세상에서 제일 쉬운 러스트 프로그래밍

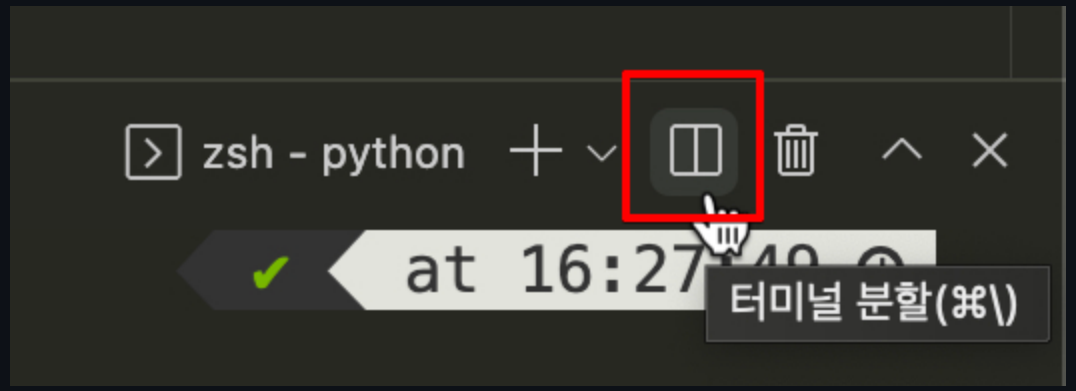
2장. 변수

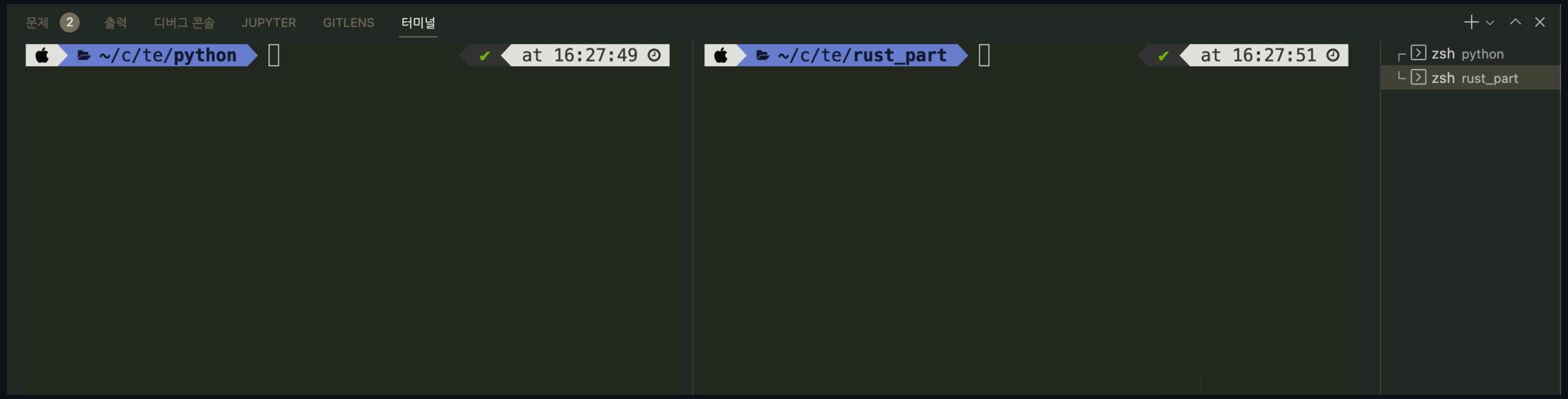
윤인도

freedomzero91@gmail.com

값 출력하기

터미널을 분할해서 사용하면 편리!





콘솔에 값 출력하는 방법

파이썬

```
print("Hello, world!")
```

러스트

```
println!("Hello, world!");
```

`!` 는 미리 만들어져 있는 기능을 의미하는 매크로(macro)를 호출한다는 기호

변수 선언

```python
x = 1.0
y = 10

print(f"x = {x}, y = {y}")
```

실행 결과

```
/code/temp/python $ python main.py
x = 1.0, y = 10
```

변수 선언 방법

```
 변수명   타입    값
let x: i32 = 10;
```

대부분의 경우 컴파일러가 타입을 추측

```
fn main() {
    let x: f64 = 1.0;
    let y = 10; // 타입 추론

    println!("x = {}, y = {}", x, y);
}
```

하위 폴더인 "rust_part" 폴더로 이동한 다음, `cargo run` 을 실행해 결과를 확인해보겠습니다.

```
/code/temp/rust_part $ cargo run
```

실행 결과

```
x = 1, y = 10
```

작명 규칙

파이썬과 러스트의 작명 규칙은 거의 동일합니다.

