Team Id	PNT2022TMID26873
Date	17 November 2022
Project Name	AI-powered Nutrition Analysis and Fitness Enthusiasts

```
# Import messagebox from tkinter
from tkinter import Tk, Frame, Label, Entry


# Create a function to display report pagedef
report(data_list, gender, age):
    # Create window for report
    report_window = Tk()
    # Add title to the window
    report_window.title('REPORT')
    # Add width and height to the window
    report_window.geometry('400x600')

    # Create a frame to add title title_frame
    = Frame(report_window)# Add title
    label to the frame
    title = Label(title_frame, text="Report", fg="white")
    title.pack()

    # Pack title frame
    title_frame.pack(fill='x')#
    Show Data
    show_data(report_window, data_list, gender, age)


# Create a function to show data
def show_data(report_window, data_list, gender, age):#
    Create a frame for all the data
    data = Frame(report_window, pady=50)
    data.pack()

    # Create a label for bmi
```

```
re_bmi_label = Label(data, text="BMI (BODY MASS INDEX):")
re_bmi_label.grid(column=0, row=0, sticky='w')
# Create entry for bmi
re_bmi_entry = Entry(data)
re_bmi_entry.grid(column=1, row=0, sticky='e')

# Create a label for bp
re_bp_label = Label(data, text="BP (High/Medium/Low):")
re_bp_label.grid(column=0, row=1, sticky='w')
# Create entry for bp
re_bp_entry = Entry(data)
re_bp_entry.grid(column=1, row=1, sticky='e')

# Create a label for Pulse Rate
re_pulse_rate_label = Label(data, text="Pulse Rate (High/Medium/Low):")
re_pulse_rate_label.grid(column=0, row=2, sticky='w')
# Create entry for Pulse Rate
re_pulse_rate_entry = Entry(data)
re_pulse_rate_entry.grid(column=1, row=2, sticky='e')

# Create a label for RBC Count
re_rbc_label = Label(data, text="RBC Count (High/Medium/Low):")
re_rbc_label.grid(column=0, row=3, sticky='w')
# Create entry for RBC Count
re_rbc_entry = Entry(data)
re_rbc_entry.grid(column=1, row=3, sticky='e')

# Create a label for WBC Count
re_wbc_label = Label(data, text="WBC Count (High/Medium/Low):")
re_wbc_label.grid(column=0, row=4, sticky='w')
# Create entry for WBC Count
re_wbc_entry = Entry(data)
re_wbc_entry.grid(column=1, row=4, sticky='e')

# Create a label for Platelets
re_platelets_label = Label(data, text="Platelets (High/Medium/Low):")
re_platelets_label.grid(column=0, row=5, sticky='w')
# Create entry for Platelets
re_platelets_entry = Entry(data)
re_platelets_entry.grid(column=1, row=5, sticky='e')
```

```
    # Create a label for HB
    re_hb_label = Label(data, text="HB (High/Medium/Low):")
    re_hb_label.grid(column=0, row=6, sticky='w')
    # Create entry for HB
    re_hb_entry = Entry(data)
    re_hb_entry.grid(column=1, row=6, sticky='e')

    # Create a label for Uric Acid
    re_uric_acid_label = Label(data, text="Uric Acid (High/Medium/Low):")
    re_uric_acid_label.grid(column=0, row=7, sticky='w')
    # Create entry for Uric Acid
    re_uric_acid_entry = Entry(data)
    re_uric_acid_entry.grid(column=1, row=7, sticky='e')

    # Create a label for Cholesterol
    re_cholesterol_label = Label(data, text="Cholesterol (High/Medium/Low):")
    re_cholesterol_label.grid(column=0, row=8, sticky='w')
    # Create entry for Cholesterol
    re_cholesterol_entry = Entry(data)
    re_cholesterol_entry.grid(column=1, row=8, sticky='e')

    # Now insert all the values
    insert_data(re_bmi_entry, re_bp_entry, re_pulse_rate_entry, re_rbc_entry, re_wbc_entry,re_platelets_entry,
            re_hb_entry, re_uric_acid_entry, re_cholesterol_entry, data_list, gender, age)


# Create a function to insert all the values
def insert_data(re_bmi_entry, re_bp_entry, re_pulse_rate_entry, re_rbc_entry, re_wbc_entry,
re_platelets_entry,
            re_hb_entry, re_uric_acid_entry, re_cholesterol_entry, data_list, gender, age): #
    Insert all the values into the entries
    re_bmi_entry.insert(0, bmi(data_list[0].get(), data_list[1].get()))
    re_bp_entry.insert(0, bp_check(data_list[2].get(), data_list[3].get()))
