UNIVERSITEIT VAN AMSTERDAM

Coding Styles & Conventions

Software Construction 2018

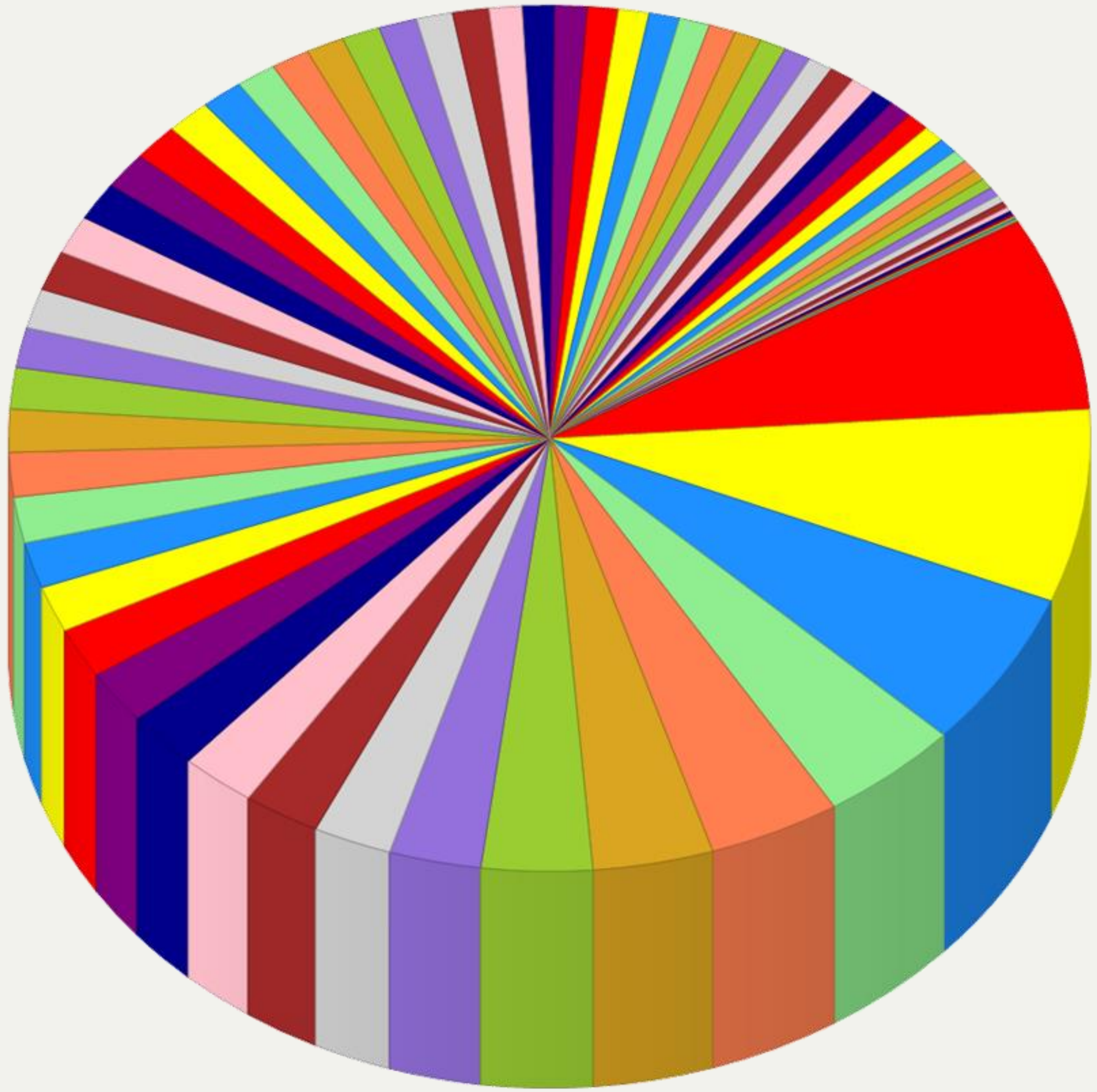
Dr. Vadim Zaytsev aka @grammarware

raincode LABS compiler experts

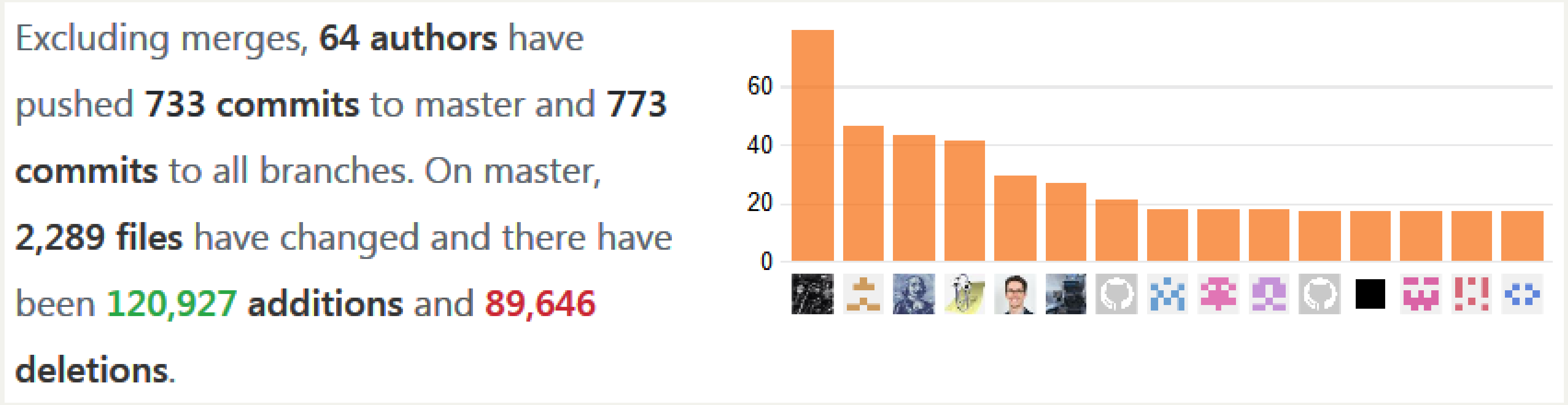
Champions: Nico, Rocco, Laurens

1750 Commits

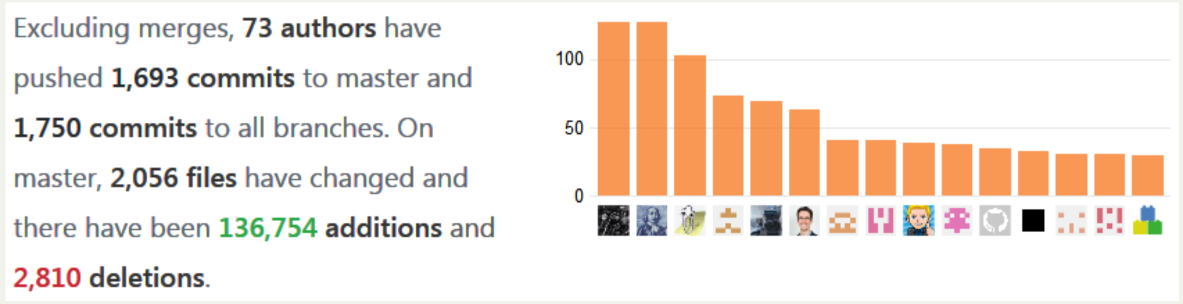
127	Nico Tromp
126	rmathijn
103	lauwdude
73	Mihai Onofrei
69	hasan
63	Peter Takacs
58	Remi van Veen
48	ighmelene
41	GrimGerbil
40	Cornelius Ries
39	Simon Schneider
38	Niels Boerkamp
35	Jordy Bottelier
33	Joanna Roczniak
30	Edwin
30	Elias
30	rashadaoud
29	carlyhill6895
28	bicker
27	Meess
26	Metchu
26	jewelEarthDeveloper
26	olimoli9160
25	Jorick van Rhenen
25	Unknown
24	Dennis Kruidenberg
24	George Vletsas
23	DennisvanderWerf
22	Hector Stenger
21	Tim Nederveen
20	Jaap Koetsier
20	Laurens de Gilde
20	TerryvanWalen
20	porke
19	Jouke
19	Michael de Lang
17	Herczeg
17	Nick
17	Scoudem
16	Dylan Bartels
16	Jovan Maric
16	Sara Oonk
16	Thijs Klaver
14	Deepa Karra
14	Joana Correia Magalhães Sousa
14	bramo
14	sangamm
13	AHerczeg
13	Leó Gunnar vidisson
13	MichielBoswijk
12	Tim
12	evanscharrenburg
12	tdobber
11	Stevan Pavlicic
10	V-Jong



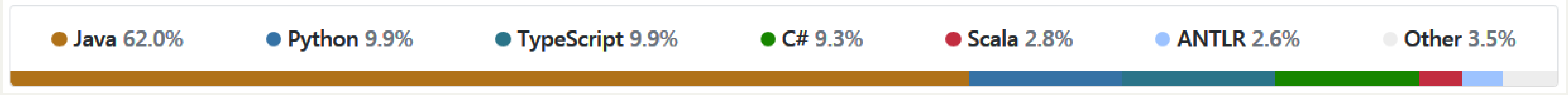
100kLOC; Python, TS, C# on the rise



