# 15차 연습문항

## **파이썬 심화문항: 그래픽스**

|  |
| --- |
| 수강생 공지 사항   * 제출은 프로그래머스를 통해 해주시기 바랍니다([링크](https://campus.programmers.co.kr/app/courses/24550/curriculum)). * 파일명은 아래와 같은 형식으로 제출해주세요   + 교육생번호\_이름\_교과목\_문항\_N차시.pdf   ex) DR-11111\_홍길동\_파이썬\_연습문항\_1차시.pdf  ex) DR-11111\_홍길동\_파이썬\_추가문항\_1차시.pdf   * 답은 "write your answer"에 적어주세요. 다만 코딩 문제의 경우 output까지 답에 포함시켜 주세요.   ex)     * 답은 캡쳐를 하셔도 되고 텍스트로 넣으셔도 됩니다. * 마감 기한은 문제가 나간 주 **일요일 23:59까지**입니다. |

### Numpy는 수치연산을 위한 핵심 라이브러리입니다. Numerical Python의 줄임말로, 다차원 배열을 다루는 기능과 선형 대수 연산과 같은 고수준 수학함수들을 제공합니다. Numpy는 리스트와 비교해 더 빠르고 메모리 효율적인 배열을 제공하며, 대용량 데이터 연산을 보다 효율적으로 수행할 수 있습니다.

아래의 수식이 제공하는 x, y 랜덤데이타를 가지고 산포그래프를 그려보세요.

x = np.random.normal(size=100) # 정규 분포를 따르는 랜덤 데이터 생성

y = 2.5 \* x + np.random.normal(size=100) # 선형 관계를 가지는 랜덤 데이터 생성

**답**

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  # 랜덤 시드 설정 (재현성을 위해)  np.random.seed(0)  # 데이터 생성  x = np.random.normal(size=100) # 정규 분포를 따르는 랜덤 데이터 생성  y = 2.5 \* x + np.random.normal(size=100) # 선형 관계를 가지는 랜덤 데이터 생성#  #산포 그래프 그리기  plt.figure(figsize=(8, 6))  plt.scatter(x, y, color='blue', alpha=0.7)    # 그래프 제목과 레이블 추가  plt.title('Scatter Plot of Random Data')  plt.xlabel('X-axis')  plt.ylabel('Y-axis')    # 그래프 표시  plt.grid(True)plt.show() |

### Tkinter는 파이썬에서 기본적으로 제공하는 GUI(Graphical User Interface) 라이브러리입니다. Tkinter를 사용해서 'Tkinter GUI를 시작하셨습니다.' 라는 메세지를 보여주는 윈도우를 만들어보세요.

**답**

|  |
| --- |
| import tkinter as tk    # Tkinter 윈도우 생성  root = tk.Tk()  root.title("Tkinter GUI")  # 윈도우 제목 설정    # 라벨 추가  label = tk.Label(root, text="Tkinter GUI를 시작하셨습니다.", padx=20, pady=20)  label.pack()  # 윈도우 실행  root.mainloop() |

### GUI 윈도우에 아래와 같은 조건을 실행하는 프로그램을 작성하시오.

- GUI 제목은 피트에서 미터로 변환

- 피트 데이터를 입력받은 후, 하단에 위치한 Calculate버튼을 누르면 미터로 변환하여 윈도우상에 보여줌

**답**

|  |
| --- |
| import tkinter as tk    def feet\_to\_meter():      try:       # Convert feet to meters       feet = float(feet\_entry.get())       meters = feet \* 0.3048         # Update result label       result\_label.config(text=f"{meters:.2f}")       error\_label.config(text="")      except ValueError:       result\_label.config(text="")       error\_label.config(text="숫자를 입력하세요.")  # Create Tkinter window  root = tk.Tk()  root.title("피트에서 미터로 변환")    # Window size and position  window\_width = 300  window\_height = 250  screen\_width = root.winfo\_screenwidth()  screen\_height = root.winfo\_screenheight()  x\_position = int((screen\_width - window\_width) / 2)  y\_position = int((screen\_height - window\_height) / 2)  root.geometry(f"{window\_width}x{window\_height}+{x\_position}+{y\_position}")    # Create a frame to hold all widgets and center it  frame = tk.Frame(root)  frame.pack(expand=True, fill=tk.BOTH)    # Input entry and '피트' label  feet\_entry = tk.Entry(frame, justify="center", width=10, font=("Arial", 16))  feet\_entry.grid(row=0, column=0, padx=50, pady=30)    feet\_label = tk.Label(frame, text="피트", font=("Arial", 16))  feet\_label.grid(row=0, column=1, padx=10, pady=10, sticky=tk.W)    # Calculate button  calculate\_button = tk.Button(frame, text="Calculate", command=feet\_to\_meter, font=("Arial", 16))  calculate\_button.grid(row=2, column=0, columnspan=2, padx=10, pady=10)    # Result label for meters  result\_label = tk.Label(frame, text="", font=("Arial", 16))  result\_label.grid(row=1, column=0, padx=50, pady=10)    meters\_label = tk.Label(frame, text="미터", font=("Arial", 16))  meters\_label.grid(row=1, column=1, padx=10, pady=10, sticky=tk.W)    # Error message label  error\_label = tk.Label(frame, text="", fg="red", font=("Arial", 12))  error\_label.grid(row=3, column=0, columnspan=2, padx=10, pady=10)    # Run the Tkinter main loop  root.mainloop() |

