PyQt6 기반 계산기 시스템

괄호 처리, 연산자 우선순위, 상태 기반 입력 흐름을 지원하는 GUI 계산기

1. GUI 화면 구성

디스플레이

구성 요소	설명	Qt 객체명
상부 디스플레이	사용자가 입력한 전체 수식 표시	lineEdit_2
하부 디스플레이	현재 입력 중인 숫자 또는 계산 결과 표시	lineEdit
텍스트 메시지 영역	입력 오류 발생 시 메시지를 2초간 표시	textBrowser

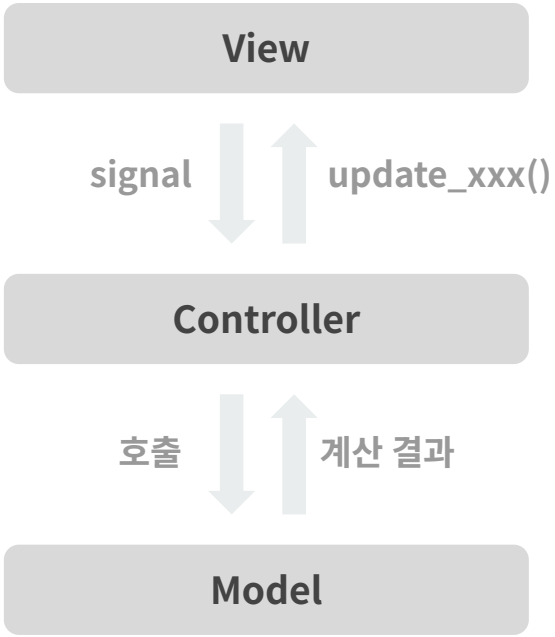
Note
입력 오류, 잘못된 괄호, 중복 소수점 등 **사용자 실수에 대한 안내 메시지**를 **텍스트 메시지 영역**에 표시합니다. QTimer를 통해 2초 후 자동으로 사라지도록 처리하여 UX를 개선했습니다.

버튼

버튼 분류	연결된 동작	호출 함수 (Slot)
숫자 버튼	숫자 입력	handle_digit('7')
소수점 버튼	소수점 입력	handle_digit('.')
연산자 버튼	연산자 입력	handle_operator()
괄호 버튼	괄호 입력	handle_lparen(), handle_rparen()
부호 전환 버튼	부호 전환	handle_sign()
등호 버튼	수식 계산	handle_equal()
C 버튼	부분 삭제	handle_c()
AC 버튼	전체 초기화	handle_ac()

2. 설계 특징

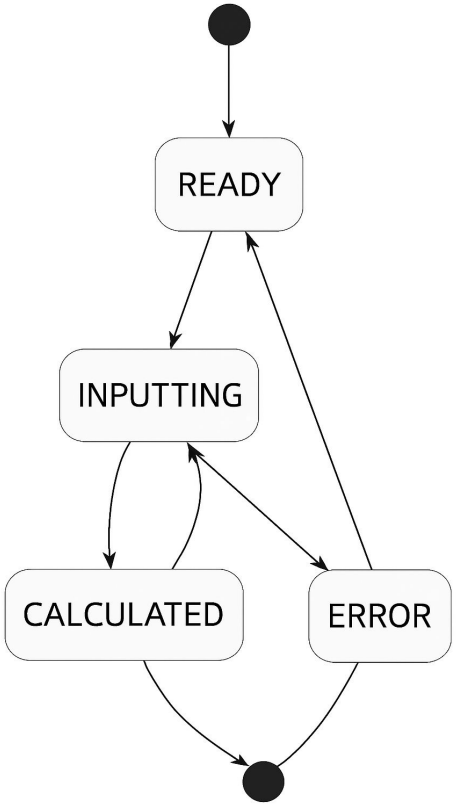
MVC 구조



핵심 내용	• PyQt 앱을 Model, View, Controller로 역할 분리 • 각 모듈은 서로 직접 접근하지 않고 Controller를 통해 간접 연결

각 요소별 설명	
Controller	**중심 허브**. View와 Model 모두를 알고 있고, 호출과 반영을 직접 처리함.
Model	**계산 로직, 상태 처리 전담**. 직접 UI 업데이트는 하지 않음.
View	**사용자 인터페이스와 signal 전달자 역할**. 자체 로직 없음.

상태 기반 입력 처리



핵심 내용	• **CalcState 상태(Enum)** ◦ READY, INPUTTING, CALCULATED, ERROR • **각 상태에 따라 입력 흐름을 제어함** ◦ CALCULATED 상태에서 숫자 입력 시 자동 초기화 ◦ ERROR 상태에서 입력 시 자동 리셋 • **입력 무결성 유지 및 사용자 의도 반영 설계**
예시 흐름	• **계산 완료(CALCULATED)** → 숫자 입력 → 수식 초기화 후 입력 시작 • **에러 발생(ERROR)** → 숫자 입력 → 전체 리셋 후 입력 시작

괄호/소수점 자동 보완 처리

핵심 내용	• **여는 괄호 뒤에 숫자 없이 닫는 괄호 입력** → 여는 괄호 자동 삭제 • **소수점 입력시 0. 또는 -0. 자동 보완** • **수식 평가 전 괄호 개수 자동 맞춤** → evaluate()에서 여는 괄호 수에 따라 자동 닫힘
사용자 UX 목적	• **입력 실수 방지 (빈 괄호 자동 제거 등)** • **소수점 입력 혼란 해소** • **괄호 누락으로 인한 수식 오류 사전 방지**

암시적 곱셈 처리

핵심 내용	• **암묵적 곱셈 자동 삽입 조건** ◦ 숫자 뒤에 여는 괄호 입력 → 예: 2(→ 2*(◦ 닫는 괄호 뒤에 숫자 입력→ 예:)2 →)*2
구현 방식	• **input_lparen(), input_digit() 등에서 tokens 상태를 확인하여 * 자동 삽입**
효과	• *** 생략에도 정확한 수식 인식 가능** • **실제 계산기와 유사한 입력 경험 제공**

입력-연산 파이프라인

숫자/연산자 입력

↓

tokens 리스트에 저장

↓

후위 표기법으로 변환

↓

스택 기반 계산

↓

결과 출력

핵심 내용	• `to_postfix()` 함수에서 후위 표기법으로 변환 • `evaluate_postfix()` 함수에서 스택 계산 수행 • 연산자 우선순위 적용 ◦ `*`, `/` 에 `+`, `-` 보다 높은 우선순위 적용 • 괄호 중첩, 소수점 연산 등 복합 수식 지원
효과	• 복잡한 수식도 정확하게 계산 • 표준 계산기 수준의 수식 평가 기능 구현

QTimer 기반 UX 개선

핵심 내용	• **입력 오류 발생 시 메시지 출력** ◦ textBrowser.setText() • **2초 후 자동 메시지 삭제** ◦ QTimer.singleshot()
메시지 출력 대상	• **괄호 개수 불일치** • **중복 소수점 입력** • **연산자 중복 입력 등**
UX 개선 효과	• **오류 메시지 수동 제거 불필요** • **자연스럽고 끊김 없는 입력 흐름 유지** • **사용자 실수는 알려주되, 흐름은 방해하지 않음**