    re_pulse_rate_entry.insert(0, pulse_check(gender.get(), data_list[4].get(), age.get()))
    re_rbc_entry.insert(0, rbc_check(gender.get(), data_list[5].get())) re_wbc_entry.insert(0,
    wbc_check(data_list[6].get()))
    re_platelets_entry.insert(0, platelets_check(data_list[7].get())) re_hb_entry.insert(0,
    hb_check(gender.get(), data_list[8].get())) re_uric_acid_entry.insert(0,
    uric_acid_check(gender.get(), data_list[9].get()))re_cholesterol_entry.insert(0,
    cholesterol_check(data_list[10].get()))
```

```
# Create a function to check bmidef
bmi(weight, height):
    # Change the values to the integerweight =
    eval(weight)
    height = eval(height) return

    weight / height ** 2


# Create a function to check bpdef
bp_check(low, high):
    # Change the values to the integerlow =
    eval(low)
    high = eval(high)

    # Return Low, Medium, High value accordinglyif
    low <= 60 or high <= 90:
        return "Low"
    elif low <= 80 or high <= 120:
        return "Medium"
    else:
        return "High"


# Create a function to check pulse def
pulse_check(gender, pulse, age):
    # Change the values to the integergender =
    int(gender)
    pulse = eval(pulse)
    age = eval(age)

    # Return Low, Medium, High value accordinglyif
    gender == 0:
        if age < 18:
            if pulse > 63:
                return "High"
            elif pulse < 61:
                return "Low"
            else:
```

```
            return "Medium"
        elif age < 35:
            if pulse > 65:
                return "High"
            elif pulse < 62:
                return "Low"
            else:
                return "Medium"
        elif age < 45:
            if pulse > 66:
                return "High"
            elif pulse < 63:
                return "Low"
            else:
                return "Medium"
        elif age <= 65:
            if pulse > 67:
                return "High"
            elif pulse < 62:
                return "Low"
            else:
                return "Medium"
        elif age > 65:
            if pulse > 65:
                return "High"
            elif pulse < 62:
                return "Low"
            else:
                return "Medium"
    else:
        if age < 18:
            if pulse > 63:
                return "High"
            elif pulse < 61:
                return "Low"
            else:
                return "Medium"
        elif age < 35:
            if pulse > 68:
                return "High"
            elif pulse < 65:
```

```
            return "Low"
        else:
            return "Medium"
    elif age < 45:
        if pulse > 66:
            return "High"
        elif pulse < 63:
            return "Low"
        else:
            return "Medium"
    elif age <= 65:
        if pulse > 68:
            return "High"
        elif pulse < 65:
            return "Low"
        else:
            return "Medium"
    elif age > 65:
        if pulse > 65:
            return "High"
        elif pulse < 62:
            return "Low"
        else:
            return "Medium"


# Create a function to check rbcdef
rbc_check(gender, rbc):
    # Change the values to the integergender =
    int(gender)
    rbc = eval(rbc)

    # Return Low, Medium, High value accordinglyif
    gender == 0:
        if rbc < 4.7:
            return "Low"
        elif rbc > 6.1:
            return "High"
        else:
            return "Medium"
    else:
```

```
    if rbc < 4.2:
        return "Low"
    elif rbc > 5.4:
        return "High"
    else:
        return "Medium"


# Create a function to check wbcdef
wbc_check(wbc):
    # Change the values to the integerwbc =
    eval(wbc)

    # Return Low, Medium, High value accordinglyif
    wbc < 4000:
        return "Low"
    elif wbc > 11000:
        return "High"
    else:
        return "Medium"


# Create a function to check plateletsdef
platelets_check(platelets):
    # Change the values to the integerplatelets =
    eval(platelets)

    # Return Low, Medium, High value accordinglyif
    platelets < 150000:
