RE & Negation

- Hi. My name is Tony and I’m a computer scientist
- Armchair interests here
    - Computational linguistics
    - PIE

RE & Negation

- REs in programming are about ***pattern-matching***
- Matching numbers (decimal integer numerals) with REs
    - Numeral = string
    - Number = interpretation of numeral
- Informal: 0, or a decimal digit [1-9] maybe followed by digits [0-9], e.g.
    - 7
    - 8675309
    - 043 × (but valid octal / base-8)
    - 3.1415927 × (but valid floating-point / real)
    - 299792458

RE & Negation

- How to match (accept) with RE?
  - `[123456789][0123456789]*`
  - Programmers are lazy, above is too austere ⇒ use range shortcut
  - `[1-9][0-9]*`
  - Lazier ⇒ use meta-character shortcut
  - `[1-9]\d*`
  - If string is meant like an ID# then could perhaps relax first digit
  - `\d\d*`
  - Even lazier! Simply 1-or-more
  - `\d+`

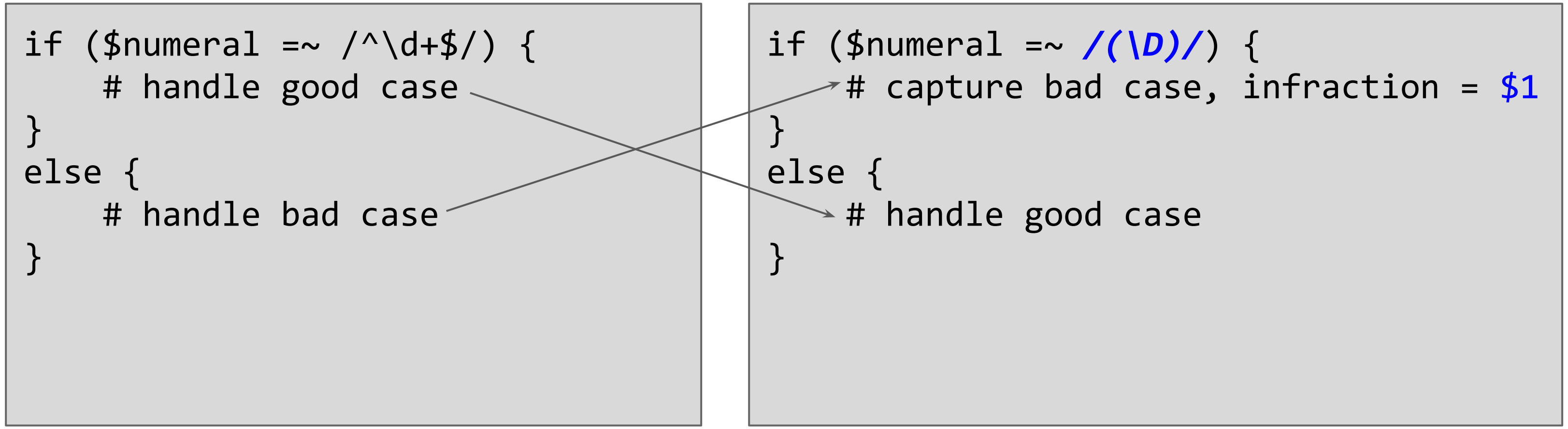
RE & Negation

- Numeral must be digits all the way
  - `123a4`
  - Is not a numeral
    - (well, it *contains* 2 of them, but the entire thing has an infraction @ “a”)
- Anchors:
  - “^” = start of string
  - “$” = end of string
- So
  - ^\d+$

RE & Negation

- But there’s a better/easier way!
  - Use test- negation / inversion (complement)
  - If *any 1 character anywhere* is NOT a digit, then we fail
  - So we can just match
    - `\D =` any single character NOT a digit
    - Austere: [^0123456789]
    - Or is this just … confusion?
      - Here “^” isn’t an anchor, it also means “not” at the beginning of a character class
        - […]
  - Notes
    - No Kleene star
    - No anchors
    - First infraction = immediate fail

RE & Negation



```
if ($numeral =~ /^\d+$/) {
    # handle good case
}
else {
    # handle bad case
}
```

```
if ($numeral =~ /(\D)/) {
    # capture bad case, infraction = $1
}
else {
    # handle good case
}
```