3. 기능 구현

주요 기능

주요 기능

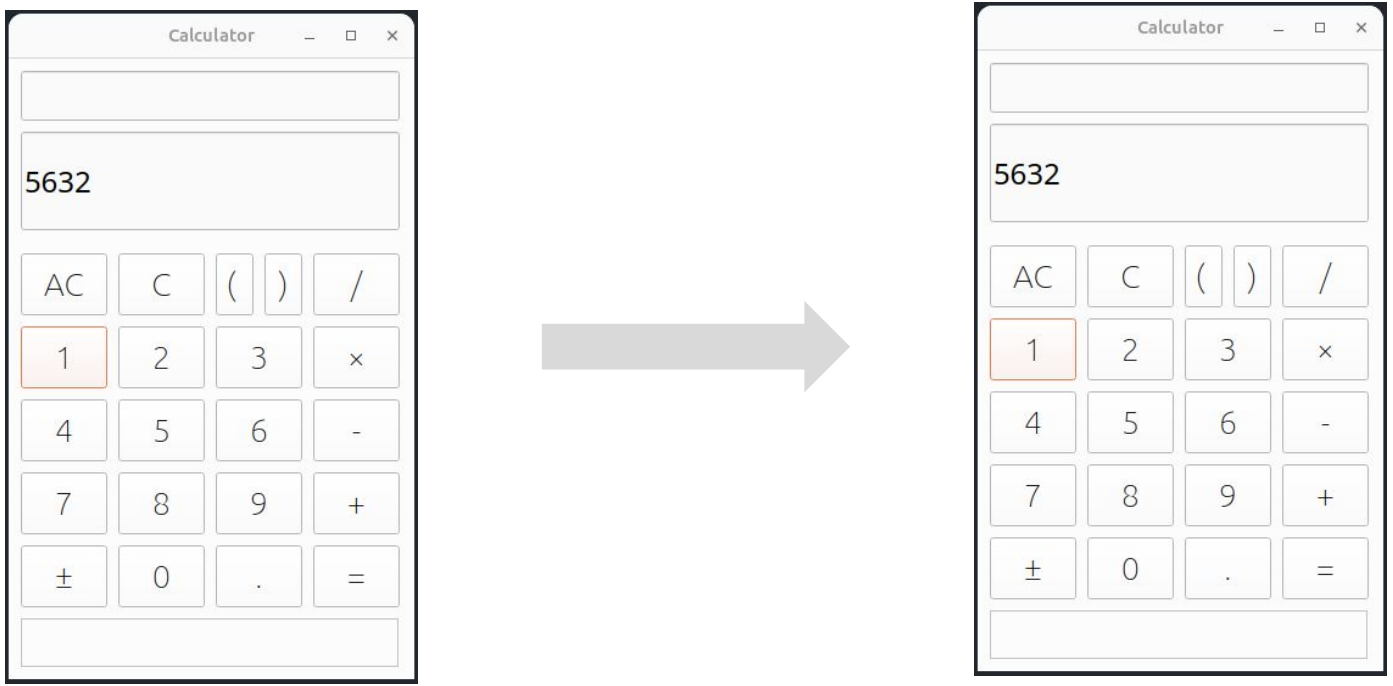
#	분류	System Requirements	
1	입력 처리 기능	SR_01	숫자를 입력하는 기능
		SR_02	소수점을 입력하고 중복을 방지하는 기능
		SR_03	± 부호를 토글하는 기능
		SR_04	연산자를 입력하고 중복을 처리하는 기능
		SR_05	괄호를 입력하고 곱셈을 자동 처리하는 기능
		SR_06	괄호의 개수를 검증하고 빈 괄호를 방지하는 기능
2	입력 및 상태 초기화 기능	SR_07	계산 후 새로운 입력을 초기화하는 기능
		SR_08	전체 초기화(AC) 또는 부분 삭제(C)를 수행하는 기능

주요 기능

#	분류	System Requirements	
3	**계산 처리 기능**	SR_09	중위 수식을 후위 표기법으로 변환하는 기능
		SR_10	후위 표기 수식을 스택으로 계산하는 기능
		SR_11	괄호와 연산자를 자동 보완하는 기능
4	**예외 처리 및 결과 출력 기능**	SR_12	수식 오류 발생 시 메시지를 출력하는 기능
		SR_13	수식과 결과를 라인에 나누어 출력하는 기능
		SR_14	정수 결과 시 .0을 제거하여 출력하는 기능
		SR_15	입력 UX 보완 메시지를 출력하는 기능

입력 처리 기능

SR_01. 숫자(0-9)를 입력하는 기능

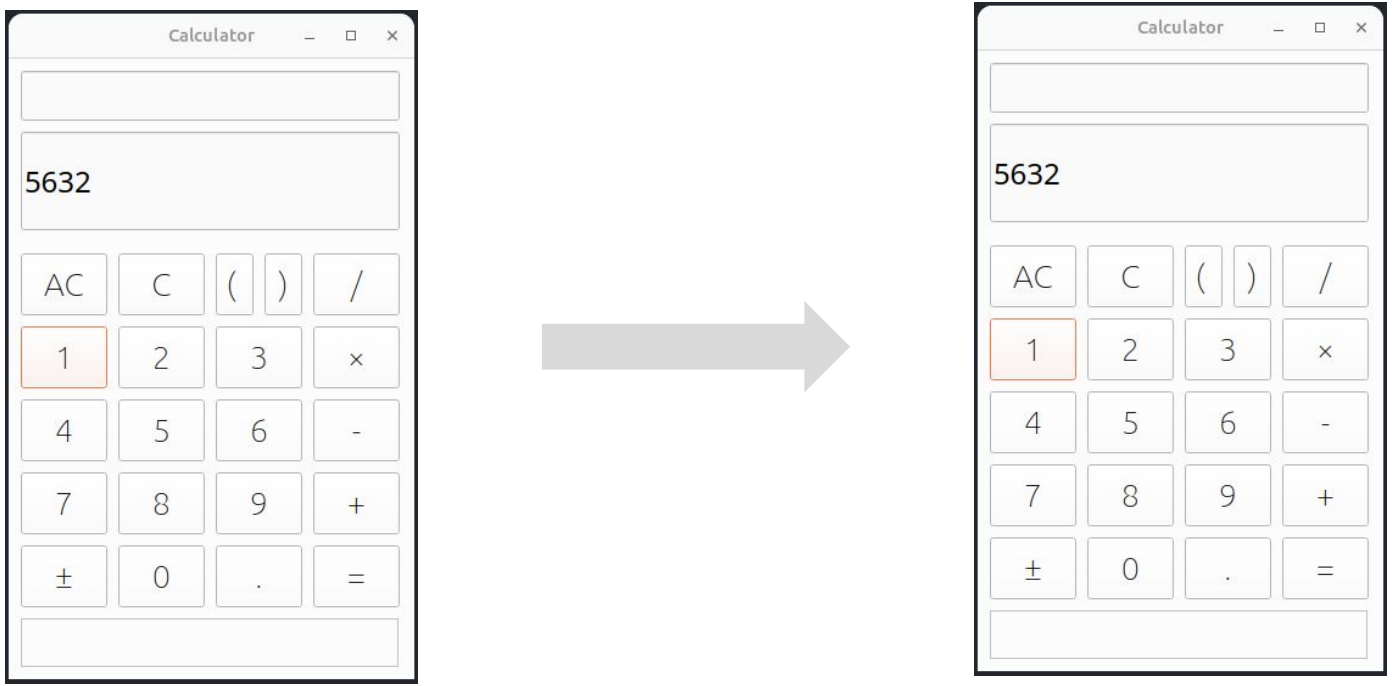


기능 요약	0~9 숫자 버튼 클릭 시 숫자를 current_input에 추가

SR_01. 숫자(0-9)를 입력하는 기능

적용 조건	READY, INPUTTING, CALCULATED 상태
핵심 로직	handle_digit() → model.input_digit()
예외 처리	• 최대 20자리까지 입력 제한 (len < 20) • 에러 상태 시 초기화 후 입력
사용자 UX 의도	• 계산 완료 후 새 숫자 입력 시 자동 초기화 • 지속적인 입력 흐름 유지 • 과도한 숫자 입력 제한 → 입력 오류 방지