        return "Low"
    elif platelets > 450000:
        return "High"
    else:
        return "Medium"


# Create a function to check hbdef
hb_check(gender, hb):
    # Change the values to the integergender =
    int(gender)
    hb = eval(hb)
```

```
    # Return Low, Medium, High value accordinglyif
    gender == 0:
        if hb < 13.5:
            return "Low"
        elif hb > 17.5:
            return "High"
        else:
            return "Medium"
    else:
        if hb < 12.0:
            return "Low"
        elif hb > 15.5:
            return "High"
        else:
            return "Medium"


# Create a function to check uric_acid def
     uric_acid_check(gender, uric_acid):#
         Change the values to the integer
    gender = int(gender)
    uric_acid = eval(uric_acid)

    # Return Low, Medium, High value accordinglyif
    gender == 0:
        if uric_acid < 3.4:
            return "Low"
        elif uric_acid > 7.0:
            return "High"
        else:
            return "Medium"
    else:
        if uric_acid < 2.4:
            return "Low"
        elif uric_acid > 7.0:
            return "High"
        else:
            return "Medium"
```

```
# Create a function to check cholesteroldef
cholesterol_check(cholesterol):
    # Change the values to the integer
    cholesterol = eval(cholesterol)

    # Return Low, Medium, High value accordinglyif
    cholesterol < 200:
        return "Low"
    elif cholesterol > 240:
        return "High"
    else:
        return "Medium"
# Import all modules from tkinter
from tkinter import Tk, Frame, Label, Entry, IntVar, Radiobutton, Button, messagebox#
Import required function from report
from report import report

# Create a list to store input datadata_list =
[]
# Create a variable to store gender value
gender_value = 0
# Create a variable to store ageage_entry =
0


# Create a function to display home pagedef
home():
    # Create window for home pagewindow =
    Tk()
    # Add title of the window
    window.title("Fitness  Calculator")
    # Add width and height of the window
    window.geometry('500x600')

    # Add fields
    fields(window)

    # Loop the window display
    window.mainloop()
```

```
# Create a function to create different fielddef
fields(window):
    # Create name and age field
    name_age_field(window)

    # Create gender field
    gender_field(window)

    # Create health data entry field
    data_field(window)

    # Create a button to generate report
    generate = Button(window, text="Generate Report", command=check, height=2)generate.pack(pady=20)


# Create a function to create fields for name and age def
name_age_field(window):
    # Create a frame for name and age
    name_age_frame = Frame(window)
    name_age_frame.pack(pady=20)

    # Create name label
    name_label = Label(name_age_frame, text="Name:")
    name_label.grid(column=0, row=0)
    # Create a name entry
    name_entry = Entry(name_age_frame, bd=1)
    name_entry.grid(column=1, row=0, padx=5)

    # Create age label
    age_label = Label(name_age_frame, text="Age:")
    age_label.grid(column=3, row=0)
    # Create age entry
    global age_entry
    age_entry = Entry(name_age_frame, bd=1)
    age_entry.grid(column=4, row=0, padx=5)


# Create a function to create fields for genderdef
gender_field(window):
    # Create a frame for gender
```

```
    gender_frame = Frame(window)
    gender_frame.pack()

    # Create label for gender
    gender_label = Label(gender_frame, text="Gender:")
    gender_label.grid(column=0, row=0)

    # Create a variable to store gender value using IntVarglobal
    gender_value
    gender_value = IntVar()

    # Create radio for male and female
    male_radio = Radiobutton(gender_frame, text="Male", variable=gender_value, value=0)
    male_radio.grid(column=1, row=0)
    female_radio = Radiobutton(gender_frame, text="Female", variable=gender_value, value=1)
    female_radio.grid(column=2, row=0)


# Create a function to create fields for health datadef
data_field(window):
