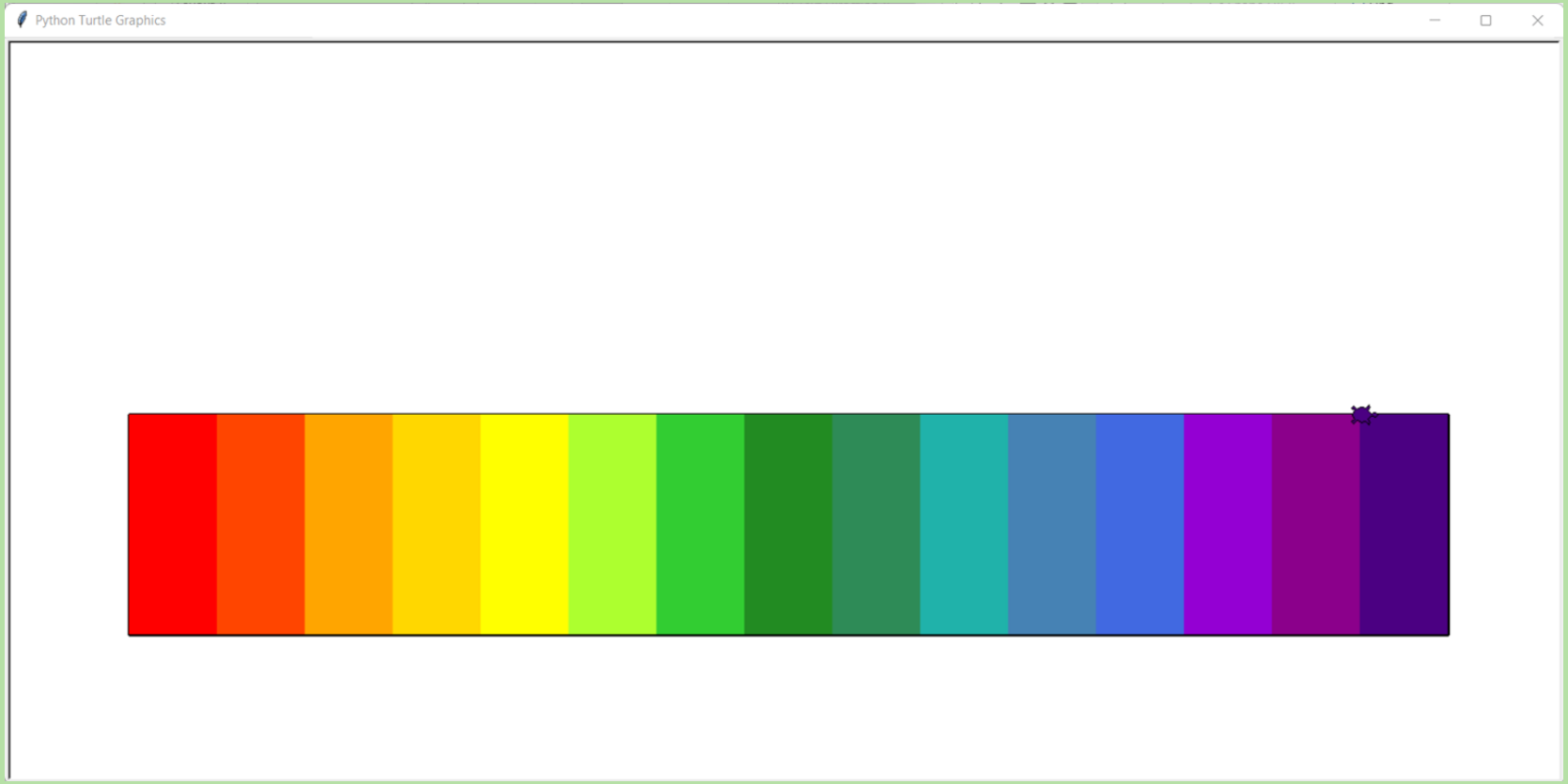
```
171 scale()
172 colour0()
173 fd(80)
174 colour1()
175 fd(80)
176 colour2()
177 fd(80)
178 colour3()
179 fd(80)
180 colour4()
181 fd(80)
182 colour5()
183 fd(80)
184 colour6()
185 fd(80)
186 colour7()
187 fd(80)
188 colour8()
189 fd(80)
190 colour9()
191 fd(80)
192 colour10()
193 fd(80)
194 colour11()
195 fd(80)
196 colour12()
197 fd(80)
198 colour13()
199 fd(80)
200 colour14()
```