SR_02. 소수점을 입력하고 중복을 방지하는 기능

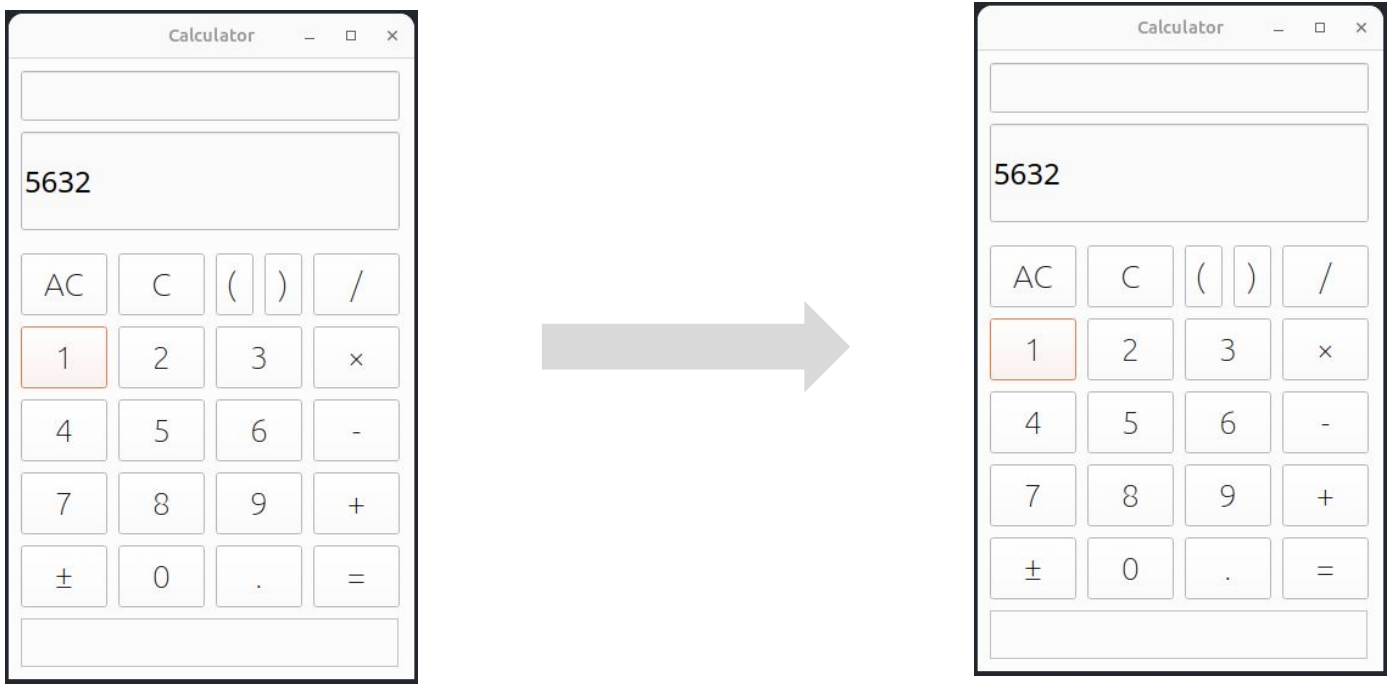


기능 요약	‘.’ 버튼 입력 시 ‘0.’ 또는 ‘-0.’으로 자동 보완 소수점 중복 입력 차단

SR_02. 소수점을 입력하고 중복을 방지하는 기능

적용 조건	READY, INPUTTING, CALCULATED 상태
핵심 로직	handle_digit(‘.’) → 내부에서 중복 여부 확인
예외 처리	• 이미 소수점 포함된 경우 ‘.’ 입력 무시
사용자 UX 의도	• ‘.’만 입력 시 ‘0.’으로 보완 → 혼란 방지 • 중복 소수점 차단으로 계산 오류 사전 방지 • 실제 계산기와 같은 소수점 입력 UX 제공

SR_03. ± 부호를 토글하는 기능

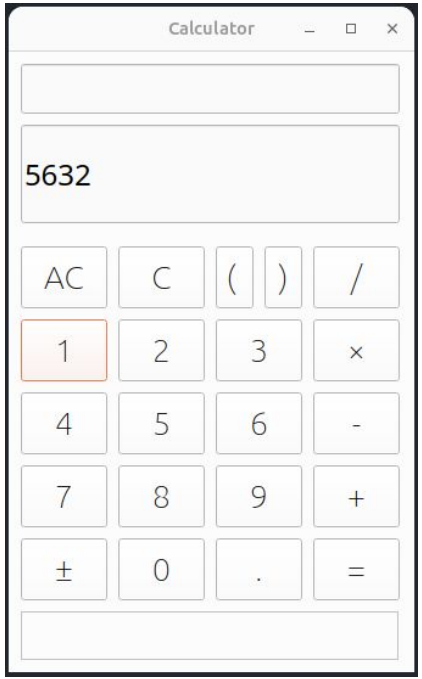


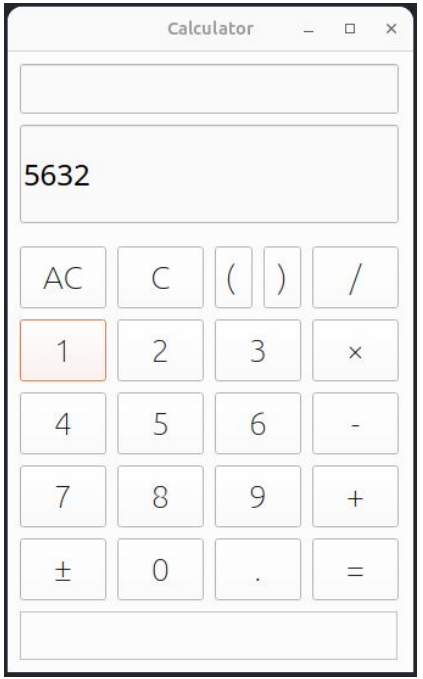
기능 요약	부호 전환 버튼 클릭 시 현재 입력 수에 ‘-’ 부호를 토글

SR_03. ± 부호를 토글하는 기능

적용 조건	READY, INPUTTING, CALCULATED 상태
핵심 로직	handle_sign() → model.toggle_sign()
예외 처리	• 0 입력 중일 경우 → ‘-’, 다시 눌러 0으로 복원
사용자 UX 의도	• 부호 토글을 통해 음수 입력을 명확하게 처리 • 계산기와 동일한 부호 전환 UX • 상태 유지 기반으로 부호만 바꾸고 숫자값은 그대로 유지

