Healthy side of functional programming

Bartłomiej Górny
bartlomiej.gorny@erlang-solutions.com

CODER’S HEALTH ISSUES

- Wrists
- Spine
- Eyes
- stress

CODER’S HEALTH ISSUES

- Wrists
- Spine
- Eyes
- stress

CODER'S HEALTH ISSUES

- Wrists

- Spine
- Eyes
- stress

$\}$

CODER’S HEALTH ISSUES

- Wrists

$$\left.\begin{array}{l}\text{▸ Spine}\\ \text{▸ Eyes}\\ \text{▸ stress}\end{array}\right\} \longrightarrow \text{▸ Efficient use of resources}$$

CODER’S HEALTH ISSUES

- Wrists

- Spine
- Eyes
- stress

- Efficient use of resources
- Brain organisation vs code structure

WRISTS AND FINGERS

- RSI, CTS
- Keyboards
- Trackball
- Less typing

DOES FP MEAN LESS TYPING?

- QuickCheck implementations in various languages
  - (in thousands of lines)

Clojure	6.7
Haskell	9.6
C	12.2
C++	14.2
Java	17.9
JavaScript	18.5
PHP	20.1
.NET	77.4

MAKING WORK EASIER

- Use brain efficiently
  - Reduce effort and stress
- What are we best at?

MAKING WORK EASIER

- Use brain efficiently
  - Reduce effort and stress
- What are we best at?

PATTERN RECOGNITION

MAKING WORK EASIER

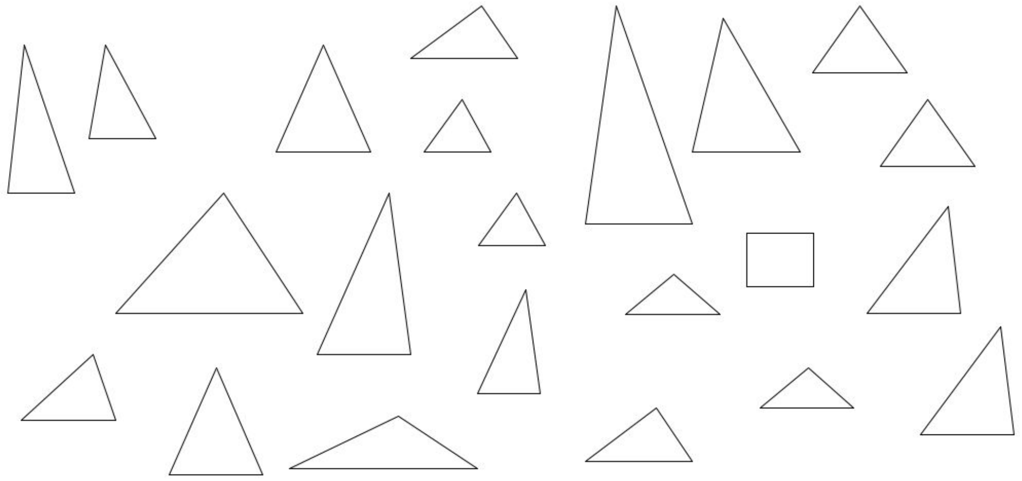
- Use brain efficiently
  - Reduce effort and stress
- What are we best at?

PATTERN RECOGNITION

- Evolutionary optimisation
  - Finding animal tracks
  - Searching edible plants
  - Recognising expressions (hostile or friendly?)
  - Danger alerts

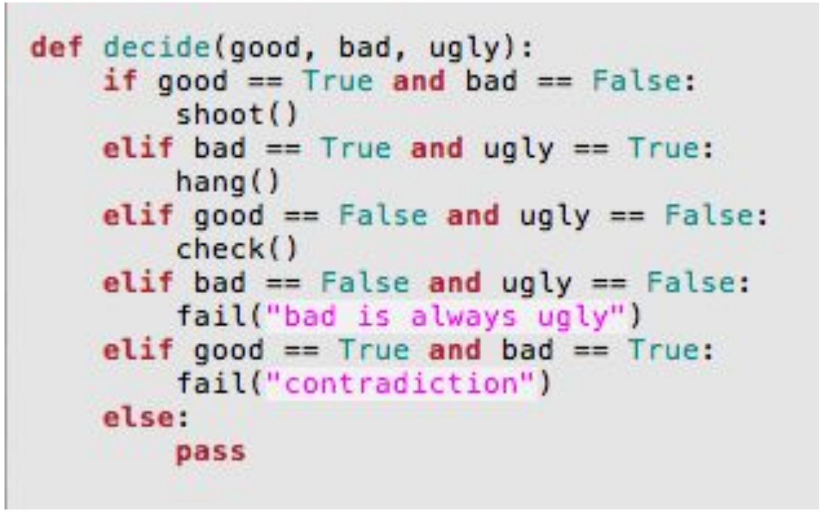
PATTERN RECOGNITION

Can you spot what’s wrong with this picture?

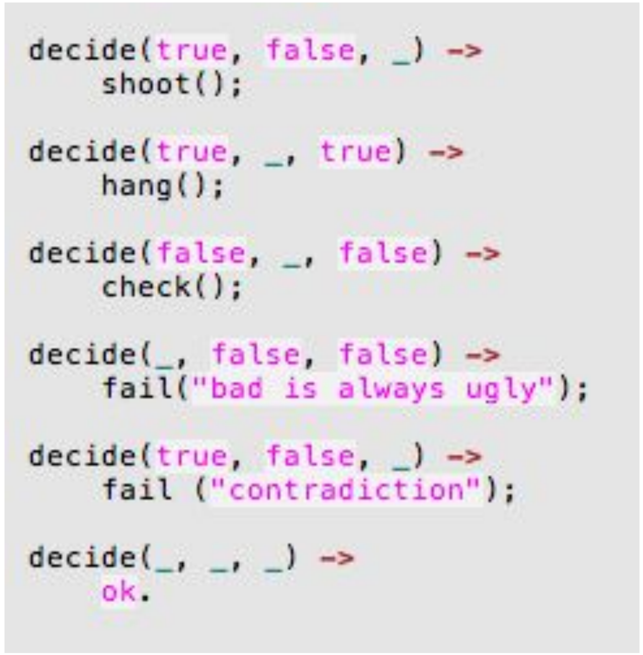


How long did it take you?

PATTERN RECOGNITION / MATCHING



```python
def decide(good, bad, ugly):
    if good == True and bad == False:
        shoot()
    elif bad == True and ugly == True:
        hang()
    elif good == False and ugly == False:
        check()
    elif bad == False and ugly == False:
        fail("bad is always ugly")
    elif good == True and bad == True:
        fail("contradiction")
    else:
        pass
```



```erlang
decide(true, false, _) ->
    shoot();

decide(true, _, true) ->
    hang();

decide(false, _, false) ->
    check();

decide(_, false, false) ->
    fail("bad is always ugly");

decide(true, false, _) ->
    fail ("contradiction");

decide(_, _, _) ->
    ok.
```

PATTERN RECOGNITION - OTHER EXAMPLES

- Chess
    - Analysis, strategic planning, forecasting
    - Recognising patterns on board
    - RPD (Recognition Primed Decisions)

RECOGNITION PRIMED DECISIONS

- Based on matching input to an in-memory pattern
- Done instantly without conscious thinking
- Often based on episodic patterns

RECOGNITION PRIMED DECISIONS

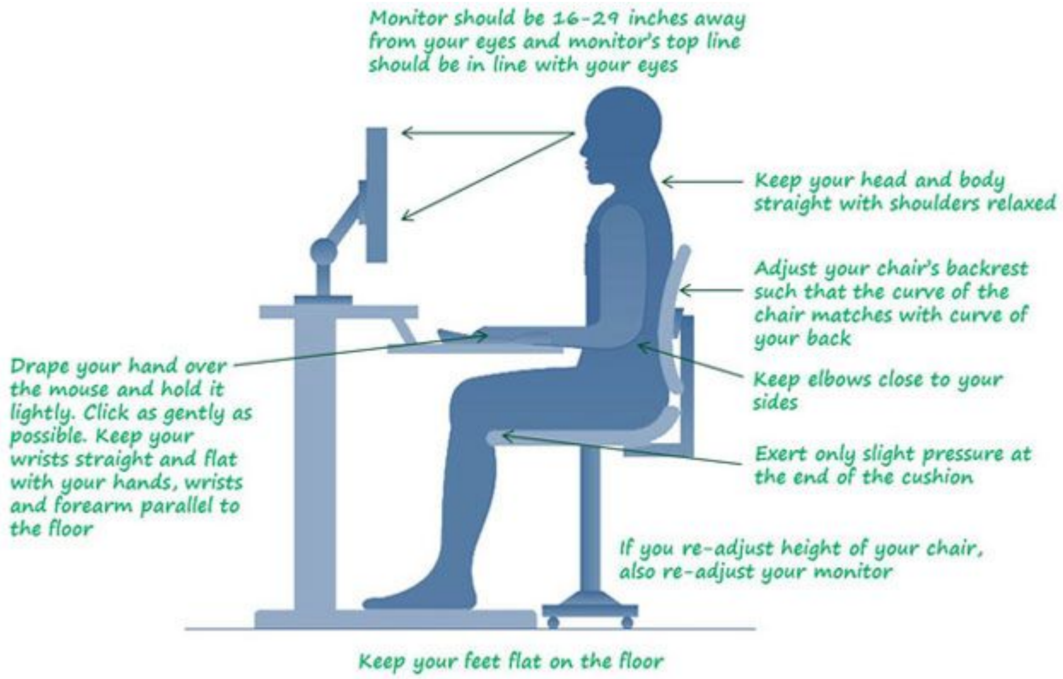
- Based on matching input to an in-memory pattern
- Done instantly without conscious thinking
- Often based on episodic patterns