	파이썬	러스트
변수	`snake_case = 3`	`let snake_case = 3;`
함수	`def my_function`	`fn my_function`
클래스/구조체	`class MyClass`	`struct MyStruct`
상수	`SCREAMING_SNAKE_CASE = 1`	`const SCREAMING_SNAKE_CASE: i32 = 1;`

불변성

파이썬의 변수는 언제든 다른 타입의 값을 넣을 수 있습니다.

```
x = 1
x = "2"
x = 3.141592
```

러스트의 모든 변수는 기본적으로 불변(immutable)입니다.

```rust
fn main() {
    let x = 1;
    x = 2; // won't compile!
    println!("{}", x);
}
```

위 코드를 실행해보면 다음과 같은 에러가 발생합니다.

```
error[E0384]: cannot assign twice to immutable variable `x`
 --> src/main.rs:3:5
  |
2 |     let x = 1;
  |         -
  |         |
  |         first assignment to `x`
  |         help: consider making this binding mutable: `mut x`
3 |     x = 2; // won't compile!
  |     ^^^^^ cannot assign twice to immutable variable
```

컴파일러의 조언에 따라 변수 `x`를 선언 시 `mut` 키워드를 추가해 가변 변수로 선언하면 됩니다.

```
let mut x = 1;
```

수정된 코드를 아래와 같이 작성하고 실행해봅시다.

```
fn main() {
    let mut x = 1;
    x = 2;
    println!("{}", x);
}
```

섀도잉

한번 선언한 불변 변수의 값을 변경하는 것은 불가능하지만, 변수 자체를 새로 선언하는 것은 가능합니다. 이렇게 변수 이름을 재사용해서 새로운 변수를 다시 선언하는 것을 섀도잉(shadowing)이라고 합니다.

```
fn main() {
    let x = "5";

    let x = 6; // x를 6으로 재선언

    println!("The value of x is: {}", x); // 6
}
```

타입

C언어 계열과 마찬가지로, Rust는 타입이 존재합니다. 러스트의 원시 타입(primitive type) 목록은 다음과 같습니다.

이름	타입	이름	타입
8비트 정수	`i8`	부호 없는 32비트 정수	`u32`
16비트 정수	`i16`	부호 없는 64비트 정수	`u64`
32비트 정수	`i32`	부호 없는 128비트 정수	`u128`
64비트 정수	`i64`	부호 없는 아키텍처	`usize`
128비트 정수	`i128`	불리언	`bool`
아키텍처	`isize`	문자열	`String`
부호 없는 8비트 정수	`u8`	문자열 슬라이스	`str`
부호 없는 16비트 정수	`u16`	32비트 부동소수점 실수	`f32`
		64비트 부동소수점 실수	`f64`

타입 추론

다음 코드를 VSCode에 붙여넣으면 아래 그림과 같이 타입이 추론되는 것을 볼 수 있습니다.

```
fn main(){
    let x = 1;
    let y = 1.0;
    println!("{} {}", x, y);
}
```

```
fn main(){
    let x: i32 = 1;
    let y: f64 = 1.0;
    println!("{} {}", x, y);
}
```

타입 캐스팅

변수의 타입을 다른 타입으로 바꾸는 타입 변환(Casting)도 가능합니다. 파이썬에서는 타입 이름을 바로 사용해 타입 변환을 수행합니다.

```
x = 1.2
y = int(x)
print(f"{x} -> {y}");
```

실행결과

```
1.2 -> 1
```

러스트에서는 아래와 같이 `as` 키워드를 사용하면 됩니다.

```
fn main() {
    let x: f64 = 1.2;
    let y = x as i32;
    println!("{} -> {}", x, y);
}
```

실행결과

```
1.2 -> 1
```

상수

상수(constant)란, 한 번 선언되면 값이 바뀌지 않는 변수를 의미합니다. 먼저 파이썬에서 상수를 다음과 같이 선언해 보겠습니다.

```
THRESHOLD = 10


def is_big(n: int) -> bool:
    return n > THRESHOLD


if __name__ == '__main__':
    print(THRESHOLD)
    print(is_big(THRESHOLD))

    THRESHOLD = 5
```

상수 선언 시 반드시 타입을 표기해주어야 합니다.

```
const THRESHOLD: i32 = 10;

fn is_big(n: i32) -> bool {
    n > THRESHOLD
}

fn main() {
    println!("{}", THRESHOLD);
    println!("{}", is_big(5));
}
```