SR_04. 연산자를 입력하고 중복을 처리하는 기능



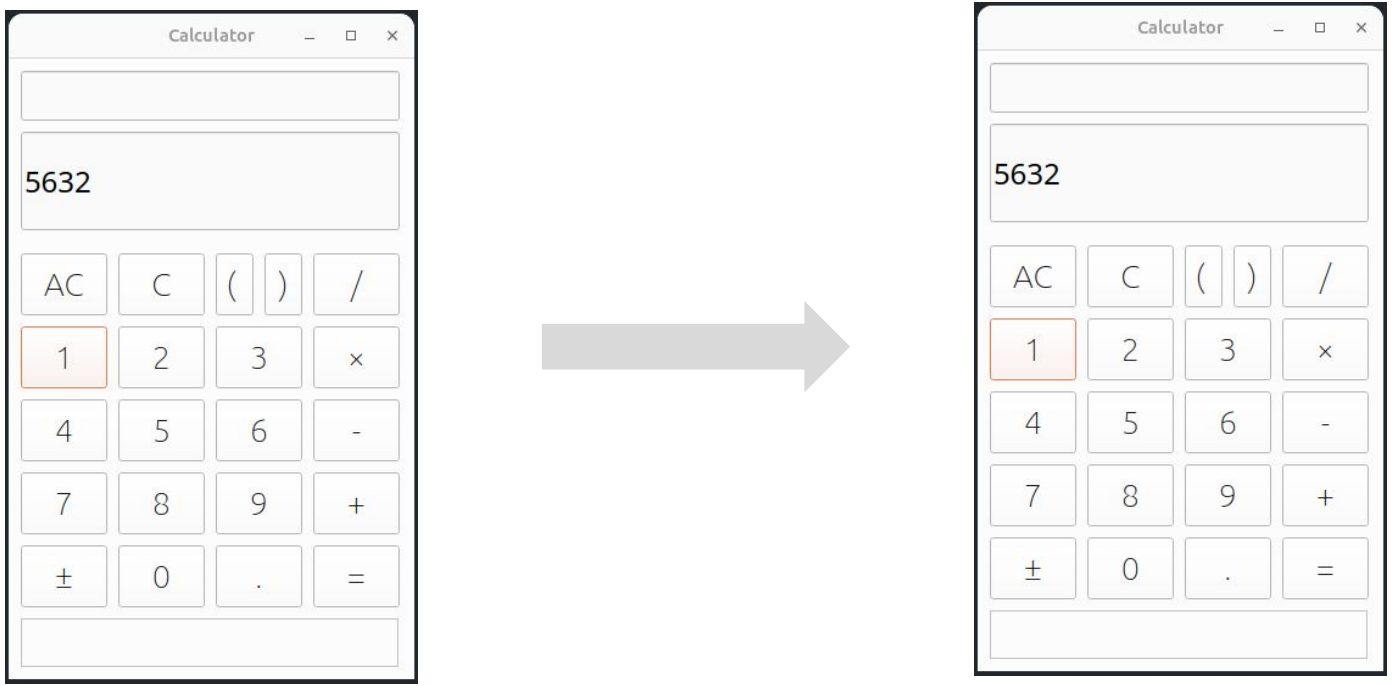


기능 요약	+, -, *, / 연산자 버튼 클릭 시 현재 수식 뒤에 연산자 추가 이전에 연산자가 있으면 새 연산자로 덮어씀

SR_04. 연산자를 입력하고 중복을 처리하는 기능

적용 조건	INPUTTING, CALCULATED 상태
핵심 로직	handle_operator() → model.input_operator()
예외 처리	• 연산자 연속 입력 시 마지막 연산자만 유지
사용자 UX 의도	• 연산자 중복 오류 자동 보정 • 부정확한 수식 입력 방지 • 실제 계산기처럼 직관적 처리 유지

SR_05. 괄호를 입력하고 곱셈을 자동 처리하는 기능

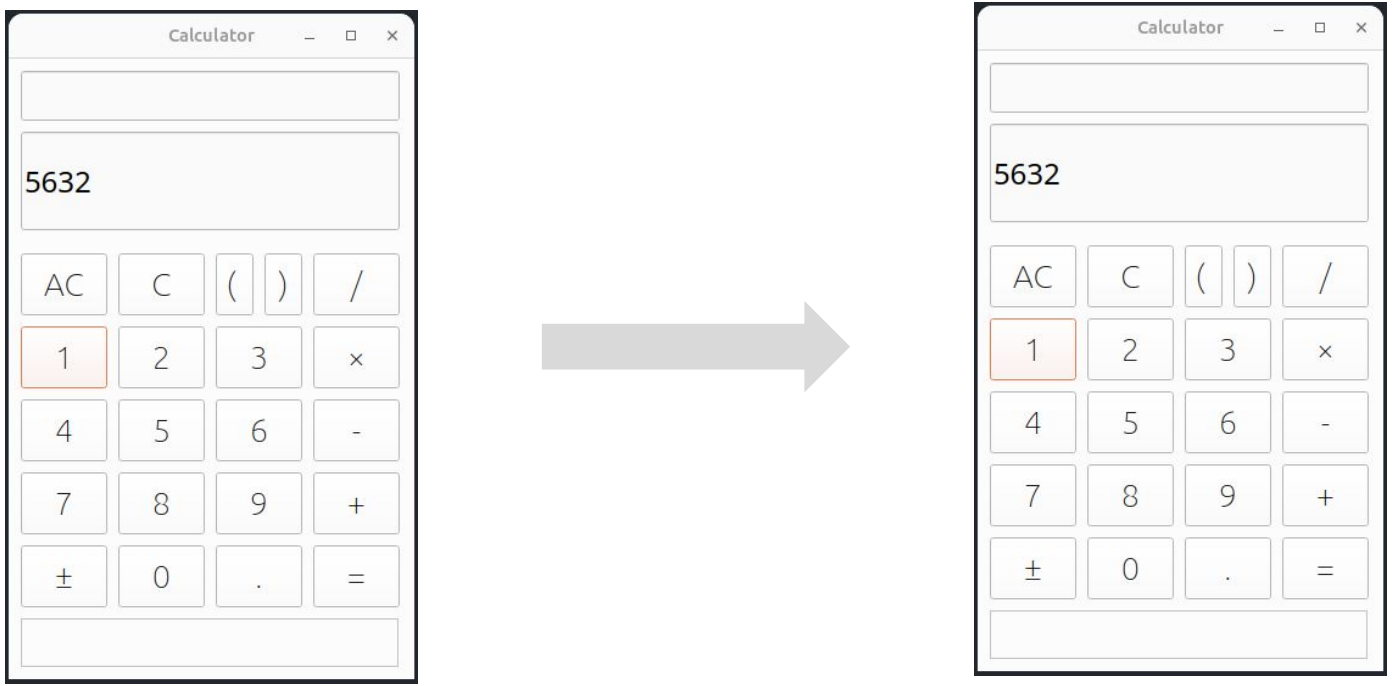


기능 요약	숫자 뒤 또는 닫는 괄호 뒤에 여는 괄호 입력시 * 자동 추가

SR_05. 괄호를 입력하고 곱셈을 자동 처리하는 기능

적용 조건	INPUTTING 상태
핵심 로직	handle_lparen() → model.input_lparen()
예외 처리	• 괄호 앞에 숫자/닫는 괄호 → ‘*’ 자동 삽입 • 시작 괄호는 ‘*’ 없이 입력
사용자 UX 의도	• 곱셈 생략을 자동으로 보완 • 수식 오류 방지 • 수학적 자연스러움 유지