- …some call it “experience”

IN SEARCH FOR THE RIGHT POSTURE

- …and how FP can help with that

SPINE - THEORY


Monitor should be 16-29 inches away from your eyes and monitor's top line should be in line with your eyes
Keep your head and body straight with shoulders relaxed
Adjust your chair's backrest such that the curve of the chair matches with curve of your back
Keep elbows close to your sides
Exert only slight pressure at the end of the cushion
If you re-adjust height of your chair, also re-adjust your monitor
Drape your hand over the mouse and hold it lightly. Click as gently as possible. Keep your wrists straight and flat with your hands, wrists and forearm parallel to the floor
Keep your feet flat on the floor

PRACTICE - COMMON TYPING POSITION

COMMON CODE READING POSITION

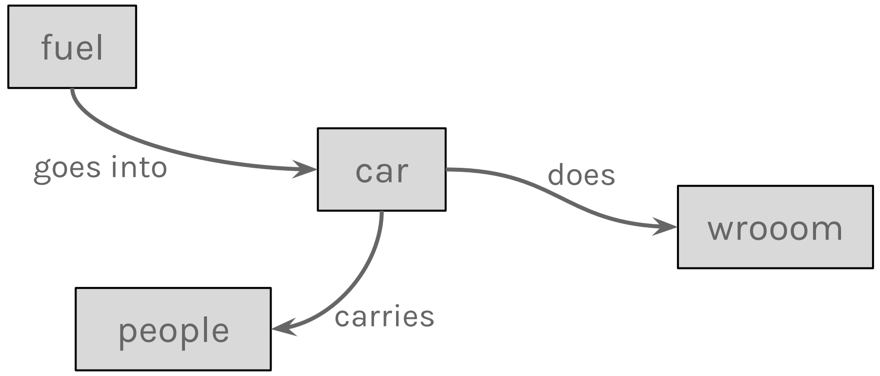
THINKING

READING AND UNDERSTANDING

- We do much more reading than writing
  - Legacy code, morning bootstrap, code reviews…
- We have to understand the code

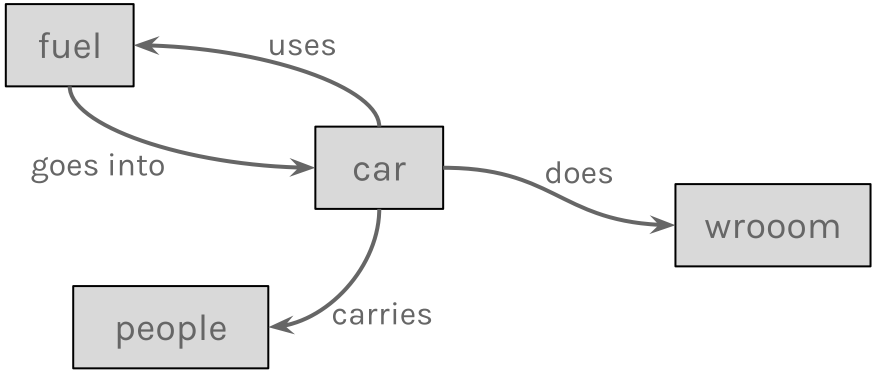
READING AND UNDERSTANDING

- To understand = to build a concept network



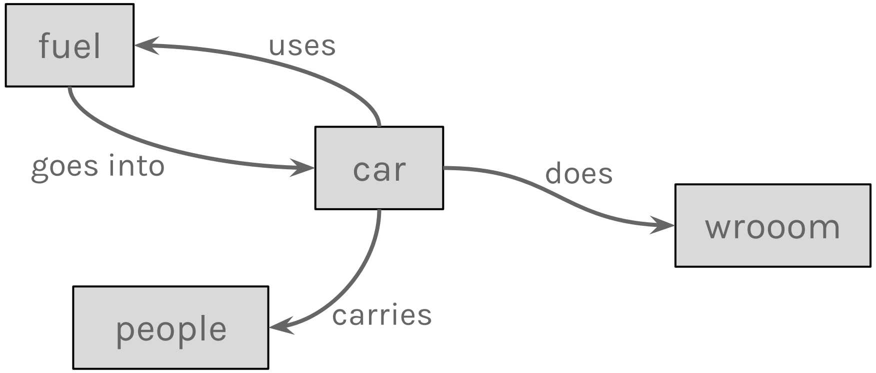
READING AND UNDERSTANDING

- To understand = to build a concept network



READING AND UNDERSTANDING

- To understand = to build a concept network



- Thinking is (re)building concept network

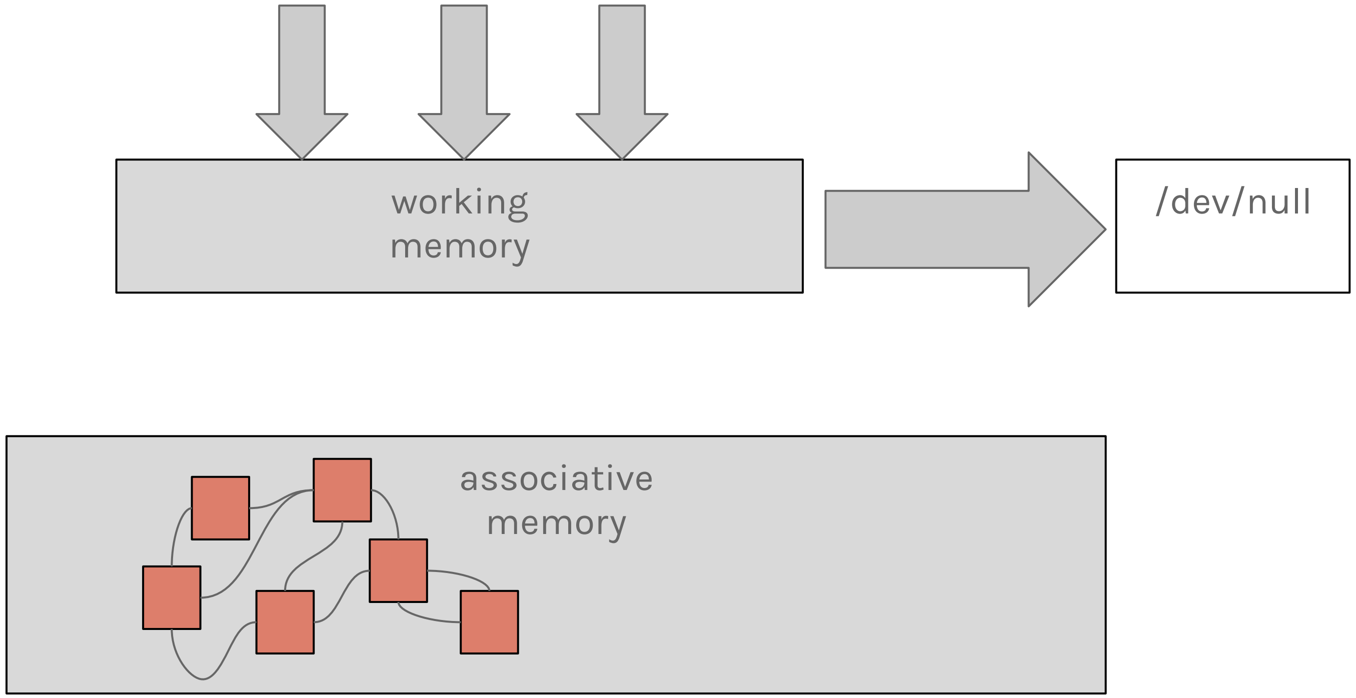
WHERE DO CONCEPTS COME FROM?

- Brain processes a stream of input
- Concepts must be:
  - Isolated
  - Formulated
  - stored

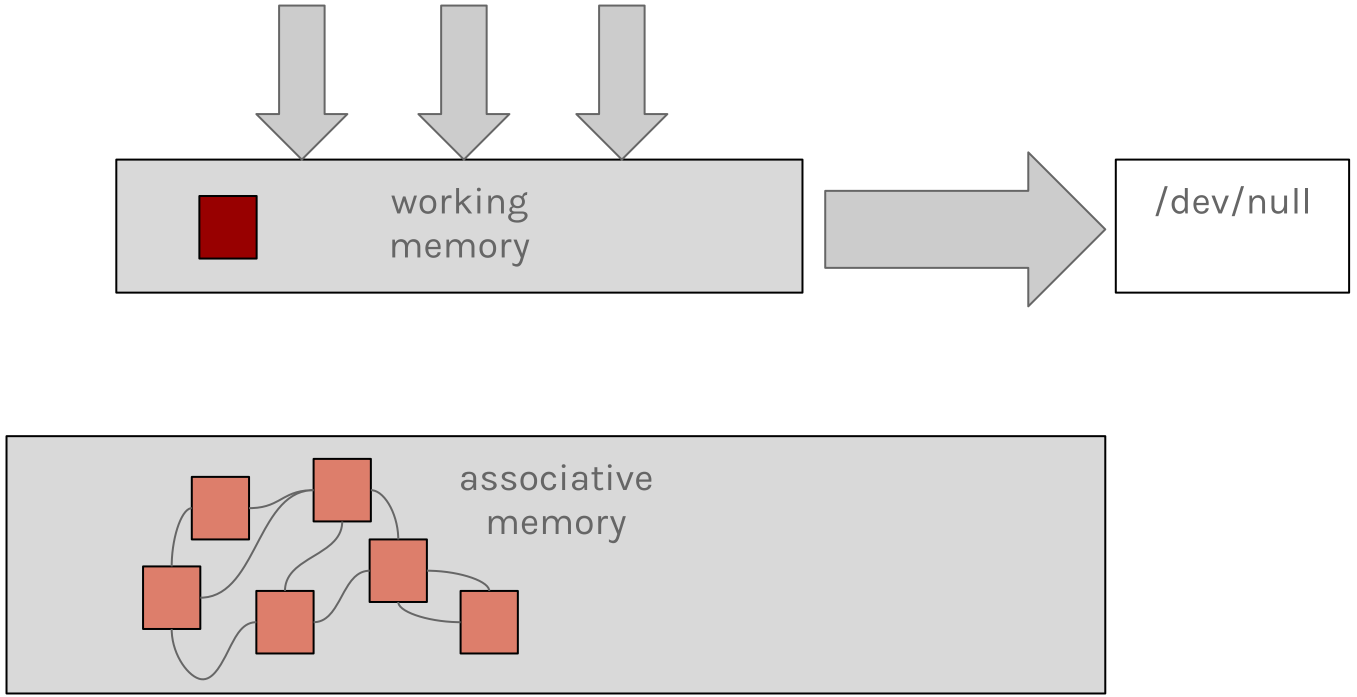
MEMORY STRUCTURE

- Working memory
  - “Input buffer”
  - High throughput
  - Low capacity
  - Volatile
- Associative memory
  - Low throughput
  - Unknown (infinite?) capacity
  - persistent