    # Create a frame for all health data valuesdata_frame =
    Frame(window, pady=50) data_frame.pack()

    # Create a label for weight
    weight_label = Label(data_frame, text="Weight (Kgs):")
    weight_label.grid(column=0, row=0, sticky='w')
    # Create weight entry
    weight_entry = Entry(data_frame, bd=1)
    weight_entry.grid(column=1, row=0, sticky='e')#
    Add entry to the list
    data_list.append(weight_entry)

    # Create a label for height
    height_label = Label(data_frame, text="Height (Ms):")
    height_label.grid(column=0, row=1, sticky='w')
    # Create height entry
    height_entry = Entry(data_frame, bd=1)
    height_entry.grid(column=1, row=1, sticky='e')#
    Add entry to the list
    data_list.append(height_entry)
```

```
# Create a label for BP Low
bp_low_label = Label(data_frame, text="BP Low (mmHg):")
bp_low_label.grid(column=0, row=2, sticky='w')
# Create bp_low entry
bp_low_entry = Entry(data_frame, bd=1)
bp_low_entry.grid(column=1, row=2, sticky='e')#
Add entry to the list
data_list.append(bp_low_entry)

# Create a label for BP High
bp_high_label = Label(data_frame, text="BP High (mmHg):")
bp_high_label.grid(column=0, row=3, sticky='w')
# Create bp_high entry
bp_high_entry = Entry(data_frame, bd=1)
bp_high_entry.grid(column=1, row=3, sticky='e')#
Add entry to the list
data_list.append(bp_high_entry)

# Create a label for Pulse Rate
pulse_rate_label = Label(data_frame, text="Pulse Rate (bpm):")
pulse_rate_label.grid(column=0, row=4, sticky='w')
# Create Pulse Rate entry
pulse_rate_entry = Entry(data_frame, bd=1)
pulse_rate_entry.grid(column=1, row=4, sticky='e')#
Add entry to the list
data_list.append(pulse_rate_entry)

# Create a label for RBC
rbc_label = Label(data_frame, text="RBC (million/mm3):")
rbc_label.grid(column=0, row=5, sticky='w')
# Create RBC entry
rbc_entry = Entry(data_frame, bd=1)
rbc_entry.grid(column=1, row=5, sticky='e')#
Add entry to the list
data_list.append(rbc_entry)

# Create a label for WBC
wbc_label = Label(data_frame, text="WBC (cells/mm3):")
wbc_label.grid(column=0, row=6, sticky='w')
# Create WBC entry
```

```
wbc_entry = Entry(data_frame, bd=1)
wbc_entry.grid(column=1, row=6, sticky='e')#
Add entry to the list
data_list.append(wbc_entry)

# Create a label for Platelets
platelets_label = Label(data_frame, text="Platelets (billion/L):")
platelets_label.grid(column=0, row=7, sticky='w')
# Create Platelets entry
platelets_entry = Entry(data_frame, bd=1)
platelets_entry.grid(column=1, row=7, sticky='e')#
Add entry to the list
data_list.append(platelets_entry)

# Create a label for HB
hb_label = Label(data_frame, text="HB (g/dl):")
hb_label.grid(column=0, row=8, sticky='w')
# Create HB entry
hb_entry = Entry(data_frame, bd=1)
hb_entry.grid(column=1, row=8, sticky='e')#
Add entry to the list
data_list.append(hb_entry)

# Create a label for Uric Acid
uric_acid_label = Label(data_frame, text="Uric Acid (mg/dl):")
uric_acid_label.grid(column=0, row=9, sticky='w')
# Create Uric Acid entry
uric_acid_entry = Entry(data_frame, bd=1)
uric_acid_entry.grid(column=1, row=9, sticky='e')#
Add entry to the list
data_list.append(uric_acid_entry)

# Create a label for Cholesterol
cholesterol_label = Label(data_frame, text="Cholesterol (mg/dl):")
cholesterol_label.grid(column=0, row=10, sticky='w')
# Create Cholesterol entry cholesterol_entry =
Entry(data_frame, bd=1)
cholesterol_entry.grid(column=1, row=10,  sticky='e')#
Add entry to the list data_list.append(cholesterol_entry)
```

```
# Create a function to check if data is valid or notdef
check():
    # Create a loop to check if the values are valid or notfor data
    in data_list:
        # Show warning if value is emptyif
        data.get() == "":
            messagebox.showwarning("Warning", "All values Required")break
    else:
        # Show report window
        report(data_list, gender_value,  age_entry) #
Import the required functions from home from
home import home


# Create a function to run the appdef
run():
    home()


# Run the app
run()
```