SR_06. 괄호의 개수를 검증하고 빈 괄호를 방지하는 기능



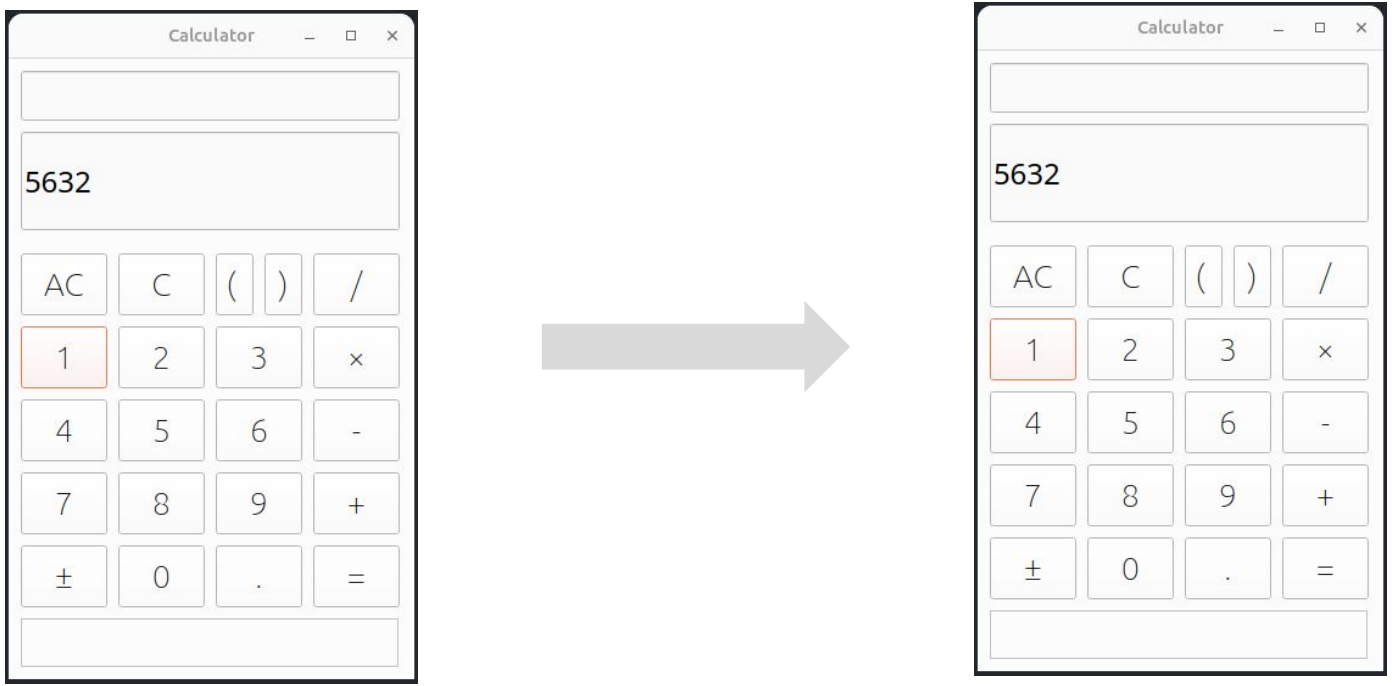
기능 요약	빈 괄호 입력 방지, 괄호 불균형 자동 보안

SR_06. 괄호의 개수를 검증하고 빈 괄호를 방지하는 기능

적용 조건	INPUTTING, CALCULATED 상태
핵심 로직	handle_rparen() 및 evaluate() 내부 괄호 처리
예외 처리	• 여는 괄호 뒤에 숫자 없이 닫는 괄호 입력 → 자동 제거 • 평가 전 괄호 자동 보정
사용자 UX 의도	• 빈 괄호로 인한 오류 방지 • 괄호 개수 실수 자동 보완 • 실 계산기 수준의 수식 완성 UX 제공

입력 및 상태 초기화 기능

SR_07. 계산 후 새로운 입력을 초기화하는 기능

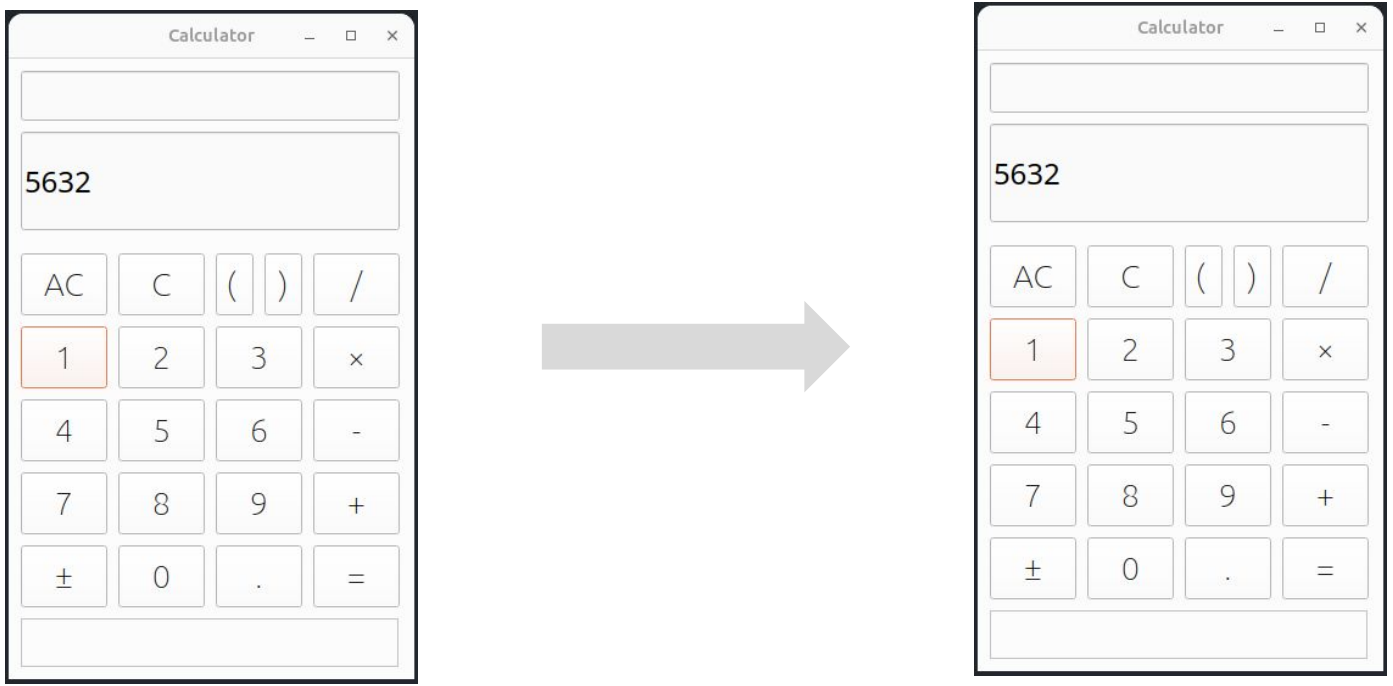


기능 요약	계산 완료 상태(CALCULATED)에서 숫자 입력 시 기존 수식을 초기화하고 새 입력 시작

SR_07. 계산 후 새로운 입력을 초기화하는 기능

적용 조건	CALCULATED 상태
핵심 로직	handle_digit() 내부에서 CALCULATED 상태 감지 → model.clear() 후 입력 시작
예외 처리	• 수식 유지가 필요한 경우(예: 연산자 입력)는 초기화하지 않음.
사용자 UX 의도	• 계산 후 새로 입력 시작 시 자동 초기화 • 입력 오류 방지 • 계산기와 유사한 직관적 흐름 제공

SR_08. 전체 초기화(AC) 또는 부분 삭제(C)를 수행하는 기능



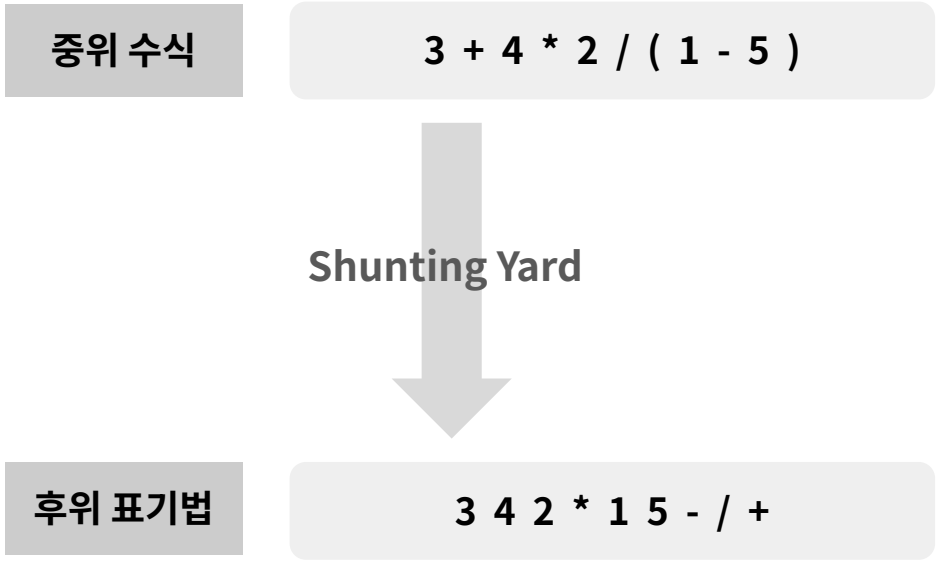
기능 요약	C 버튼은 최근 입력만 제거, AC 버튼은 수식 전체 초기화

SR_08. 전체 초기화(AC) 또는 부분 삭제(C)를 수행하는 기능

적용 조건	모든 상태
핵심 로직	handle_c(), handle_ac() → 각각 model.backspace() / model.reset()
예외 처리	• c는 current_input만, ac는 tokens까지 완전 초기화
사용자 UX 의도	• 잘못된 입력만 수정(C), 전체 수식 재입력 가능(AC) • 실제 계산기 기능 분리와 동일 • 입력 제어 유연성 확보