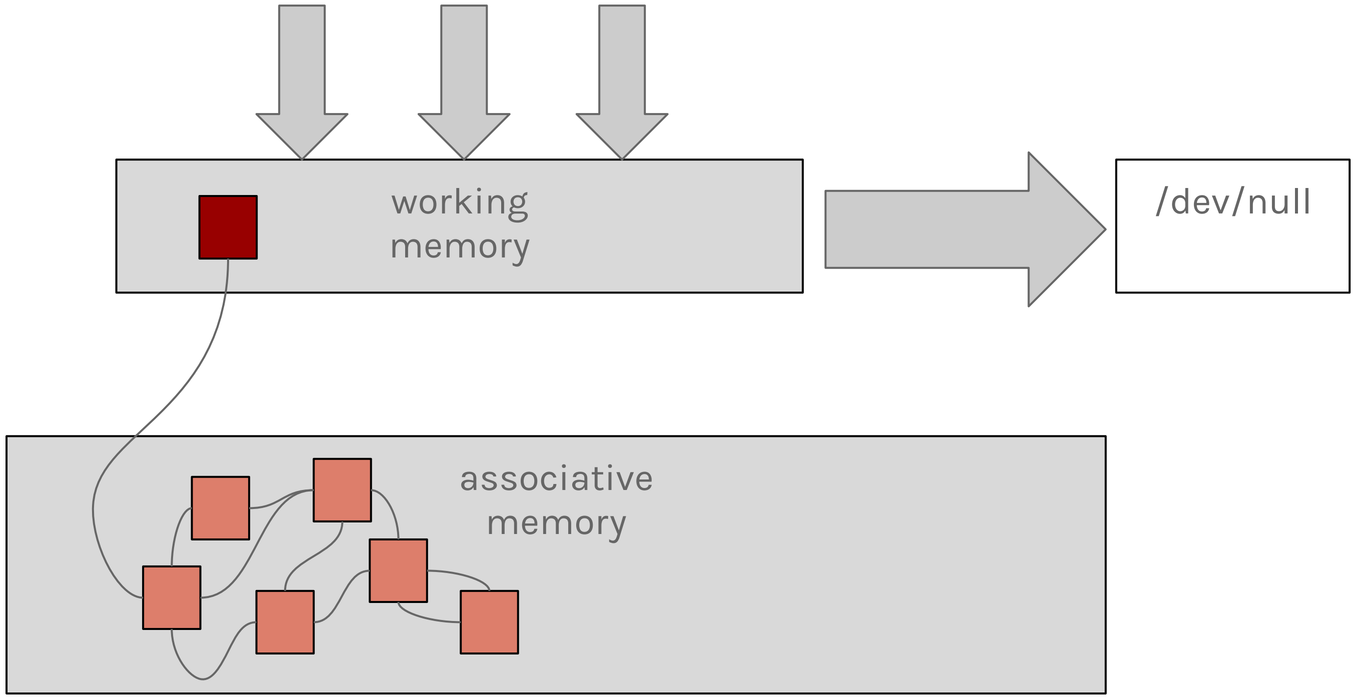
MEMORY STRUCTURE



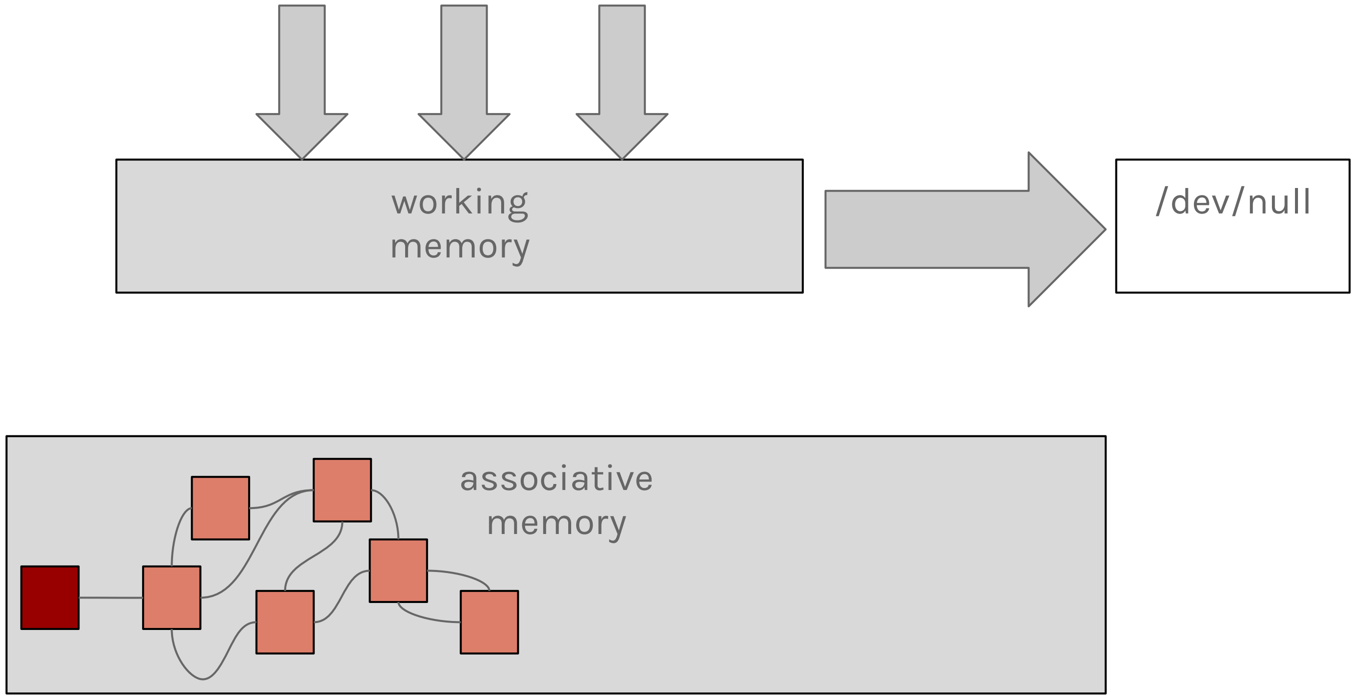
MEMORY STRUCTURE



MEMORY STRUCTURE



MEMORY STRUCTURE



ASSOCIATIONS

75

ASSOCIATIONS

75

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

  687583861

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

68 75 8 386 1

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

68 **75** 8 386 1

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

68 75 **8** 386 1

ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

68 75 8 **386** 1

ASSOCIATIONS IN ACTION

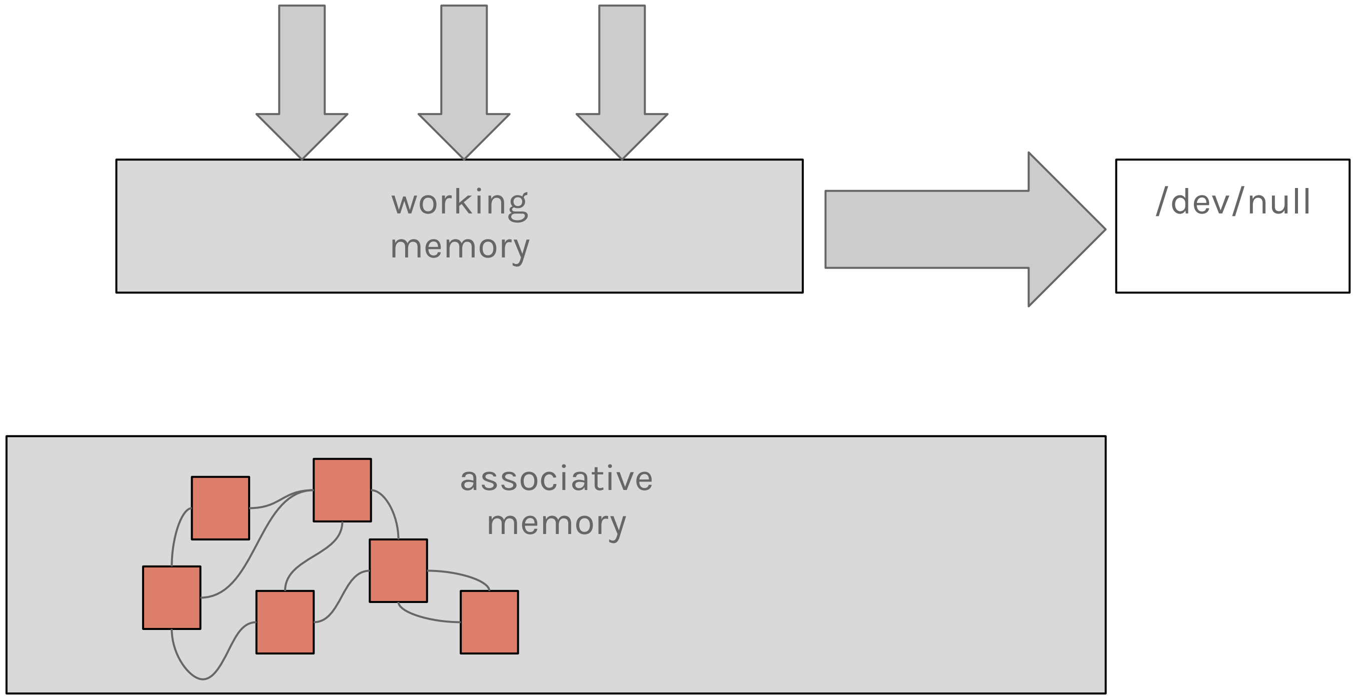
- Memorising a phone number (a tragic story)

