Pattern matching

1 Patterns

- May introduce shadowing
- irrefutable: always matches the value it's matched against
  ⇒ only contains identifiers or wildcards _
  `let (x, y) = (1, 2);`
- refutable: possibly not matching the value it's matched against
  ⇒ contains literals
  `if let (a, 2) = (1, 2) {...}`
  `else if let Some(x) = 12 {...}`
- combine them with a pipe `|`
  `if let 1 | 2 = x => {...}`

Where they can be used: Irrefutable:

- `let`-statements
- `for`-loops
- `fn` parameters
- Closures

Refutable

- `match`-Arms
- `if let`-expressions
- `while let`-loops

Kinds of Patterns Literal & Wildcard Patterns

- match same value (literals) or everything (wildcards)
- `_` for single, `..` for multiple ignored literals
- always refutable
- Floating-point literals are going to be forbidden in a future version

Range Patterns

- work for integers, characters & floating-points (deprecated)
- `a..=b` means from a to (inclusive) b with $a \leq b$
- irrefutable when spanning the whole type domain, else refutable

Grouped & Slice Patterns

- used to control operator precedence where'd be ambigous
- fixed size arrays or dynamic collections
- subslicing of slices to be stabilized e.g. `[a, ..]` will not work

Identifier Patterns

- identifier patterns bind value they match to a variable
  `let mut variable = 10;`
- @ syntax binds what matched a pattern
- identifier patterns bind by default to copy or move depending on presence of `Copy`-Trait
- use `ref` and `mut ref` to bind identifier to reference to the value's memory location
- require `ref` due to destructing patterns not allowing `&` to be

Identifier patterns: Binding modes Automatically convert non-references to `mut ref` or `ref`

- Default binding mode: move semantics
- on match of reference and non-reference patterns; deref and update binding mode:
  - move
  - ref or ref mut
  - if ref was reached it stays in ref.

Reference Patterns

- deref pointer that is beeing matched
  ⇒ Borrow them
- always refutable

Struct Patterns

- can be used to destructure a data type
- when destructuring, all fields need to be addressed or ignored by `_` or `..`
- is refutable when one of its subpatterns is refutable.

Tuple struct & Tuple Patterns both follow struct patterns analogous

Path Patterns May refer to:

- enum variants
- structs
- constants
- associated constants

Irrefutable for structs and enums with one variant, else refutable

2 Match

- Branch on Expression compared to patterns
- comparable to a `switch` statement but more powerfull
- Use Match Guards to further refine branching conditions

3 Exercise

Write code that describes a Person with a name and an age. Check wether the person is under 6, between 7 and 12, between 13 and 18 or older. Print the age of the person.