계산 처리 기능

SR_09. 중위 수식을 후위 표기법으로 변환하는 기능



기능 요약	Shunting Yard 알고리즘으로 중위 수식을 후위 표기법으로 변환

SR_09. 중위 수식을 후위 표기법으로 변환하는 기능

적용 조건	evaluate() 호출 시 내부 처리
핵심 로직	model.to_postfix() → 연산자 스택 / 출력 큐 구조 구현
예외 처리	• 괄호 처리 • 연산자 우선순위 • 입력 오류 방지
사용자 UX 의도	• 일반 수학 수식을 직관적으로 입력 가능 • 실제 수학 계산기 수준 연산 흐름 구현 • 연산자 우선순위 자동 반영으로 정확도 보장

SR_10. 후위 표기 수식을 스택으로 계산하는 기능

토큰	3	4	2	*	1	5	-	/	+
스택 상태						5			
			2		1	1	-4		
		4	4	8	8	8	8	-2	
	3	3	3	3	3	3	3	3	1

기능 요약	변환된 후위 표기 수식을 스택을 이용해 계산

SR_10. 후위 표기 수식을 스택으로 계산하는 기능

적용 조건	evaluate() 실행 후 to_postfix() 호출 성공 시
핵심 로직	model.evaluate_postfix()에서 연산자마다 2개 pop, 연산 후 push
예외 처리	• 피연산자 부족 • 0으로 나누기 • 스택에 1개 외 값 남을 경우
사용자 UX 의도	• 연산 순서를 명확히 정리 • 수식 오류 시 “Error” 표시 • 복잡한 수식도 정확히 계산 가능

SR_11. 괄호와 연산자를 자동 보완하는 기능

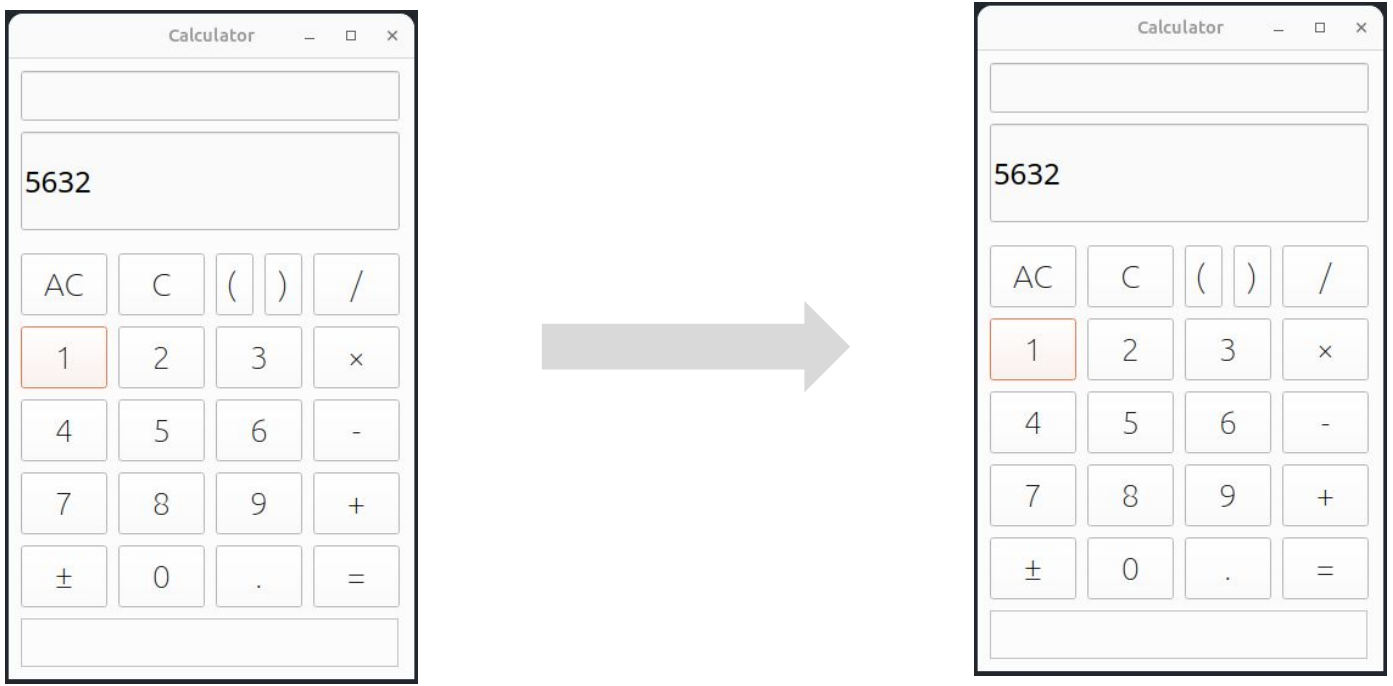
기능 요약	불완전한 괄호 또는 연산자 상태를 자동으로 보완하여 수식의 평가 가능성을 보장함

SR_11. 괄호와 연산자를 자동 보완하는 기능

적용 조건	수식 평가 전(evaluate()), 괄호 입력 시(handle_lparen/rparen())
핵심 로직	evaluate() 내 괄호 개수 자동 보완, input_rparen()에서 빈 괄호 검출 제거
예외 처리	• 여는 괄호 뒤 숫자 없이 닫는 괄호 입력 → 자동 삭제 • 수식 평가 시 괄호 개수 부족 → 자동 추가
사용자 UX 의도	• 괄호 오류로 인한 계산 실패 방지 • 불완전 수식도 자동 보완 • 수식 완결성을 유지하면서 입력 부담 완화