### 문제13에서 피트에서 미터로 일방 변환인 프로그램을 피트나 미터 어느 한쪽에 입력이 들어오면 다른 단위로 변환시키는 프로그램으로 변경하고, 그에 맞춰 제목이나 화면 구성을 수정하시오.

**답**

|  |
| --- |
| import tkinter as tk    def convert\_units():      try:       if feet\_entry.get():                 # Convert feet to meters                 feet = float(feet\_entry.get())                 meters = feet \* 0.3048                 # Update meter entry field                 meter\_entry.delete(0, tk.END)                 meter\_entry.insert(0, f"{meters:.2f}")       elif meter\_entry.get():                 # Convert meters to feet                 meters = float(meter\_entry.get())                 feet = meters / 0.3048                 # Update feet entry field                 feet\_entry.delete(0, tk.END)                 feet\_entry.insert(0, f"{feet:.2f}")       error\_label.config(text="")      except ValueError:       error\_label.config(text="유효한 숫자를 입력하세요.")    def reset\_fields():      feet\_entry.delete(0, tk.END)      meter\_entry.delete(0, tk.END)      error\_label.config(text="")    # Create Tkinter window  root = tk.Tk()  root.title("단위 변환기")    # Window size and position  window\_width = 300  window\_height = 250  screen\_width = root.winfo\_screenwidth()  screen\_height = root.winfo\_screenheight()  x\_position = int((screen\_width - window\_width) / 2)  y\_position = int((screen\_height - window\_height) / 2)  root.geometry(f"{window\_width}x{window\_height}+{x\_position}+{y\_position}")    # Create a frame to hold all widgets and center it  frame = tk.Frame(root)  frame.pack(expand=True, fill=tk.BOTH)  # Input entry and '피트' label  feet\_entry = tk.Entry(frame, justify="center", width=10, font=("Arial", 16))  feet\_entry.grid(row=0, column=0, padx=10, pady=10)  feet\_label = tk.Label(frame, text="피트", font=("Arial", 16))  feet\_label.grid(row=0, column=1, padx=10, pady=10, sticky=tk.W)    # Input entry and '미터' label  meter\_entry = tk.Entry(frame, justify="center", width=10, font=("Arial", 16))  meter\_entry.grid(row=1, column=0, padx=10, pady=10)  meter\_label = tk.Label(frame, text="미터", font=("Arial", 16))  meter\_label.grid(row=1, column=1, padx=10, pady=10, sticky=tk.W)    # Calculate button  calculate\_button = tk.Button(frame, text="Calculate", command=convert\_units, font=("Arial", 16))  calculate\_button.grid(row=2, column=0, padx=10, pady=10)    # Reset button  reset\_button = tk.Button(frame, text="Reset", command=reset\_fields, font=("Arial", 16))  reset\_button.grid(row=2, column=1, padx=10, pady=10)    # Error message label  error\_label = tk.Label(frame, text="", fg="red", font=("Arial", 12))  error\_label.grid(row=3, column=0, columnspan=2, padx=10, pady=10)    # Run the Tkinter main loop  root.mainloop() |