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# What the code will look like…




Funded by the Erasmus+ Programme of the European Union

Python & Java 4 Teachers

# Extension

**Now that you have created the pH scale try to find different examples for each pH!**

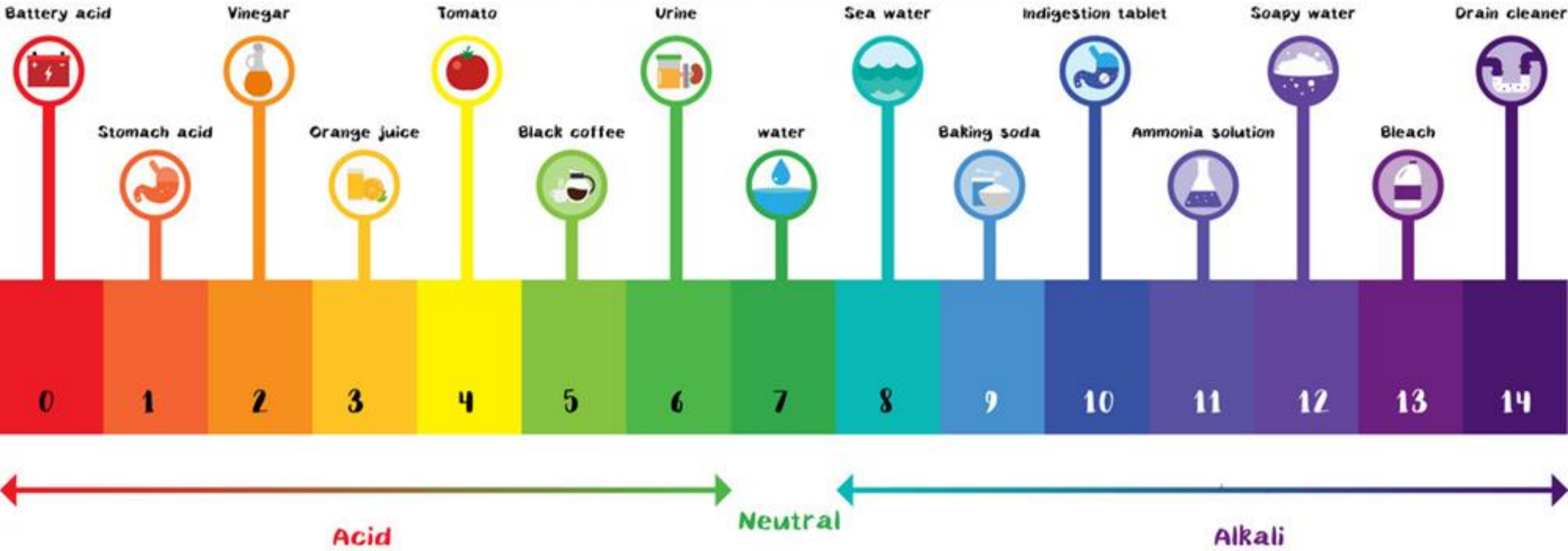

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# THE pH SCALE

## Universal Indicator pH Color Chart



# Conclusion

**Learning Outcomes:**

- ✓I can confidently import the python library for turtle graphics,
- ✓I can define and call subroutines,
- ✓I can positively use for loops,
- ✓I am able to understand python’s colour library and choose appropriate colour's for the pH scale.

Funded by the Erasmus+ Programme of the European Union

{4}

Python & Java 4 Teachers

# Congratulations!

You have created your very own PH scale!

Funded by the Erasmus+ Programme of the European Union