**4조(Tkinter) 수업정리 자료**

**이름 : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**《수업 목표》**

* **Tkinter를 이용한 여러 예제를 함께 실습해보고 이를 활용하여**

**스터디 플래너를 구현해 본다**

**Tkinter란?**

* **Tcl/Tk를 파이썬에 사용할 수 있도록 한 경량의 그래픽 사용자 인터페이스**
* **파이썬에 기본적으로 내장되어 있는 표준 패키지**
* **통합 개발 환경인 파이썬 IDLE가 Tkinter를 사용하여 작성 됨**

**GUI란?**

* **그래픽 사용자 인터페이스의 약자**
* **텍스트 모드가 아닌 위젯/컴포넌트라 불리는 윈도, 각종 버튼 등을 사용하여 사용자와 프로그램 간의 정보를 교환하는 방식**

**pack()**

from tkinter import\*

win = Tk()

win.title('pack 예제')

label = Label(win, width=50, height=5)

label.pack()

btn1 = Button(label, text="top")

btn1.pack(side = "top")

btn2 = Button(label, text="left")

btn2.pack(side = "left")

btn3 = Button(label, text="right")

btn3.pack(side = "right")

btn4 = Button(label, text="bottom")

btn4.pack(side = "bottom")

win.geometry("300x300")

win.mainloop()

1. **위젯들을 부모 위젯에 패킹하여**

**불필요한 공간을 없앤다**

1. **Side 옵션을 사용하여 위치**

**지정이 가능하다**

**레이블, 엔트리, 버튼 위젯으로 윈도 생성**

from tkinter import \*

win = Tk()

win.title('여러 위젯 구성')

lb1 = Label(win, text="레이블(Lable)")

lb1.pack()

txt = Entry(win)

txt.insert(0, '엔트리(Entry)')

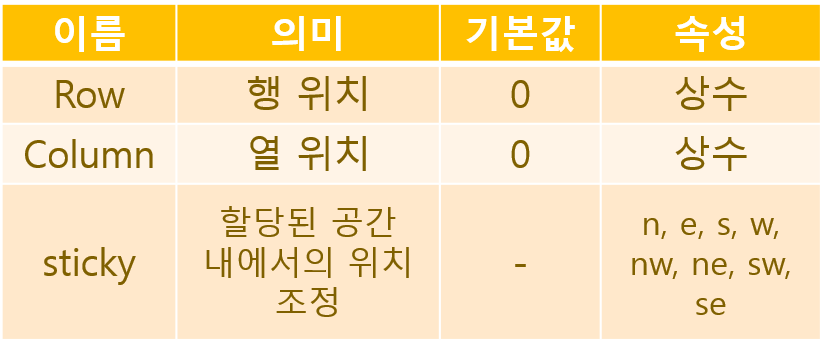
txt.pack()

btn = Button(win, text = "OK")

btn.pack()

win.mainloop()

**체크박스 선택 후 버튼 클릭 시 문구 출력 이벤트**



**Grid()**

import tkinter as tk

from tkinter import \*

win = tk.Tk()

win.title("체크박스 예제")

win.geometry("600x400")

#버튼 클릭하면 선택한 체크버튼을 선택하였다는 문구 출력하는 함수 설정

def sub\_chk():

if ChkVar1.get() == 1:

lb2.configure(text = "파이썬을 선택하였습니다.")

elif ChkVar2.get() == 1:

lb2.configure(text = "자바를 선택하였습니다.")

elif ChkVar3.get() == 1:

lb2.configure(text = "알고리즘을 선택하였습니다.")

#"과목을 선택하세요" 라벨 생성

lb1 = tk.Label(win, text="과목을 선택하세요.", bg = "yellow")

lb1.grid(row = 0, column = 0, columnspan = 3)

**if\_\_name\_\_ == “\_\_main\_\_”:**

#"파이썬", "자바", "알고리즘" 체크버튼 생성

ChkVar1 = tk.IntVar()

chk1 = tk.Checkbutton(win, text = "파이썬", variable = ChkVar1)

chk1.grid(row = 1, column = 0, sticky = "w")

ChkVar2 = tk.IntVar()

chk2 = tk.Checkbutton(win, text = "자바", variable = ChkVar2)

chk2.grid(row = 1, column = 1, sticky = "w")

ChkVar3 = tk.IntVar()

chk3 = tk.Checkbutton(win, text = "알고리즘", variable = ChkVar3)

chk3.grid(row = 1, column = 2, sticky = "w")

#결과물 출력되는 라벨 생성

lb2 = tk.Label(win)

lb2.grid(row = 2, column = 0, sticky = "w", columnspan = 3)

#체크버튼을 선택 후 클릭 시 함수 실행

btn1 = tk.Button(win, text = "선택", command = sub\_chk)

btn1.grid(row = 1, column = 3, sticky = "w")

win.mainloop()

