

날씨 온도 기반 업/다운 게임

1 개요:

이 코드는 전통적인 업다운 게임에 실시간 날씨 API를 결합하여, 세계 주요 도시의 현재 온도를 맞추는 흥미로운 게임을 구현하였다. 추가로 Advice Slip API를 통합하여 게임 중 조언을 받을 수 있는 기능을 포함하였다.

1.1 게임 규칙 및 특징

플레이어는 7개 도시(서울, 부산, 뉴욕, 런던, 도쿄, 파리, 시드니) 중 선택
 선택한 도시의 실시간 온도를 10번의 기회 안에 맞추기
 추측할 때마다 UP/DOWN 힌트와 근접도 피드백 제공
 실시간 날씨 정보(날씨 상태, 습도) 표시
 언제든지 "조언 받기" 버튼으로 랜덤 조언 수신

2 사용된 라이브러리 및 API

tkinter: GUI 프레임워크
 requests: HTTP 통신 라이브러리
 Open-Meteo API: 무료 날씨 데이터 API (API 키 불필요)
 Advice Slip API: 무료 조언 제공 API

3 구현 결과 및 코드 분석

3.1 GUI 구성 및 사용자 경험

LabelFrame을 활용한 논리적 섹션 구분 (도시 선택, 날씨 정보, 게임 정보, 조언)
 색상 코딩: 각 섹션별 배경색 차별화 (#BBDEFB, #FFF9C4, #C8E6C9 등)
 Combobox: 드롭다운 메뉴로 도시 선택의 편의성 제공
 Enter 키 바인딩: self.temp_entry.bind('<Return>', ...) - 빠른 입력 지원

3.2 게임 로직 구현

버튼 실수 조작 방지: 게임 진행 상태에 따라 버튼의 활성화/비활성화를 동적으로 제어하여 사용자의 잘못된 조작을 방지한다. game_active 플래그로 게임 진행 여부를 명확히 관리한다. (line 232~)

```

209 def start_game(self):
210     """게임 시작"""
211     city = self.city_var.get()
212     self.city_name = city
213     lat, lon = self.cities[city]
214
215     # 날씨 정보 가져오기
216     weather = self.fetch_weather(lat, lon)
217
218     if weather is None:
219         return
220
221     self.target_temp = weather['temperature']
222     weather_desc = self.get_weather_description(weather['weather_code'])
223
224     # 날씨 정보 표시
225     weather_info = (f"🌤️ 도시: {city}\n"
226                    f"🌡️ 현재 날씨: {weather_desc}\n"
227                    f"💧 습도: {weather['humidity']}%\n"
228                    f"❓ 현재 온도를 맞춰보세요!")
229
230     self.weather_label.config(text=weather_info)
231
232     # 게임 초기화
233     self.attempts = 0
234     self.game_active = True
235     self.update_attempts()
236     self.hint_label.config(text="범위: -30°C ~ 50°C")
237
238     # 버튼 상태 변경
239     self.start_btn.config(state='disabled')
240     self.guess_btn.config(state='normal')
241     self.temp_entry.delete(0, 'end')
242     self.temp_entry.focus()
243
244     messagebox.showinfo("게임 시작",
245                          f"{city}의 현재 온도를 맞춰보세요!\n"
246                          f"10번의 기회가 있습니다.")
  
```

```

252 def check_guess(self):
253     """추측 확인"""
254     if not self.game_active:
255         messagebox.showwarning("알림", "먼저 게임을 시작하세요!")
256         return
257
258     try:
259         guess = int(self.temp_entry.get())
260     except ValueError:
261         messagebox.showerror("오류", "올바른 숫자를 입력하세요!")
262         return
263
264     self.attempts += 1
265     self.update_attempts()
266
267     diff = abs(self.target_temp - guess)
268
269     # 정답 확인
270     if guess == self.target_temp:
271         self.game_active = False
272         self.guess_btn.config(state='disabled')
273         self.start_btn.config(state='normal')
274
275         messagebox.showinfo("축하합니다! 🎉",
276                             f"정답입니다!\n\n"
277                             f"온도: {self.target_temp}°C\n"
278                             f"시도 횟수: {self.attempts}회")
279         return
280
281     # 힌트 제공
282     if diff <= 2:
283         hint = "🔥 매우 뜨겁습니다! (±2°C 이내)"
284     elif diff <= 5:
285         hint = "🔥 뜨겁습니다! (±5°C 이내)"
286     elif diff <= 10:
287         hint = "🌡️ 따뜻합니다! (±10°C 이내)"
288     else:
289         hint = "❄️ 차갑습니다! (±10°C 이상)"
290
291     if guess < self.target_temp:
292         direction = "⬆️ UP! 더 높은 온도입니다."
293     else:
294         direction = "⬆️ DOWN! 더 낮은 온도입니다."
295
296     self.hint_label.config(text=f"{hint}\n{direction}")
297
298     # 시도 횟수 초과
299     if self.attempts >= self.max_attempts:
300         self.game_active = False
301         self.guess_btn.config(state='disabled')
302         self.start_btn.config(state='normal')
303
304         messagebox.showinfo("게임 종료",
305                             f"시도 횟수를 모두 사용했습니다.\n\n"
306                             f"정답: {self.target_temp}°C\n"
307                             f"{self.city_name}의 현재 온도였습니다.")
308
309     self.temp_entry.delete(0, 'end')
310     self.temp_entry.focus()

