Lab #4

TU850/1 - Creative Coding

*You are welcome to submit this lab before Monday at 12 noon (and there’s no need to come into the lab), or during the lab before 2pm.*

**QUESTION 1**

Python has a thousands of libraries that can be categorized into two main types:

* Built-in libraries.
* Third-party libraries.

**Built-in libraries** come pre-installed with Python. These libraries provide essential functions, such as working with files, handling system operations, performing mathematical calculations, and managing dates and times. The Python Standard Library **contains over 200 libraries**, though the exact number may vary depending on the Python version.

**Third-party libraries** are not included in the default Python installation and must be installed separately. These libraries extend Python’s capabilities and are widely used in specialized fields such as data science, machine learning, web development, automation, and game development. For example, libraries like numpy and pandas are essential for data analysis. There are currently **over 450,000 third-party libraries**, allowing developers to find tools for almost any task.

The key difference between these two types of libraries is that built-in libraries are always available without additional installation, whereas third-party libraries need to be installed separately.

So if I write code like the following:

Import numpy

And run it, it might give me an error saying:

ModuleNotFoundError: No module named 'numpy'

So let’s do it, go into PyCharm, and click on the Hamburger symbol in the top left corner:

A white lines on a purple background

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

You should get a menu:

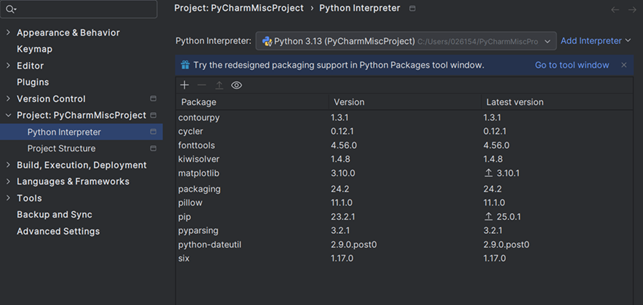


Go to File > Settings

A screenshot of a computer menu

AI-generated content may be incorrect.

You should get a window like this:



Click the + (Add) button at the top right.

In the search bar, type in the package name you are looking for.

Select numpy from the list and click Install Package.

Wait for the installation to complete.

Now run this code:

|  |  |
| --- | --- |
| Python Logo | **Python** |
| import numpy as np import tkinter as tk from tkinter import scrolledtext  def display\_output():  # Create a 5x5 NumPy array with values from 1 to 25  arr = np.arange(1, 26).reshape(5, 5)   # Perform various slicing operations  submatrix = arr[1:4, 1:4]  every\_other\_row = arr[::2, :]  last\_column = arr[:, -1]  row\_reversed = arr[::-1, :]  column\_reversed = arr[:, ::-1]  main\_diagonal = np.diag(arr)   # Prepare output as a string  output\_text = (  "Original 5x5 Array:\n" + str(arr) + "\n\n"  "3x3 Center Submatrix:\n" + str(submatrix) + "\n\n"  "Every Other Row:\n" + str(every\_other\_row) + "\n\n"  "Last Column:\n" + str(last\_column) + "\n\n"  "Row-Reversed Array:\n" + str(row\_reversed) + "\n\n"  "Column-Reversed Array:\n" + str(column\_reversed) + "\n\n"  "Main Diagonal Elements:\n" + str(main\_diagonal) + "\n"  )   # Create a new Tkinter window  root = tk.Tk()  root.title("NumPy Array Slicing Output")   # Create a scrollable text widget  text\_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=50, height=20)  text\_area.insert(tk.INSERT, output\_text)  text\_area.config(state=tk.DISABLED) # Make it read-only  text\_area.pack(padx=10, pady=10)   # Run the Tkinter event loop  root.mainloop()  # Call function to display output in a new window display\_output() | |

All you have to do is write a one-line explanation of what each array represents.

| **Operation** | **Explanation** |
| --- | --- |
| ***3×3 Center Submatrix*** |  |
| ***Every Other Row*** |  |
| ***Last Column*** |  |
| ***Row-Reversed Array*** |  |
| ***Column-Reversed Array*** |  |
| ***Main Diagonal Elements*** |  |

**QUESTION 2**

**Instructions:**

Take the following code and run it, maximise the resultant window:

|  |  |
| --- | --- |
| Python Logo | **Python** |
| import tkinter as tk import random import math import time from textwrap import fill  # Relaxing & Fun Activities activities = [  "Listen to Music", "Watch a Movie or Show", "Go for a Walk",  "Try a New Recipe", "Read a Book", "Do Some Doodling or Art",  "Play a Game", "Stretch or Meditate" ]  # Colors for sections colors = ["red", "orange", "yellow", "green", "blue", "indigo", "violet", "pink"]  # Set up window root = tk.Tk() root.title("Relax & Have Fun Spinner") root.geometry("500x550")  canvas = tk.Canvas(root, width=500, height=500, bg="white") canvas.pack()  # Spinner properties center\_x, center\_y = 250, 250 radius = 180 num\_sections = len(activities)  def wrap\_text(text, max\_width=10):  *"""Wrap text to fit within the segment width."""* return fill(text, width=max\_width)  def draw\_spinner(highlight\_index=None):  *"""Draws the spinner with text fitting neatly inside segments."""* canvas.delete("all")   for i in range(num\_sections):  angle1 = (2 \* math.pi \* i) / num\_sections  angle2 = (2 \* math.pi \* (i + 1)) / num\_sections  x1, y1 = center\_x + radius \* math.cos(angle1), center\_y + radius \* math.sin(angle1)  x2, y2 = center\_x + radius \* math.cos(angle2), center\_y + radius \* math.sin(angle2)   # Highlight selected section  fill\_color = colors[i] if i == highlight\_index else "white"   # Draw the pie section  canvas.create\_polygon(center\_x, center\_y, x1, y1, x2, y2, fill=fill\_color, outline="black")   # Calculate text position (closer to center)  text\_angle = (angle1 + angle2) / 2  text\_x = center\_x + (radius / 2) \* math.cos(text\_angle)  text\_y = center\_y + (radius / 2) \* math.sin(text\_angle)  text\_rotation = math.degrees(text\_angle)   # Flip text if upside down  if 90 < text\_rotation < 270:  text\_rotation += 180   # Wrap text and adjust font size dynamically  wrapped\_text = wrap\_text(activities[i])  canvas.create\_text(  text\_x, text\_y, text=wrapped\_text, fill="black", font=("Arial", 9, "bold"),  angle=text\_rotation, anchor="center"  )   # Draw outer circle  canvas.create\_oval(  center\_x - radius, center\_y - radius, center\_x + radius, center\_y + radius, outline="black", width=2  )   # Draw center dot  canvas.create\_oval(center\_x - 5, center\_y - 5, center\_x + 5, center\_y + 5, fill="black")  def spin():  *"""Simulates spinning animation and selects a random activity."""* for \_ in range(30): # Animation loop  random\_index = random.randint(0, num\_sections - 1)  draw\_spinner(highlight\_index=random\_index)  root.update()  time.sleep(0.05)   # Final selected activity  selected\_activity = activities[random\_index]  result\_label.config(text=f"Try: {selected\_activity}")  # Button and result label spin\_button = tk.Button(root, text="Spin!", command=spin, font=("Arial", 14)) spin\_button.pack(pady=10)  result\_label = tk.Label(root, text="", font=("Arial", 12, "bold")) result\_label.pack(pady=10)  draw\_spinner() # Initial draw root.mainloop() | |

Change the text each of the wedges to be options for anything you want, e.g.

* Different apps
* Different TV shows
* Different meals
* Different books
* etc.

Do a screengrab of the new spinner.

**QUESTION 3**

**Instructions:**

Open up a different program, “Processing 4”, run the following code, and describe what you see in 25-50 words.

|  |  |
| --- | --- |
| Processing 4 | **Processing 4** |
| float angle = 0;  float wobbleSpeed = 0.05; // Speed of the wobble  float wobbleAmount = 20; // Height range of the wobble  void setup() {  size(800, 600, P3D);  smooth(8);  }  void draw() {  background(0);  lights();    float wobble = sin(frameCount \* wobbleSpeed) \* wobbleAmount; // Up-down movement    translate(width / 2, height / 2 + wobble, -200); // Apply wobble effect  rotateX(PI / 8); // Slight tilt for depth  rotateY(angle); // Spinning effect  drawSaucer();  drawRadialLines();  drawLandingLegs();  angle += 0.02; // Smooth rotation speed  }  void drawSaucer() {  pushMatrix();  fill(180, 180, 255);  noStroke();  scale(2.5, 0.3, 2.5);  sphere(100);  popMatrix();  }  void drawRadialLines() {  int numLines = 24;  float radius = 250;  pushMatrix();  stroke(0);  strokeWeight(2);  for (int i = 0; i < numLines; i++) {  float angle = TWO\_PI \* i / numLines;  float x = cos(angle) \* radius;  float z = sin(angle) \* radius;  line(0, 0, 0, x, 0, z);  }  popMatrix();  }  void drawLandingLegs() {  int numLegs = 3;  float legRadius = 10;  float legHeight = 80;  float legOffset = 120; // Distance from center  float legY = 70; // Position below saucer  float legTilt = radians(20); // Angle for outward tilt  for (int i = 0; i < numLegs; i++) {  float angle = TWO\_PI \* i / numLegs;  float x = cos(angle) \* legOffset;  float z = sin(angle) \* legOffset;  pushMatrix();  translate(x, legY, z);  rotateX(sin(angle) \* legTilt); // Tilt outward  rotateZ(-cos(angle) \* legTilt); // Tilt outward  fill(100);  noStroke();  cylinder(legRadius, legHeight);  popMatrix();  // Draw small circular footpad at the bottom of each leg  pushMatrix();  translate(x \* 1.2, legY + legHeight, z \* 1.2); // Adjust position for tilt  fill(80);  noStroke();  cylinder(legRadius \* 1.5, 5); // Flat disc shape for footpad  popMatrix();  }  }  // Helper function to draw a simple cylinder  void cylinder(float r, float h) {  int sides = 20;  float angleStep = TWO\_PI / sides;  beginShape(TRIANGLE\_STRIP);  for (int i = 0; i <= sides; i++) {  float angle = i \* angleStep;  float x = cos(angle) \* r;  float z = sin(angle) \* r;  vertex(x, -h / 2, z);  vertex(x, h / 2, z);  }  endShape(CLOSE);  } | |

**Submission:**

Submit the two screengrabs, as well as the code and the explaination, and paste it all onto your Template document.

|  |
| --- |
| Search in Brightspace for the following module: “**Creative Coding CMPU1042: 2024-25**” and please enroll. |
| e-mail me a completed solution to each of the above programs in your Template document. The Template document should be renamed as follows:   * Surname\_Firstname\_Student#\_\_Lab4.pdf * for example: Smith\_John\_D1234567\_Lab4.pdf   Send it to [Damian.X.Gordon@tudublin.ie](mailto:Damian.X.Gordon@tudublin.ie) with subject heading “DT850 CC Lab #4”, and put it in Brightspace as well. |