* \_\_name\_\_ 현재 모듈의 이름을 담고 있는 내장 변수
* 직접 실행한 파일은 \_\_name\_\_의 내장변수에 \_\_main\_\_이라는 값이 들어가고 그렇지 않은 import된 모듈들은 각 모듈의 이름이 들어감
* If\_\_name\_\_ == “\_\_main\_\_”이라는 조건문을 넣어주고 그 아래는 직접 실행시켰을 때만 실행되길 원하는 코드들을 넣어줌

**막대그래프**

# tkinter 모듈을 tk로 불러옴

import tkinter as tk

# tkinter.messagebox 모듈을 mb로 불러옴

import tkinter.messagebox as mb

# matplotlib.pyplot 모듈을 plt로 불러옴

import matplotlib.pyplot as plt

# BarChartApp 클래스 정의

class BarChartApp(tk.Frame):

# 생성자 함수

def \_\_init\_\_(self, master=None):

super().\_\_init\_\_(master)

self.master = master

self.master.title("Bar Chart App")

self.create\_widgets()

# 위젯을 생성하는 함수

def create\_widgets(self):

# 데이터 입력을 위한 라벨과 엔트리 위젯 생성

tk.Label(self.master, text="apple").grid(row=0, column=0, padx=10, pady=10)

self.apple\_entry = tk.Entry(self.master)

self.apple\_entry.grid(row=0, column=1)

tk.Label(self.master, text="banana").grid(row=1, column=0, padx=10, pady=10)

self.banana\_entry = tk.Entry(self.master)

self.banana\_entry.grid(row=1, column=1)

tk.Label(self.master, text="pear").grid(row=2, column=0, padx=10, pady=10)

self.pear\_entry = tk.Entry(self.master)

self.pear\_entry.grid(row=2, column=1)

# 막대그래프를 출력하는 버튼 생성

tk.Button(self.master, text="막대 그래프 보기",

command=self.show\_bar\_chart).grid(row=3, column=0,

columnspan=2, padx=10, pady=10)

# 막대그래프 출력 함수

def show\_bar\_chart(self):

# 엔트리 위젯에서 입력받은 값을 변수에 저장

try:

apple\_value = float(self.apple\_entry.get())

banana\_value = float(self.banana\_entry.get())

pear\_value = float(self.pear\_entry.get())

# 숫자가 아닌 값을 입력한 경우 오류창을 띄움

except ValueError:

mb.showerror("Error", "숫자를 입력해주세요.")

return

# 막대그래프 생성

labels = ["apple", "banana", "pear"]

values = [apple\_value, banana\_value, pear\_value]

plt.bar(labels, values)

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

# tkinter 루트 윈도우 생성

root = tk.Tk()

# BarChartApp 클래스의 인스턴스 생성

app = BarChartApp(master=root)

# 이벤트 루프 실행

app.mainloop()

**Entry 값 입력 받아 Label에 출력**

from tkinter import \*

from tkinter import ttk

# Tkinter 인스턴스 생성

win = Tk()

# Window 사이즈 지정

win.geometry("300x200+100+100")

win.resizable(False, False)

# get() : entry에 텍스트를 문자열로 반환

def Cal():

label.configure(text="결과 값 = " + str(eval(entry.get())))

entry = Entry(win)

entry.pack()

button = Button(win, text="계산", command=Cal)

button.pack()

label = Label(win)

label.pack()

win.mainloop()

**간단한 더하기 계산 프로그램(+예외처리)**

import tkinter as tk

def calculate():

try:

result = int(data1\_entry.get()) + int(data2\_entry.get())

result\_label.config(text="결과: " + str(result))

except ValueError:

result\_label.config(text="숫자를 입력해주세요.")

root = tk.Tk()

root.title("간단한 계산기")

data1\_label = tk.Label(root, text="Data1:")

data1\_label.pack()

data1\_entry = tk.Entry(root)

data1\_entry.pack()

data2\_label = tk.Label(root, text="Data2:")

data2\_label.pack()

data2\_entry = tk.Entry(root)

data2\_entry.pack()

calculate\_button = tk.Button(root, text="계산", command=calculate)

calculate\_button.pack()

result\_label = tk.Label(root)

result\_label.pack()

root.mainloop()

**--스터디 플래너 프로그램 구현--**

**기능**

* **공부 할 과목을 입력**
* **공부를 시작한 시간과 끝난 시간 입력하고 시간 간의 차이를 계산하여 과목별 공부 시간을 계산**
* **공부 한 시간 그래프에 저장**