예외 처리 및 결과 출력 기능

SR_12. 에러 발생 시 오류 메시지를 출력하는 기능

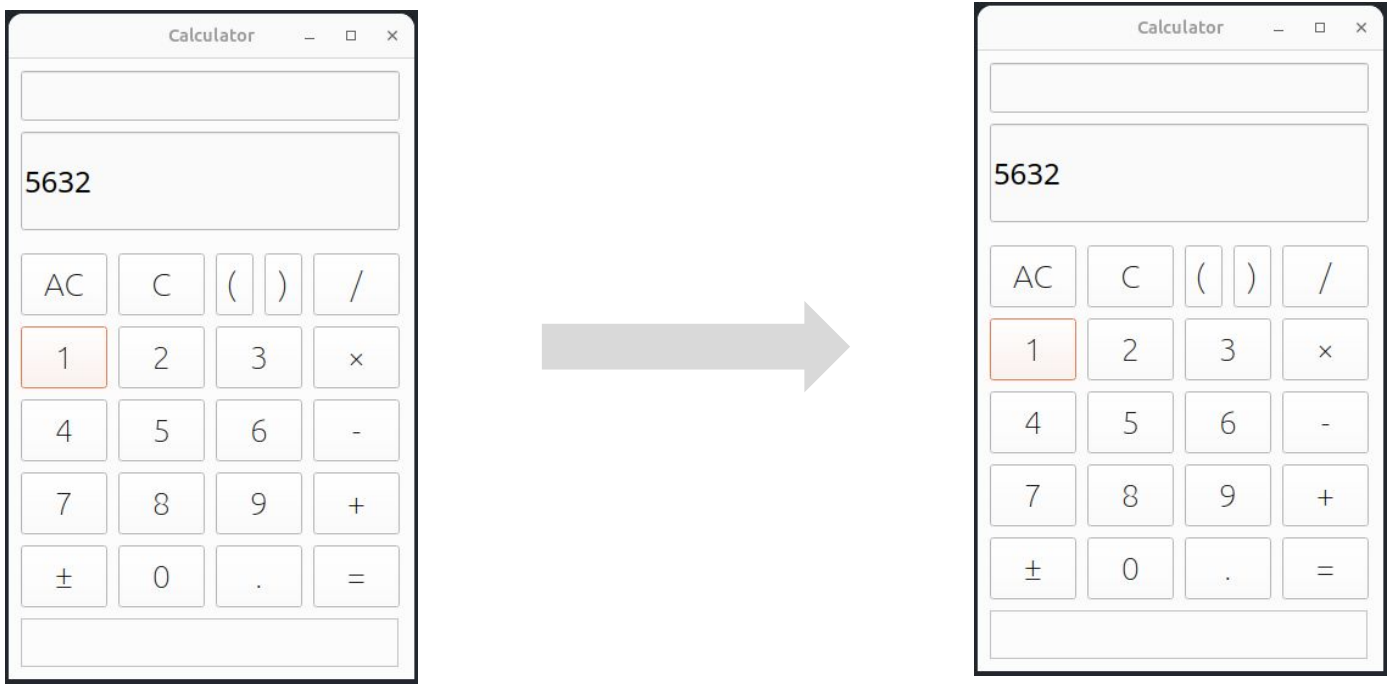


기능 요약	evaluate 중 오류 발생 시 사용자에게 메시지를 자동 출력하고 일정 시간 후 사라짐

SR_12. 에러 발생 시 오류 메시지를 출력하는 기능

적용 조건	evaluate() 실행 후 to_postfix() 호출 성공 시
핵심 로직	model.evaluate_postfix()에서 연산자마다 2개 pop, 연산 후 push
예외 처리	• 0으로 나누기 • 스택에 1개 외 값 남을 경우
사용자 UX 의도	• 연산 순서를 명확히 정리 • 수식 오류 시 “Error” 표시 • 복잡한 수식도 정확히 계산 가능

SR_13. 수식과 결과를 라인에 나누어 출력하는 기능

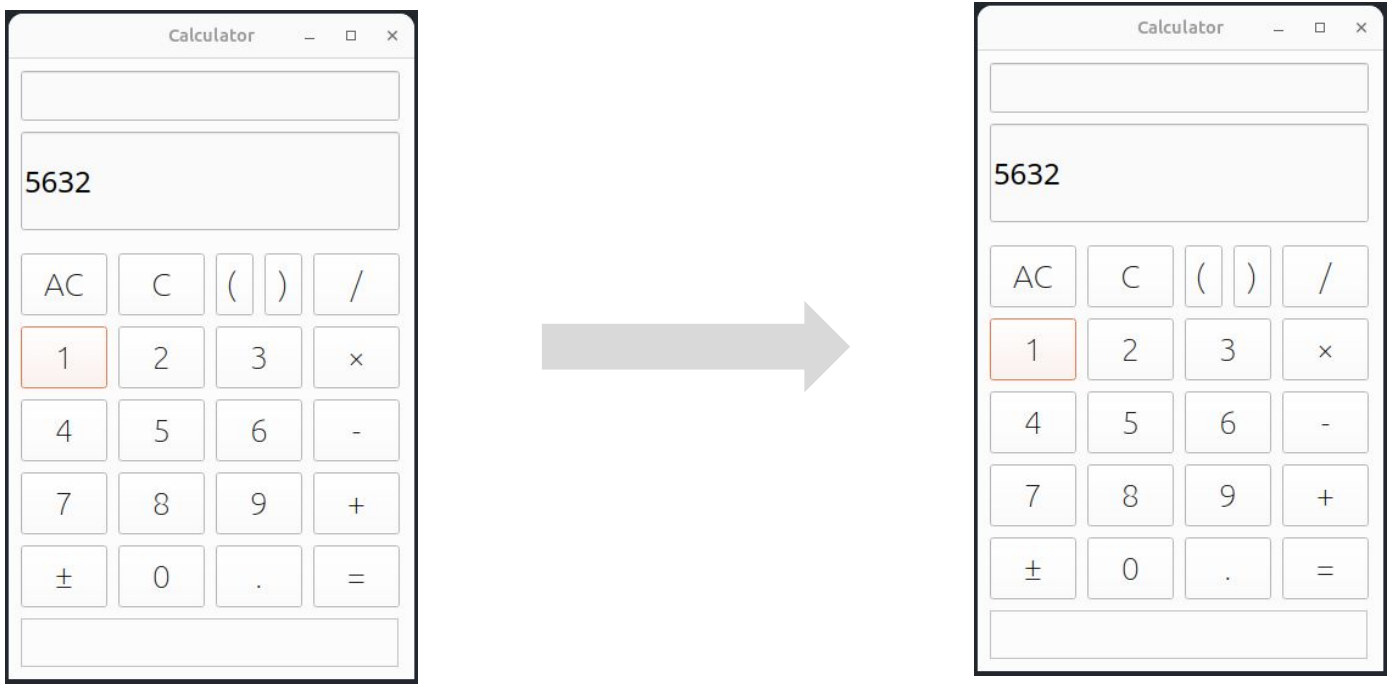


기능 요약	수식과 결과를 각각 다른 영역(lineEdit, lineEdit_2)에 분리하여 출력

SR_13. 수식과 결과를 라인에 나누어 출력하는 기능

적용 조건	모든 상태에서 update_display() 호출 시
핵심 로직	view.update_expression_display(tokens), view.update_result_display(current)
예외 처리	• 소수점 처리 (ex. 8.0 → “8”) • 결과가 없을 경우 “0” 표시
사용자 UX 의도	• 수식과 결과의 시각적 분리 • 계산 흐름을 더 직관적으로 이해할 수 있음 • 실제 계산기 디스플레이와 유사한 구조 유지