68 75 8 386 **1**

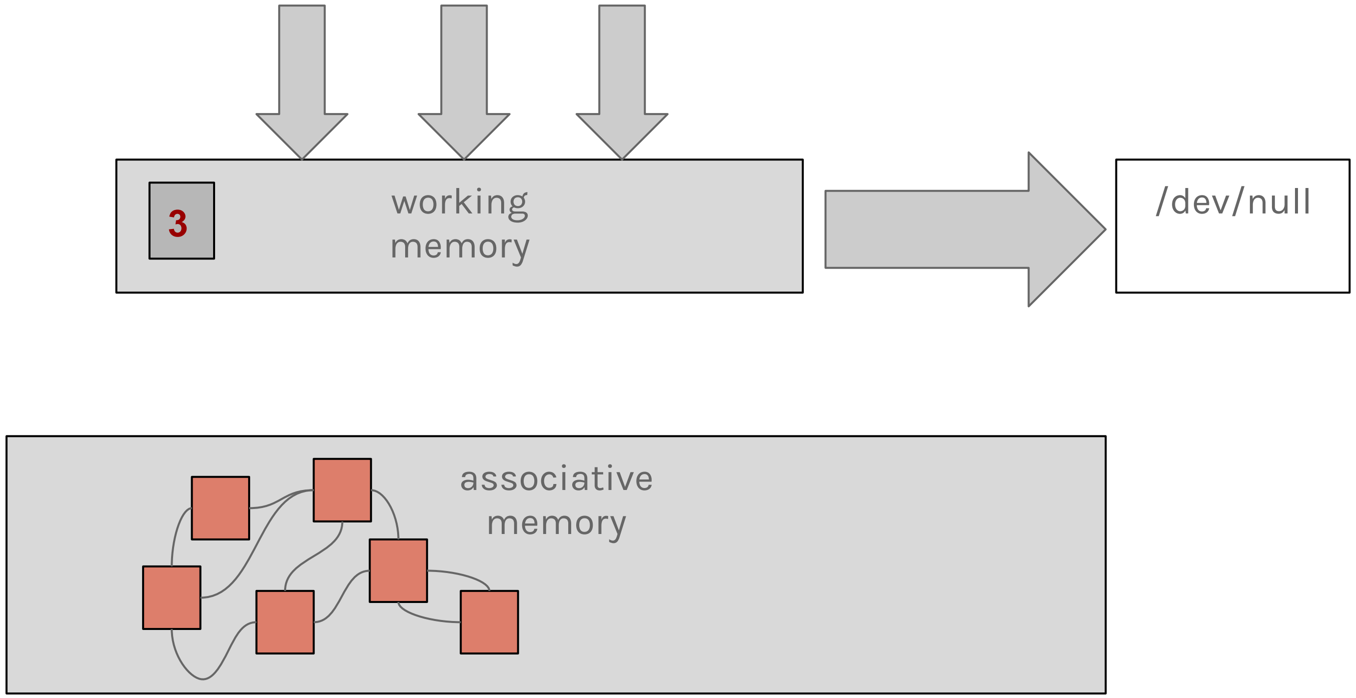
ASSOCIATIONS IN ACTION

- Memorising a phone number (a tragic story)