**--스터디 플래너 구현 코드--**

import tkinter as tk

from tkinter import messagebox

import matplotlib.pyplot as plt

event\_counter = 1

def add\_event():

global event\_counter

event = event\_entry.get()

start\_time = start\_time\_entry.get()

end\_time = end\_time\_entry.get()

if event and start\_time and end\_time:

if start\_am\_pm.get() == "PM":

start\_time = add\_hours(start\_time, 12)

if end\_am\_pm.get() == "PM":

end\_time = add\_hours(end\_time, 12)

study\_time = calculate\_study\_time(start\_time, end\_time)

event\_info = f"{event} (시작시각: {start\_time}, 종료시각: {end\_time}, 공부시간: {study\_time})"

event\_listbox.insert(tk.END, f"{event\_counter}. {event\_info}")

event\_counter += 1

clear\_entries()

else:

messagebox.showwarning("경고", "공부, 시작시각, 종료시각을 모두 입력해주세요.")

def add\_hours(time\_str, hours):

hour, minute = map(int, time\_str.split(":"))

total\_minutes = hour \* 60 + minute + (hours \* 60)

new\_hour = total\_minutes // 60

new\_minute = total\_minutes % 60

return f"{new\_hour:02d}:{new\_minute:02d}"

def calculate\_study\_time(start\_time, end\_time):

start\_hour, start\_minute = map(int, start\_time.split(":"))

end\_hour, end\_minute = map(int, end\_time.split(":"))

total\_minutes = (end\_hour \* 60 + end\_minute) - (start\_hour \* 60 + start\_minute)

hours = total\_minutes // 60

minutes = total\_minutes % 60

study\_time = f"{hours}.{minutes // 10} 시간"

return study\_time

def clear\_entries():

event\_entry.delete(0, tk.END)

start\_time\_entry.delete(0, tk.END)

end\_time\_entry.delete(0, tk.END)

def delete\_event():

selected\_index = event\_listbox.curselection()

if selected\_index:

event\_listbox.delete(selected\_index)

global event\_counter

event\_counter = 1

events = event\_listbox.get(0, tk.END)

event\_listbox.delete(0, tk.END)

for event in events:

event\_listbox.insert(tk.END, f"{event\_counter}. {event}")

event\_counter += 1

def clear\_all():

event\_listbox.delete(0, tk.END)

global event\_counter

event\_counter = 1

def show\_bar\_chart():

study\_times = []

events = event\_listbox.get(0, tk.END)

for event in events:

study\_time = float(event.split("공부시간: ")[-1].split(" 시간")[0])

study\_times.append(study\_time)

labels = [event.split(". ")[1].split(" (")[0] for event in events]

plt.bar(labels, study\_times)

plt.xlabel('subject')

plt.ylabel('studytime(hour)')

plt.title('studytime for each subject')

plt.show()

root = tk.Tk()

event\_frame = tk.Frame(root)

event\_frame.pack(pady=10)

event\_label = tk.Label(event\_frame, text="공부 할 과목:")

event\_label.grid(row=0, column=0)

event\_entry = tk.Entry(event\_frame)

event\_entry.grid(row=0, column=1)

start\_time\_label = tk.Label(event\_frame, text="시작시각:")

start\_time\_label.grid(row=1, column=0)

start\_time\_entry = tk.Entry(event\_frame)

start\_time\_entry.grid(row=1, column=1)

start\_am\_pm = tk.StringVar(value="AM")

start\_am\_radio = tk.Radiobutton(event\_frame, text="AM", variable=start\_am\_pm, value="AM")

start\_am\_radio.grid(row=1, column=2)

start\_pm\_radio = tk.Radiobutton(event\_frame, text="PM", variable=start\_am\_pm, value="PM")

start\_pm\_radio.grid(row=1, column=3)

end\_time\_label = tk.Label(event\_frame, text="종료시각:")

end\_time\_label.grid(row=2, column=0)

end\_time\_entry = tk.Entry(event\_frame)

end\_time\_entry.grid(row=2, column=1)

end\_am\_pm = tk.StringVar(value="AM")

end\_am\_radio = tk.Radiobutton(event\_frame, text="AM", variable=end\_am\_pm, value="AM")

end\_am\_radio.grid(row=2, column=2)

end\_pm\_radio = tk.Radiobutton(event\_frame, text="PM", variable=end\_am\_pm, value="PM")

end\_pm\_radio.grid(row=2, column=3)

add\_button = tk.Button(event\_frame, text="추가", command=add\_event)

add\_button.grid(row=0, column=4, padx=10)

event\_listbox = tk.Listbox(root, width=50)

event\_listbox.pack(pady=10)

delete\_button = tk.Button(root, text="삭제", command=delete\_event)

delete\_button.pack()

clear\_button = tk.Button(root, text="전체 삭제", command=clear\_all)

clear\_button.pack(pady=10)

show\_chart\_button = tk.Button(root, text="그래프로 보기", command=show\_bar\_chart)

show\_chart\_button.pack(pady=10)

root.mainloop()