SR_14. 정수 결과 시 .0을 제거하여 출력하는 기능

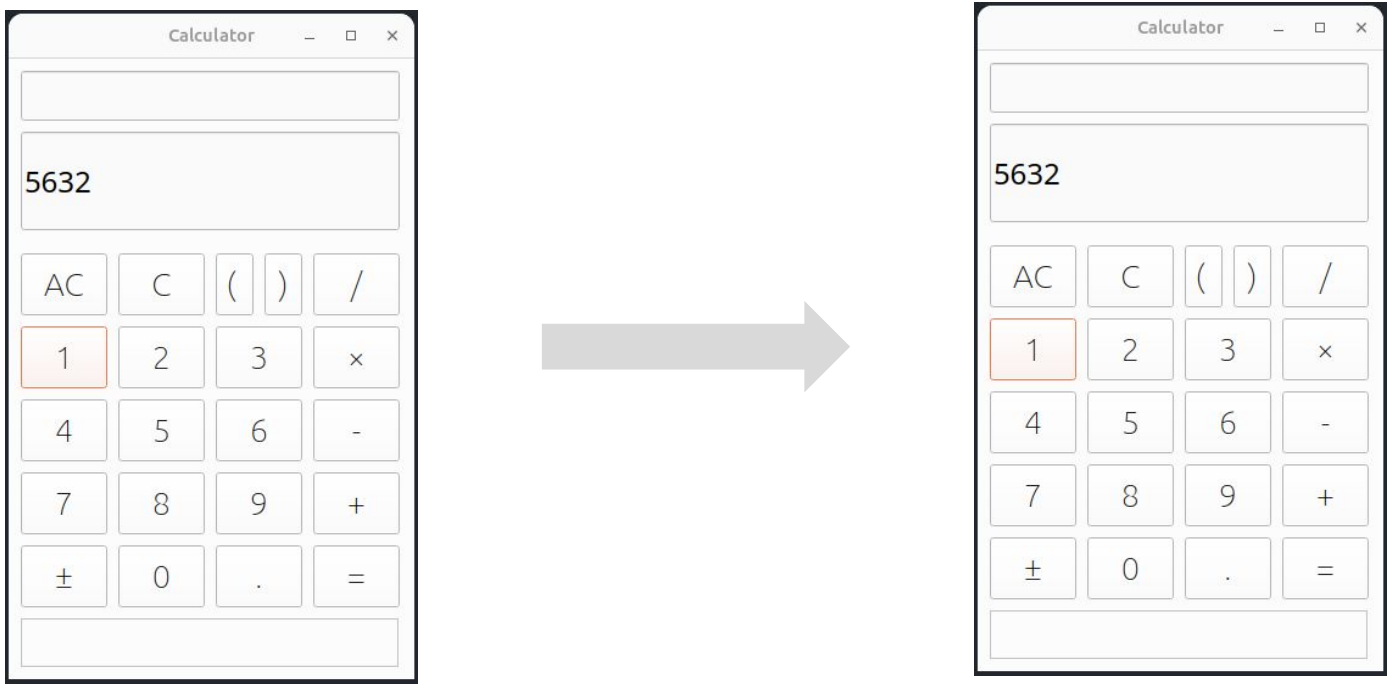


기능 요약	계산 결과가 .0으로 끝나는 정수일 경우, .0을 제거하여 깔끔하게 출력

SR_14. 정수 결과 시 .0을 제거하여 출력하는 기능

적용 조건	evaluated() 후 상태가 CALCULATED인 경우
핵심 로직	view.update_result_display() 내부에서 float → int 캐스팅 수행
예외 처리	• .0 외의 소수점 포함 수(예: 2.5)는 그대로 유지
사용자 UX 의도	• “5.0”이 아닌 ‘5’로 보여주는 직관적인 정리 • 결과 해석을 더 명확하게 하고, 계산기의 실제 출력 방식과 일치

SR_15. 입력 UX 보완 메시지를 출력하는 기능



기능 요약	사용자 입력 실수(소수점 중복, 연산자 누락 등)에 대해 즉시 메시지 출력

SR_15. 입력 UX 보완 메시지를 출력하는 기능

적용 조건	handle_digit(), handle_operator() 내 예외 코드 감지 시
핵심 로직	“TOO_LONG”, “MULTIPLE_DOT”, “NO_OPERATION” 등 문자열 반환 후 메시지 출력
예외 처리	• 소수점 중복, 숫자 뒤 연산자 누락, 입력 길이 초과 등
사용자 UX 의도	• 실수를 즉시 알려줌으로써 정정 기회를 제공 • **입력 → 오류 → 안내 → 재입력**의 자연스러운 입력 루프 형성

4. 테스트 및 검증

테스트 - 초기상태 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_01	초기상태 → 입력 없음	없음	없음	기본값(0)	✅
TC_02	초기상태 → AC 또는 C 입력	AC 또는 C	없음	기본값(0)	✅
TC_03	초기상태 → = 입력	=	없음	기본값(0)	✅
TC_04	초기상태 → (입력	(	(	없음	✅
TC_05	초기상태 →) 입력	)	없음	기본값(0)	✅
TC_06	초기상태 → ± 입력	±	없음	기본값(0)	✅
TC_07	초기상태 → 소수점 입력	.	없음	0.	✅
TC_08	초기상태 → 0 2회 입력	0 → 0	없음	기본값(0)	✅
TC_09	초기상태 → 0이 아닌 숫자 입력	1부터 9까지의 임의의 수	없음	입력과 동일	✅

테스트 - 오류 직후 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_10	오류 상태 → 숫자 입력	1/0= → 5	없음	5	✅
TC_11	오류 상태 → 연산자 입력	1/0= → +	없음	기본값(0)	✅
TC_12	오류 상태 → (입력	1/0= → (	(	없음	✅
TC_13	오류 상태 →) 입력	1/0= →)	없음	기본값(0)	✅
TC_14	오류 상태 → 소수점 입력	1/0= → .	없음	0.	✅
TC_15	오류 상태 → ± 입력	1/0= → ±	없음	기본값(0)	✅
TC_16	오류 상태 → C 입력	1/0= → C	없음	기본값(0)	✅
TC_17	오류 상태 → AC 입력	1/0= → AC	없음	기본값(0)	✅
TC_18	오류 상태 → = 입력	1/0==	없음	기본값(0)	✅

테스트 - 계산 직후 입력 처리

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_19	계산 직후 → 숫자 입력	1+2= → 5	없음	5	✅
TC_20	계산 직후 → 연산자 입력	1+2= → +	3 +	없음	✅
TC_21	계산 직후 → (입력	1+2= → (	3 * (	없음	✅
TC_22	계산 직후 →) 입력	1+2= →)	1+2	3	✅
TC_23	계산 직후 → 소수점 입력	1+2= → .	없음	0.	✅
TC_24	계산 직후 → ± 입력	1+2= → ±	없음	-3	✅
TC_25	계산 직후 → C 입력	1+2= → C	없음	기본값(0)	✅
TC_26	계산 직후 → AC 입력	1+2= → AC	없음	기본값(0)	✅
TC_27	계산 직후 → = 입력	1+2= → =	3	3	✅

테스트 - 우선순위 및 괄호

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_28	곱셈 우선	1+2*3=	1+2*3	7	✅
TC_29	괄호로 우선순위 변경	(1+2)*3=	(1+2)*3	9	✅
TC_30	중첩 괄호 계산	(1+(2+3))*2=	(1+(2+3))*2	12	✅
TC_32	괄호 안 곱셈 먼저	1+(2*3)=	1+(2*3)=	7	✅
TC_33	괄호만 사용	((3))=	((3))	3	✅
TC_34	괄호 생략 곱셈 테스트	2(3+4)=	2*(3+4)	14	✅
TC_36	연속 괄호 닫기	(1+2) →)	(1+2)	기본값(0)	✅
TC_37	괄호 안 연산자 끝남	(1+)=	(1)	1	✅

테스트 - 오류 감지

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_40	0으로 나누기	1/0=	1/0	Error	✅

테스트 - 오류 방지 및 오류 정정

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_41	괄호 이중 닫힘	(2+3) →)	(2+3)	없음	✅
TC_42	괄호 안 연산자로 끝남	(4+ →)	(4)	없음	✅
TC_43	연속 연산자 입력	5+ → -3=	5 - 3	2	✅
TC_44	소수점 중복 입력	2.3 → .	없음	2.3	✅
TC_39	연산자 끝 수식	5+ → =	5+	Error	✅

테스트 - 초기화

TC_ID	시나리오	입력	예상 결과(상단)	예상 결과(하단)	결과
TC_41	괄호 이중 닫힘	(2+3) →)	(2+3)	없음	✅
TC_42	괄호 안 연산자로 끝남	(4+ →)	(4)	없음	✅
TC_43	연속 연산자 입력	5+ → -3=	5 - 3	2	✅
TC_44	소수점 중복 입력	2.3 → .	없음	2.3	✅
TC_45	괄호 미닫힘	(3+(3+(3+3=	(3+(3+(3+3)))	12	✅