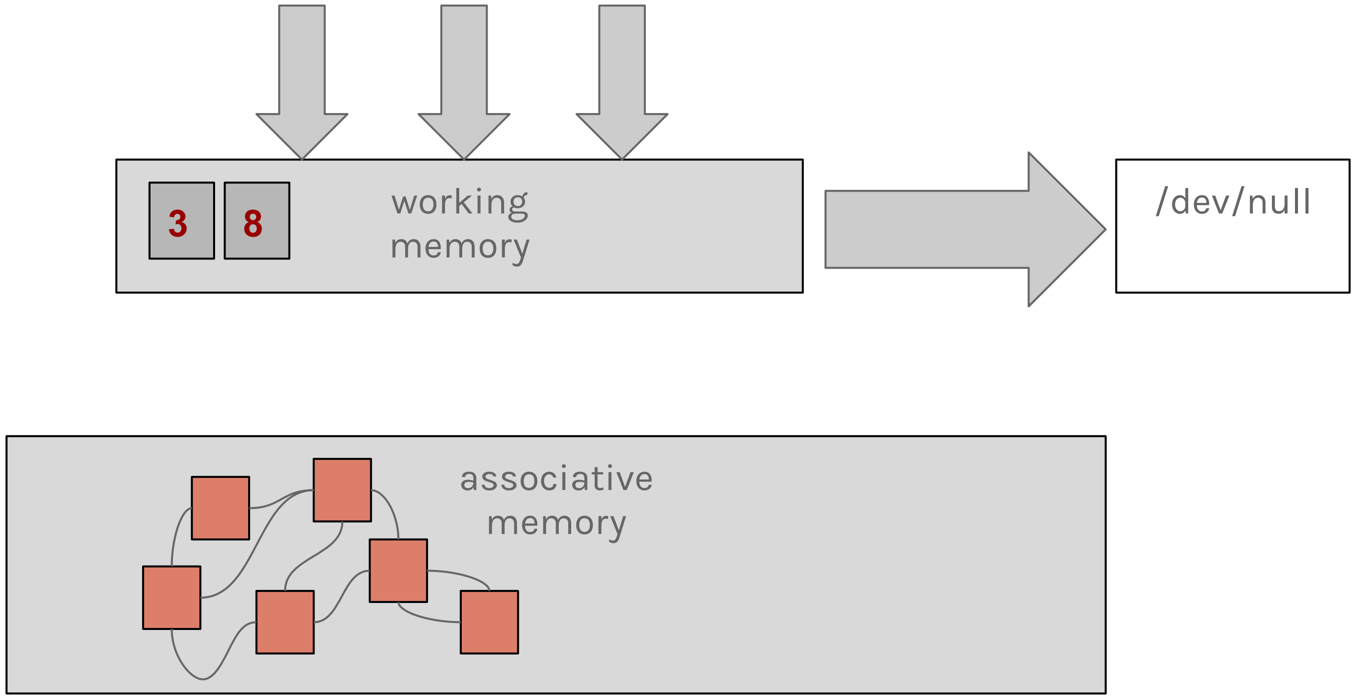
FROM INPUT TO CONCEPT



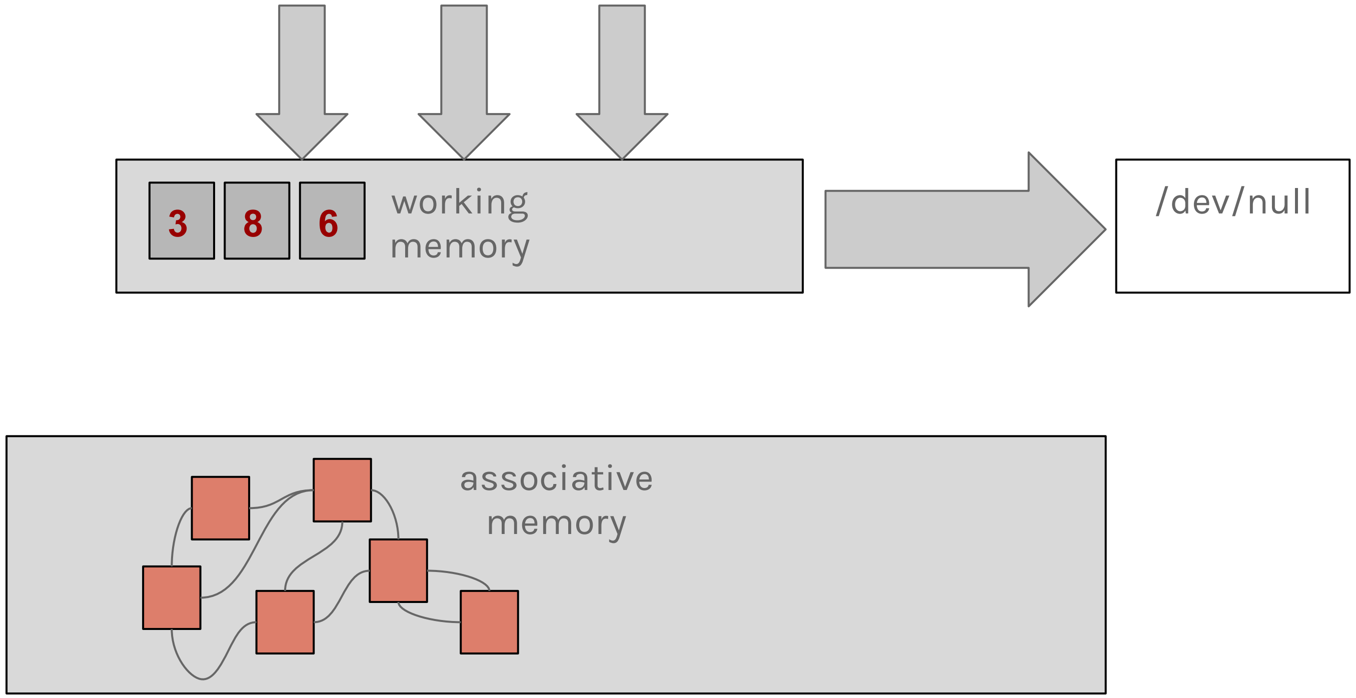
FROM INPUT TO CONCEPT



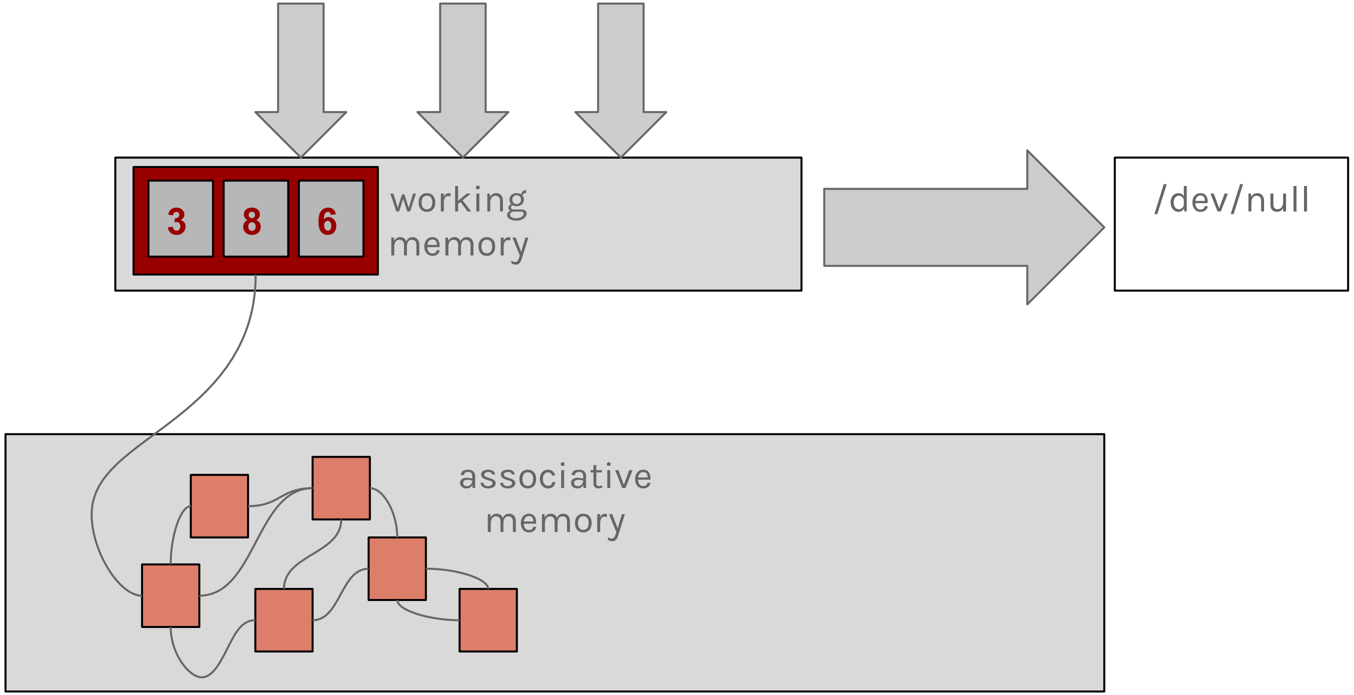
FROM INPUT TO CONCEPT



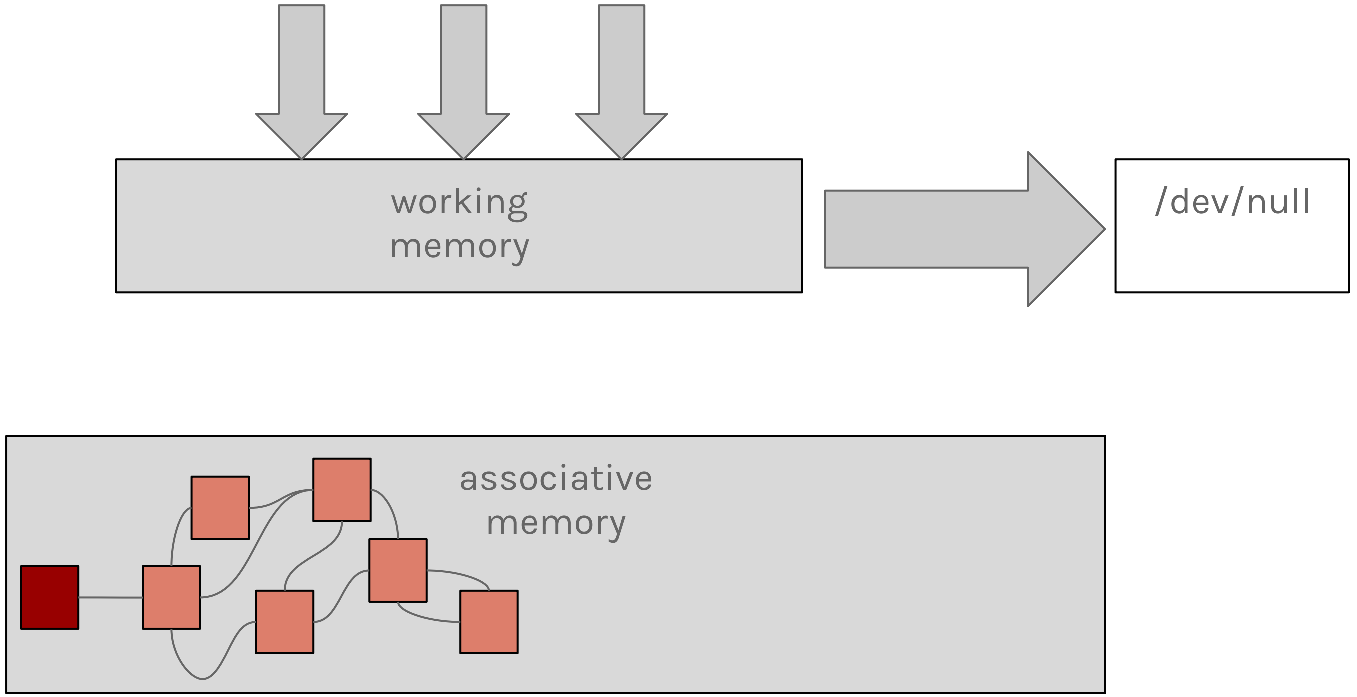
FROM INPUT TO CONCEPT



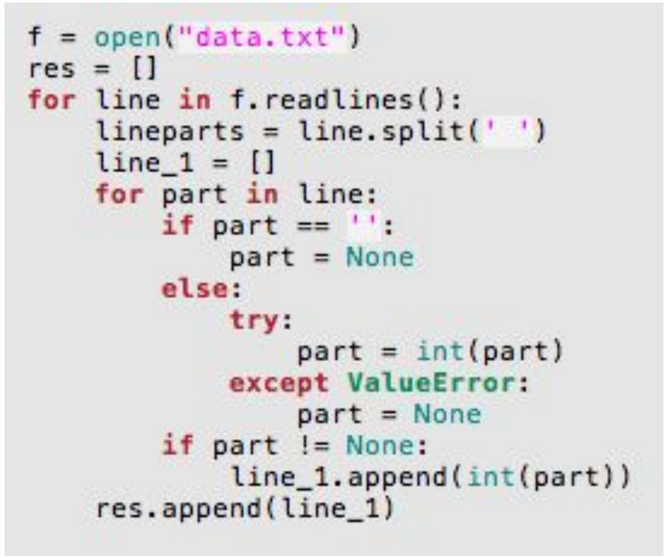
FROM INPUT TO CONCEPT



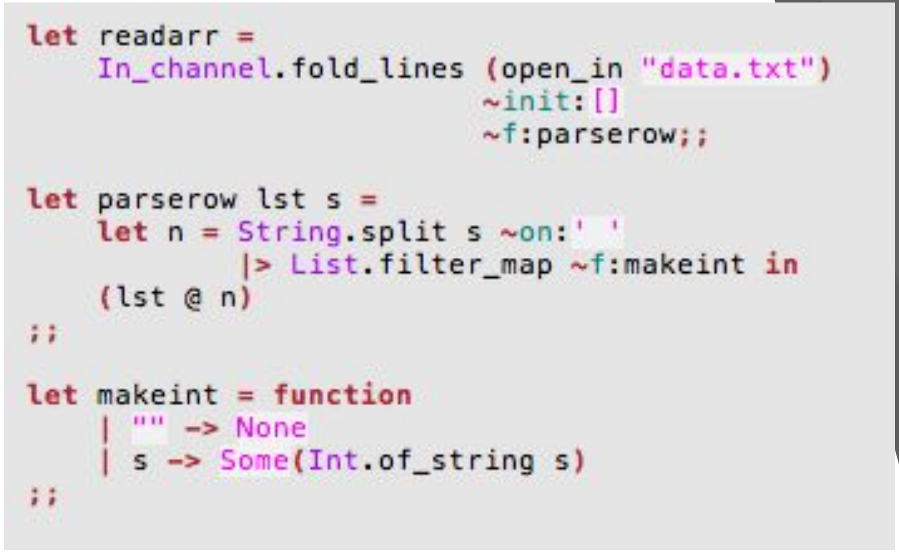
FROM INPUT TO CONCEPT



READING CODE, REVISITED

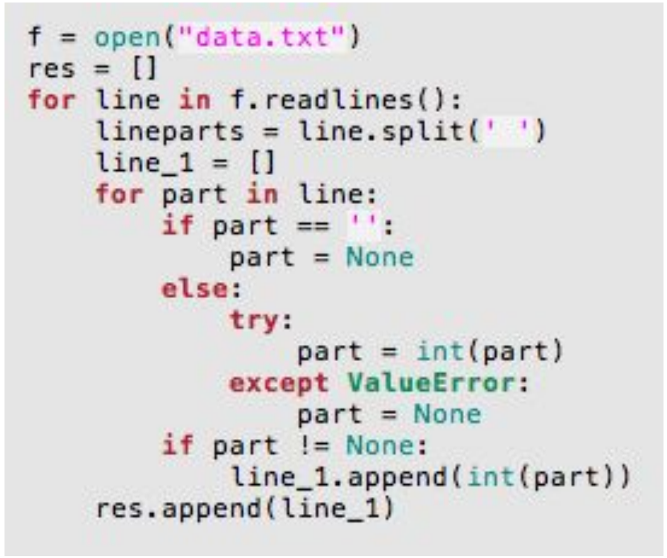


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

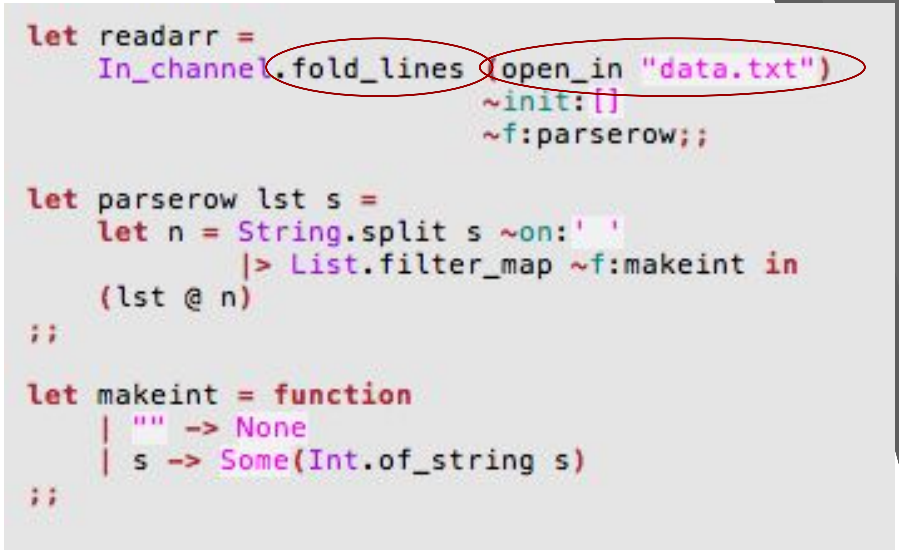


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                          ~init:[]
                          ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
            |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