실행결과

```
10
false
```

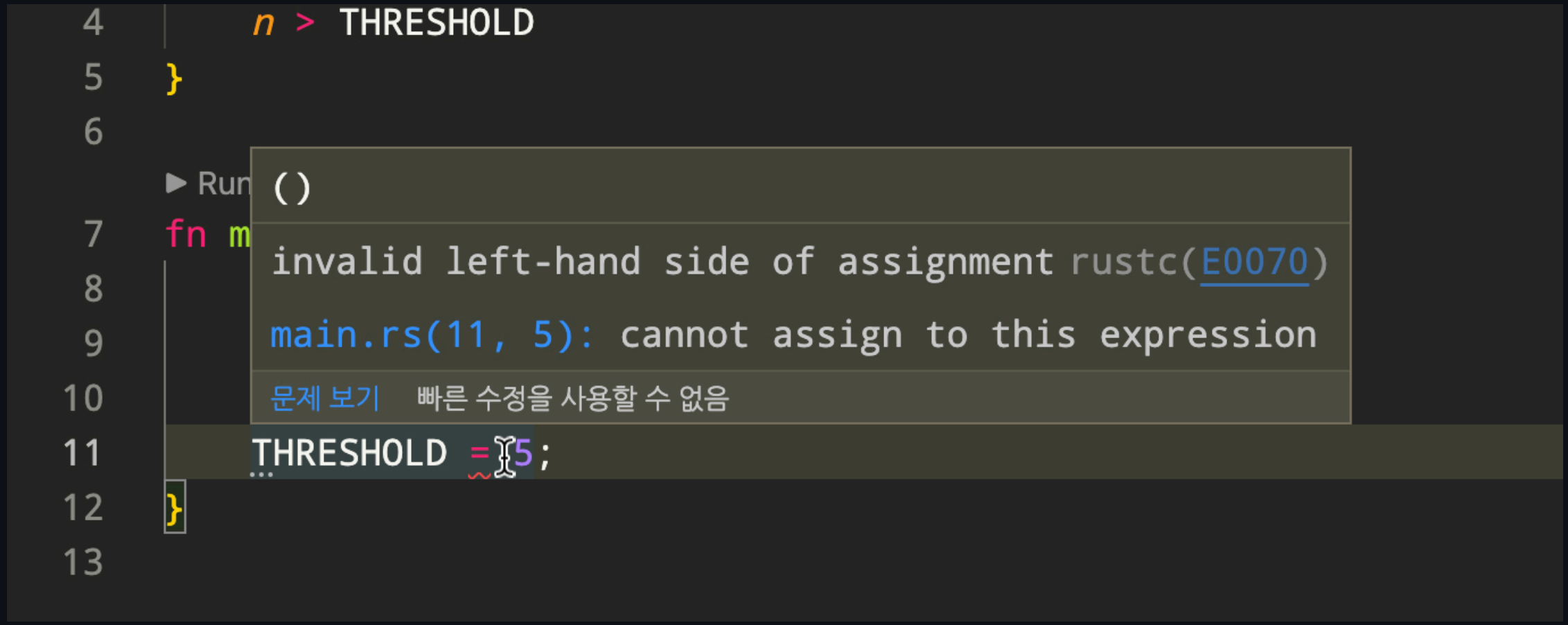
값이 불변이기 때문에 `THRESHOLD = 5;`와 같이 새로운 값을 할당하게 되면 오류가 발생합니다.

```
...
    THRESHOLD = 5;
...
```

실행결과

```
  --> src/main.rs:11:15
   |
11 |     THRESHOLD = 5;
   |     --------- ^
   |     |
   |     cannot assign to this expression
```

컴파일러가 친절하게 상수 `THRESHOLD` 에는 새로운 값을 할당할 수 없다고 알려주게 됩니다. 실행하기 전 편집기 안에서도 빨간 줄로 해당 코드에 문제가 있음을 알려주기 때문에 문제를 빠르게 찾고 해결할 수 있습니다.



연습문제

1. 다음 코드를 실행해보고, 컴파일러가 에러를 발생시키는 이유를 설명해보세요.

```
fn main() {
    let x = 1;
    x = 2;
    println!("{}", x);
}
```

불변 변수에는 새로운 값을 할당할 수 없기 때문에 에러가 발생합니다.

```
error[E0384]: cannot assign twice to immutable variable `x`
 --> src/main.rs:3:5
  |
2 |     let x = 1;
  |         -
  |         |
  |         first assignment to `x`
  |         help: consider making this binding mutable: `mut x`
3 |     x = 2;
  |     ^^^^^ cannot assign twice to immutable variable

For more information about this error, try `rustc --explain E0384`.
```

2. 다음 코드를 실행해보고, 컴파일러가 에러를 발생시키는 이유를 설명해보세요.

```
fn main() {
    let x = 1;
    let y = 2;
    x += y;
    println!("{}", x);
}
```

```
error[E0384]: cannot assign twice to immutable variable `x`
 --> src/main.rs:4:5
  |
2 |     let x = 1;
  |         -
  |         |
  |         first assignment to `x`
  |         help: consider making this binding mutable: `mut x`
3 |     let y = 2;
4 |     x += y;
  |     ^^^^^^ cannot assign twice to immutable variable

For more information about this error, try `rustc --explain E0384`.
```

3. 다음 코드가 컴파일되어 결과가 3이 나오도록 타입 캐스팅을 추가하세요.

```
fn main() {
    let x = 1.2;
    let y = x;
    let z = 2;
    println!("y + z = {}", y + z);
}
```

정답:

```
fn main() {
    let x = 1.2;
    let y = x;
    let z = 2;
    println!("y + z = {}", y as i32 + z);
}
```

4. 다음 코드가 컴파일되도록 수정하세요.

```
const PI = 3.14;

fn main() {
    println!("원주율: {}", PI);
}
```

정답:

```
const PI: f64 = 3.14;

fn main() {
    println!("원주율: {}", PI);
}
```