```

1. 입력 검증: try-except 블록으로 숫자 입력 검증
2. 근접도 피드백: 4단계 힌트 시스템 ($\pm 2^{\circ}\text{C}$, $\pm 5^{\circ}\text{C}$, $\pm 10^{\circ}\text{C}$, 그 이상)
3. 방향 힌트: UP/DOWN 아이콘(⬆️⬆️)으로 직관적 안내
4. 게임 종료 조건: 정답 맞추기 또는 10회 시도 소진

3.3 API 구현 내용

3.3.1 Weather API 활용 - Open-Meteo

Open-Meteo API의 장점:

데이터 파싱 전략:

```

178 def get_weather_description(self, code):
179     """날씨 코드를 설명으로 변환"""
180     weather_codes = {
181         0: "맑음", 1: "대체로 맑음", 2: "부분적으로 흐림", 3: "흐림",
182         45: "안개", 48: "서리 안개",
183         51: "이슬비", 53: "중간 이슬비", 55: "강한 이슬비",
184         61: "약한 비", 63: "보통 비", 65: "강한 비",
185         71: "약한 눈", 73: "보통 눈", 75: "강한 눈",
186         80: "약한 소나기", 81: "보통 소나기", 82: "강한 소나기",
187         95: "뇌우"
188     }
189     return weather_codes.get(code, "알 수 없음")

```

WMO 날씨 코드 표준: Open-Meteo API는 세계기상기구(WMO) 표준 코드를 사용한다. 이를 한 글로 변환하여 사용자에게 직관적인 정보를 제공한다. 또한 다음과 같은 이점이 있다.

실시간 데이터: 현재 온도, 습도, 날씨 코드 제공

위도/경도 기반: 전 세계 모든 위치의 날씨 정보 접근 가능

타임아웃 설정: 5초 제한으로 네트워크 지연 대응

temperature_2m: 지표면 2m 높이의 온도 (가장 표준적인 측정값)

round() 함수로 소수점 제거하여 게임 난이도 적정화

3.3.2 Advice Slip API 활용

원할 때 “조언 받기” 버튼을 누르면 랜덤한 조언을 호출 받아 제공한다.

```

191     def get_advice(self):
192         """Advice Slip API에서 조언 가져오기"""
193         try:
194             response = requests.get("https://api.adiceslip.com/advice", timeout=5)
195
196             if response.status_code == 200:
197                 data = response.json()
198                 advice = data['slip']['advice']
199
200                 self.advice_text.config(state='normal')
201                 self.advice_text.delete('1.0', 'end')
202                 self.advice_text.insert('1.0', f"💡 {advice}")
203                 self.advice_text.config(state='disabled')
204             else:
205                 messagebox.showwarning("알림", "조언을 불러올 수 없습니다.")
206         except Exception as e:
207             messagebox.showerror("오류", f"조언 API 오류: {str(e)}")

```

4 에러 방어:

4.1 각 함수에서 try-except 구문을 사용하여 언제나 오류가 발생했을 때, 오류가 발생했다는 팝업 창을 띄워. 코드가 멈추거나 영원히 동작하는 상황을 예방하였다.

4.2 API 에러 핸들링

1. 네트워크 타임아웃: 5초 제한으로 무한 대기 방지
2. HTTP 상태 코드 검증: 200 응답만 처리
3. 포괄적 예외 처리: 모든 예외를 캐치하여 프로그램 중단 방지

5 결론

두 가지 무료 API(Weather, Advice) 성공적으로 사용가능하게 통합하였다.

API 키 불필요한 즉시 실행 가능한 코드를 완성하여 누구나 쉽게 실행할 수 있도록 하였다.

전 세계 7개 도시의 실시간 날씨 데이터 활용하여 재미있으며 지루하지 않은 게임을 만들었다.

4단계 근접도 힌트로 게임성 향상시켰고, UI를 보는 맛이 있게 되었다.

명확한 게임 상태 관리 및 예외 처리에 대해 새롭게 배우는 것이 많았다.

추후에 온도 업/다운 게임 외에 날씨나 습도까지도 맞추게 하는 기능을 추가할 수 있겠다는 생각을 하게 되었다.

이로 본 보고서를 마친다.