### 문제9의 거북이는 윈도우를 벗어나면 보이지 않는다. 거북이가 윈도우를 벗어나면 반대편으로 나오도록 프로그램을 보완해 보자.

**답**

|  |
| --- |
| import turtle as t    # 터틀 객체 생성  turtle = t.Turtle()    # 화면 경계 설정  screen = t.Screen()  screen\_width = screen.window\_width() // 2  screen\_height = screen.window\_height() // 2    # 이동 중인지를 나타내는 변수  moving = False    # 이동 함수 정의  def move\_forward():      global moving      if not moving:       moving = True       x, y = turtle.position()       if is\_out\_of\_bounds(x, y):                 teleport\_to\_opposite\_side()       turtle.forward(5)       moving = False  def move\_backward():      global moving      if not moving:       moving = True        x, y = turtle.position()       if is\_out\_of\_bounds(x, y):                 teleport\_to\_opposite\_side()       turtle.backward(5)       moving = False    def move\_right():      global moving      if not moving:       moving = True       x, y = turtle.position()       if is\_out\_of\_bounds(x, y):                 teleport\_to\_opposite\_side()               turtle.right(5)       turtle.forward(5)       moving = False    def move\_left():     global moving      if not moving:       moving = True        x, y = turtle.position()       if is\_out\_of\_bounds(x, y):                  teleport\_to\_opposite\_side()       turtle.left(5)               turtle.forward(5)               moving = False    # 화면 경계를 벗어났는지 확인하는 함수  def is\_out\_of\_bounds(x, y):     return x >screen\_width or x <-screen\_width or y >screen\_height or y <-screen\_height    # 반대편 벽으로 이동하는 함수  def teleport\_to\_opposite\_side():      x, y = turtle.position()     if x >screen\_width:       turtle.penup()       turtle.setx(-screen\_width)       turtle.pendown()      elif x <-screen\_width:               turtle.penup()       turtle.setx(screen\_width)       turtle.pendown()      if y >screen\_height:       turtle.penup()               turtle.sety(-screen\_height)       turtle.pendown()      elif y <-screen\_height:                turtle.penup()                turtle.sety(screen\_height)                turtle.pendown()    # 초기 설정  turtle.speed(0)  turtle.shape('turtle')    # 키보드 이벤트 처리  screen.onkeypress(move\_forward, "Up")  screen.onkeypress(move\_backward, "Down")  screen.onkeypress(move\_right, "Right")  screen.onkeypress(move\_left, "Left")    # 이벤트 리스너 활성화  screen.listen()    # 터틀 그래픽 창을 닫을 때 종료하기 위해 대기  screen.mainloop() |

### Pygame은 파이썬으로 게임을 개발할 때 유용하게 사용되는 크로스 플랫폼 게임 개발 라이브러리입니다. 주로 2D 게임 개발에 많이 사용되며, 다양한 멀티미디어 처리 기능을 제공하여 게임 개발자들이 게임을 만드는 데 필요한 여러 기능을 간편하게 구현할 수 있도록 돕습니다. Pygame을 사용하여 윈도우를 생성하고 배경색 을 설정하는 프로그램을 작성해 보세요.

- Pygame을 설치하려면 명령 프롬프트나 터미널에서 다음 명령을 실행합니다.

. python -m pip install --upgrade pip

. pip install pygame

**답**

|  |
| --- |
| import pygame  import sys  # 초기화  pygame.init()  # 화면 설정  screen\_width = 800  screen\_height = 600  screen = pygame.display.set\_mode((screen\_width, screen\_height))  pygame.display.set\_caption("Pygame Example")  # 색깔 정의  WHITE = (255, 255, 255)  # 게임 루프  running = True  while running:      # 이벤트 처리      for event in pygame.event.get():       if event.type == pygame.QUIT:             running = False      # 화면 업데이트     screen.fill(WHITE)     pygame.display.flip()  # 종료  pygame.quit()  sys.exit() |

### 생성된 윈도우안에서 공이 움직이다가 벽을 만나면 튀어 나가는 프로그램을 작성해 봅니다.

- 공이 화면 중앙에 멈춰 있다가 아무 키나 입력되면 움직이기 시작함.

- 장애물(벽)을 만나면 입사각과 같은 각도로 반사됨

- 게임엔진은 그래픽 처리와 물리 시뮬레이션, 게임 루프등과 같은 일반 소프트웨어하고 다른 구조적 차이가 있음

**답**

|  |
| --- |
| import pygame  import sys  # 초기화  pygame.init()    # 화면 설정  screen\_width = 800  screen\_height = 600  screen = pygame.display.set\_mode((screen\_width, screen\_height))  pygame.display.set\_caption("공 튀기기")    # 공 관련 변수  ball\_radius = 20  ball\_color = (255, 0, 0)  ball\_x = screen\_width // 2  # 화면 가로 중앙  ball\_y = screen\_height // 2  # 화면 세로 중앙  ball\_speed\_x = 0  # 초기 속도는 0  ball\_speed\_y = 0  move\_started = False  # 움직임 시작 여부    # 게임 루프  running = True  while running:      # 이벤트 처리     for event in pygame.event.get():           if event.type == pygame.QUIT:           running = False           elif event.type == pygame.KEYDOWN:           # 아무 키나 눌렀을 때 움직임 시작           if not move\_started:               ball\_speed\_x = 5           ball\_speed\_y = 5           move\_started = True       # 공 위치 업데이트     ball\_x += ball\_speed\_x     ball\_y += ball\_speed\_y       # 벽 충돌 처리     if ball\_x + ball\_radius >= screen\_width or ball\_x - ball\_radius <= 0:        ball\_speed\_x = -ball\_speed\_x     if ball\_y + ball\_radius >= screen\_height or ball\_y - ball\_radius <= 0:        ball\_speed\_y = -ball\_speed\_y       # 화면 색상 설정     screen.fill((255, 255, 255))       # 공 그리기     pygame.draw.circle(screen, ball\_color, (int(ball\_x), int(ball\_y)), ball\_radius)       # 화면 업데이트     pygame.display.flip()       # 초당 프레임 설정     pygame.time.Clock().tick(60)    # 종료  pygame.quit()  sys.exit() |