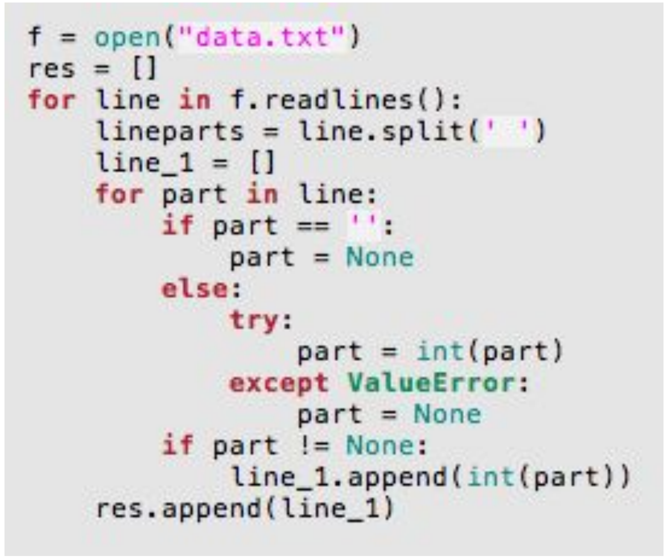


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

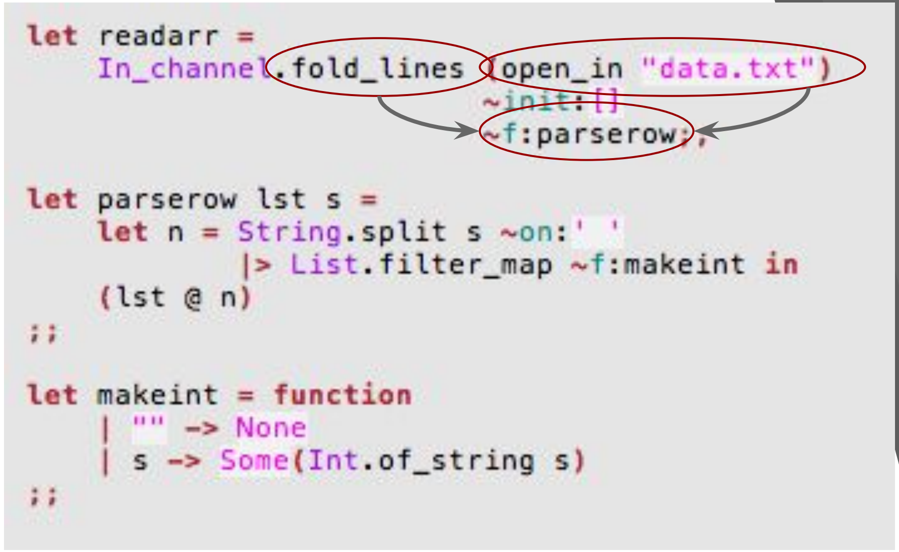


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                          ~init:[]
                          ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
            |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

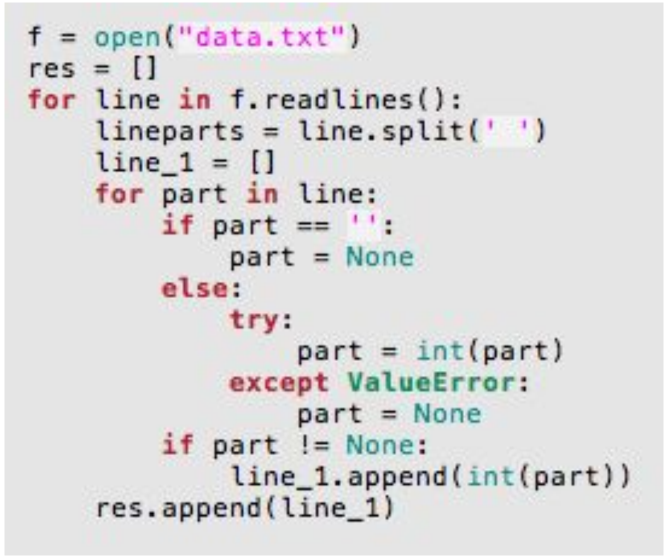


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

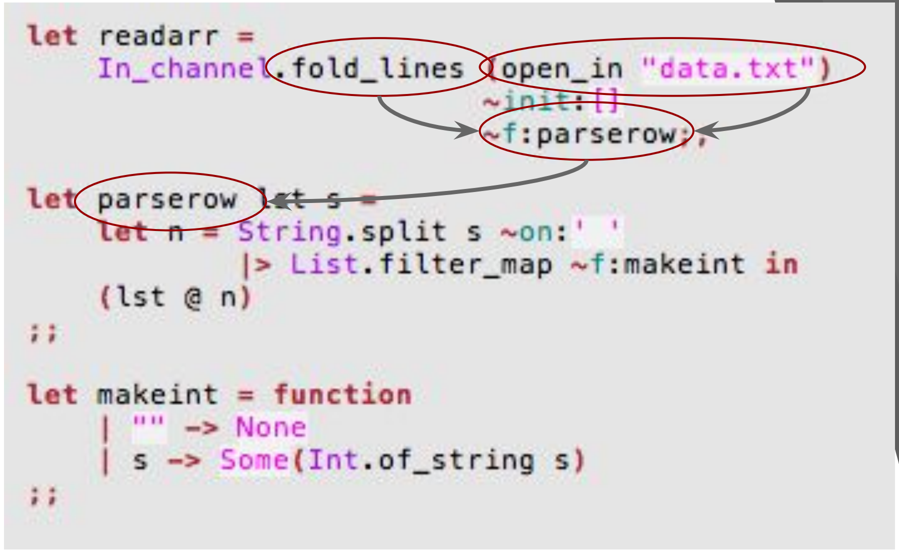


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                          ~init:[]
                          ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
            |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

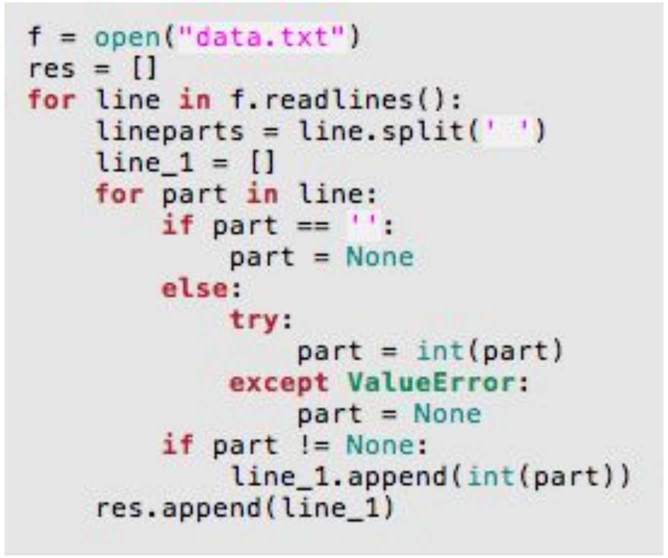


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

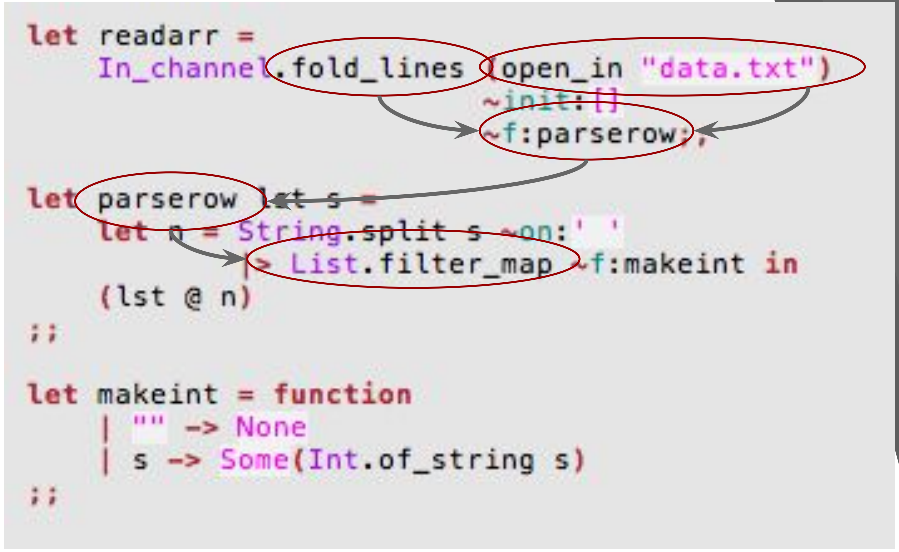


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                                 ~init:[]
                                 ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
              |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

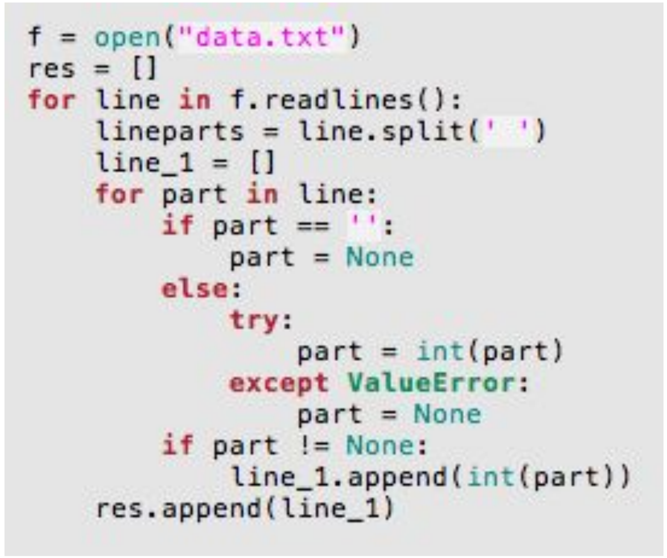


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

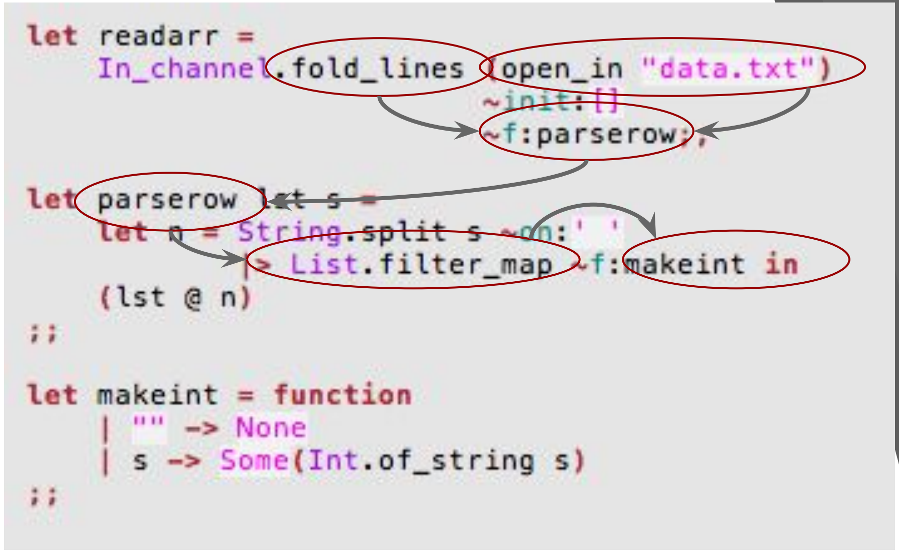


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                          ~init:[]
                          ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
            |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

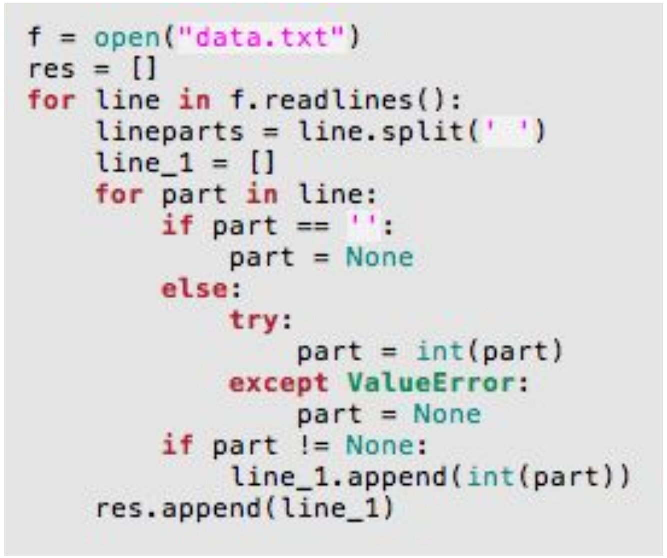


```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```

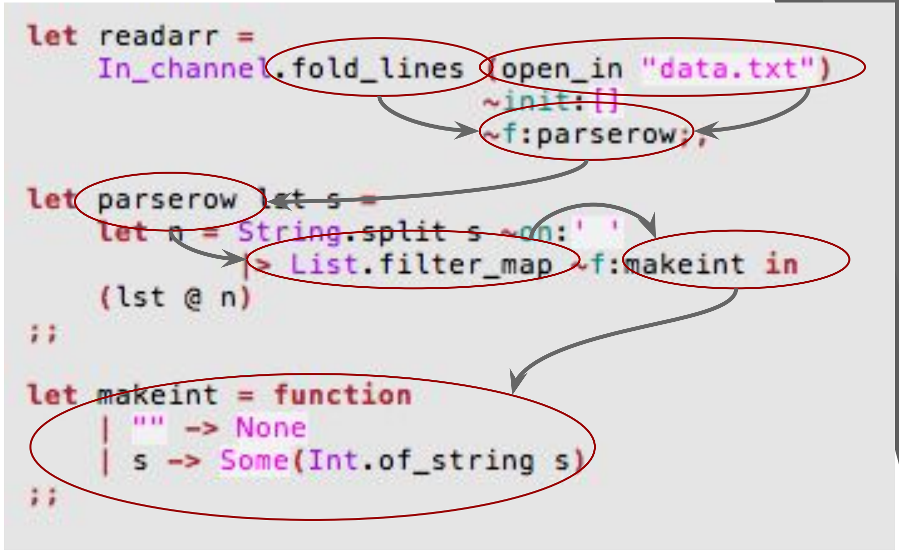


```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                          ~init:[]
                          ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
          |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED



```python
f = open("data.txt")
res = []
for line in f.readlines():
    lineparts = line.split(' ')
    line_1 = []
    for part in line:
        if part == '':
            part = None
        else:
            try:
                part = int(part)
            except ValueError:
                part = None
        if part != None:
            line_1.append(int(part))
    res.append(line_1)
```



```ocaml
let readarr =
    In_channel.fold_lines (open_in "data.txt")
                                              ~init:[]
                                              ~f:parserow;;

let parserow lst s =
    let n = String.split s ~on:' '
      |> List.filter_map ~f:makeint in
    (lst @ n)
;;

let makeint = function
    | "" -> None
    | s -> Some(Int.of_string s)
;;
```

READING CODE, REVISITED

- FP is explicit about concepts
  - Isolated
  - Formulated
  - Easy to associate

READING CODE, REVISITED

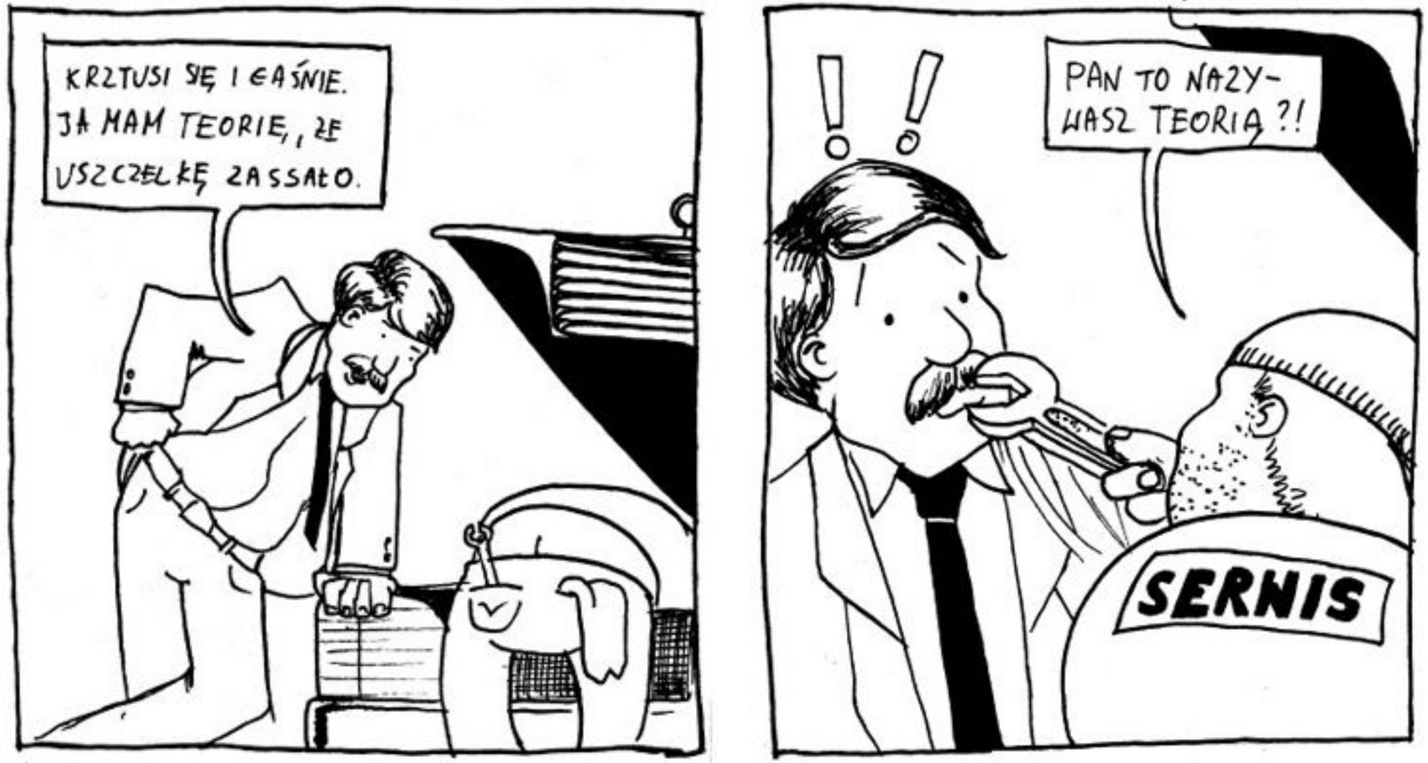
- FP is explicit about concepts
  - Isolated
  - Formulated
  - Easy to associate
- Understanding
  - Thinking = building a concept network

UNDERSTANDING THE CODE

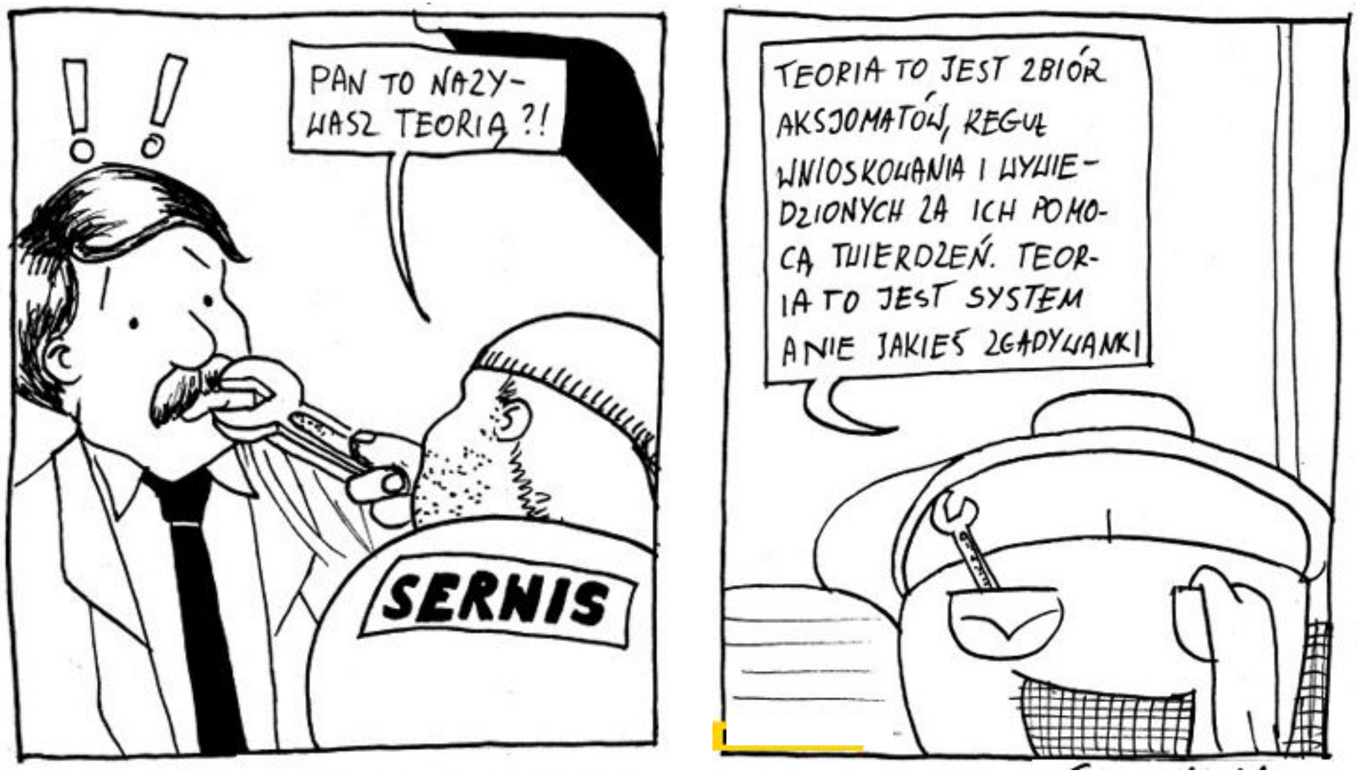
FUNCTIONAL PROGRAMMING

- Better aligned with human brain

“Coughs and stops. My theory is that a seal got stuck.”


KRZTUSI SIĘ I GAŚNIE. JA MAM TEORIĘ, ŻE USZCZELKĘ ZASSAŁO.
! !
PAN TO NAZYWASZ TEORIĄ ?!
SERNIS

"And you call it a theory?!"


PAN TO NAZYWASZ TEORIĄ ?!
SERNIS
TEORIA TO JEST ZBIÓR AKSJOMATÓW, REGUŁ WNIOSKOWANIA I WYWIEDZIONYCH ZA ICH POMOCĄ TWIERDZEŃ. TEORIA TO JEST SYSTEM A NIE JAKIEŚ ZGADYWANKI

“Theory is a collection of axioms, rules of inferences and theorems derived from them. Theory is a system, not some stupid guesswork.”

TEORIA TO JEST ZBIÓR AKSJOMATÓW, REGUŁ WNIOSKOWANIA I WYWIE-DZIONYCH ZA ICH POMOCĄ TWIERDZEŃ. TEORIA TO JEST SYSTEM A NIE JAKIEŚ ZGADYWANKI

PÓŹNIEJ
HIPOTEZY OD TEORII NIE ODRÓŻNIA, ALE SIĘ MĄDRZY.

LATER:
"He thinks he's so smart, while he can't tell theory from hypothesis."

IN SEARCH FOR A PROOF

- Meta-research
- Experiment

IN SEARCH FOR A PROOF

- Meta-research
- Experiment
- Which language is most readable for you?

IN SEARCH FOR A PROOF

- Meta-research
- Experiment
- Which language is most readable for you?
- Idea I: multi-language assignment
  - Reading+coding task in language to choose from
  - Measure average performance and stress level

IN SEARCH FOR A PROOF

- Meta-research
- Experiment
- Which language is most readable for you?
- Idea I: multi-language assignment
  - Reading+coding task in language to choose from
  - Measure average performance and stress level
- Idea II: natural language comparison
  - Natural language instruction
  - Written imperatively or functionally

IN SEARCH FOR A PROOF

Place the steak between two sheets of heavy plastic (resealable freezer bags work well) on a solid, level surface. Firmly pound the beef with the smooth side of a meat mallet to a thickness of 1/8 inch. Combine the olive oil, 2 tablespoons of cilantro, cumin, oregano, 1 pinch of cayenne in a large glass or ceramic bowl; season to taste with salt and pepper. Add the beef and toss to evenly coat. Cover the bowl with plastic wrap, and marinate in the refrigerator for 30 minutes.

IN SEARCH FOR A PROOF

Place the steak between two sheets of heavy plastic (resealable freezer bags work well) on a solid, level surface. Firmly pound the beef with the smooth side of a meat mallet to a thickness of 1/8 inch. Combine the olive oil, 2 tablespoons of cilantro, cumin, oregano, 1 pinch of cayenne in a large glass or ceramic bowl; season to taste with salt and pepper. Add the beef and toss to evenly coat. Cover the bowl with plastic wrap, and marinate in the refrigerator for 30 minutes.

```
(Prepare steak)

Prepare steak:
    (wrap beef)
    (pound beef)
    (mix sauce)

Pound beef:
    beat it with the smooth side of a
    meat mallet to a thickness of
    1/8 inch

Mix sauce:
    (get ingredients)
    mix in a glass bowl
    add salt and pepper
```

PERSONALITY

- Why do people prefer a certain programming style?
  - Accident?
  - Personality traits?
  - Attitudes?
- Requires an extensive research
  - Comparative
  - respondents?

THANK YOU FOR ATTENTION