Excluding merges, **64 authors** have pushed **733 commits** to master and **773 commits** to all branches. On master, **2,289 files** have changed and there have been 120,927 **additions** and 89,646 **deletions**.



Excluding merges, **73 authors** have pushed **1,693 commits** to master and **1,750 commits** to all branches. On master, **2,056 files** have changed and there have been 136,754 **additions** and 2,810 **deletions**.



● **Java** 62.0% ● **Python** 9.9% ● **TypeScript** 9.9% ● **C#** 9.3% ● **Scala** 2.8% ● **ANTLR** 2.6% ● **Other** 3.5%

raincode LABS
compiler experts

Styles and conventions

- There is more than one way to do it
  - cf. TMDOWTDI
- Different styles are not always equivalent in all aspects
  - harmful, incompatible, fit for domain, …
- Styles are recognizable
  - per programmer, community, company, …
- Styles are largely decouplable from language
  - modulo language level
- Styles are largely decouplable from paradigm

Reasons to bother about style

- Coherent writing improves reading
- Useful to recognise structure visually
- “Program shouldn’t have to be convoluted just to get around its data”
- “Program should work all the time” (robustness)
- Easy to spot performance bottlenecks
- Code can be self-documenting

Based on: Kernighan, Plauger, *Programming Style: Examples and Counterexamples*, 1974.

Style drivers

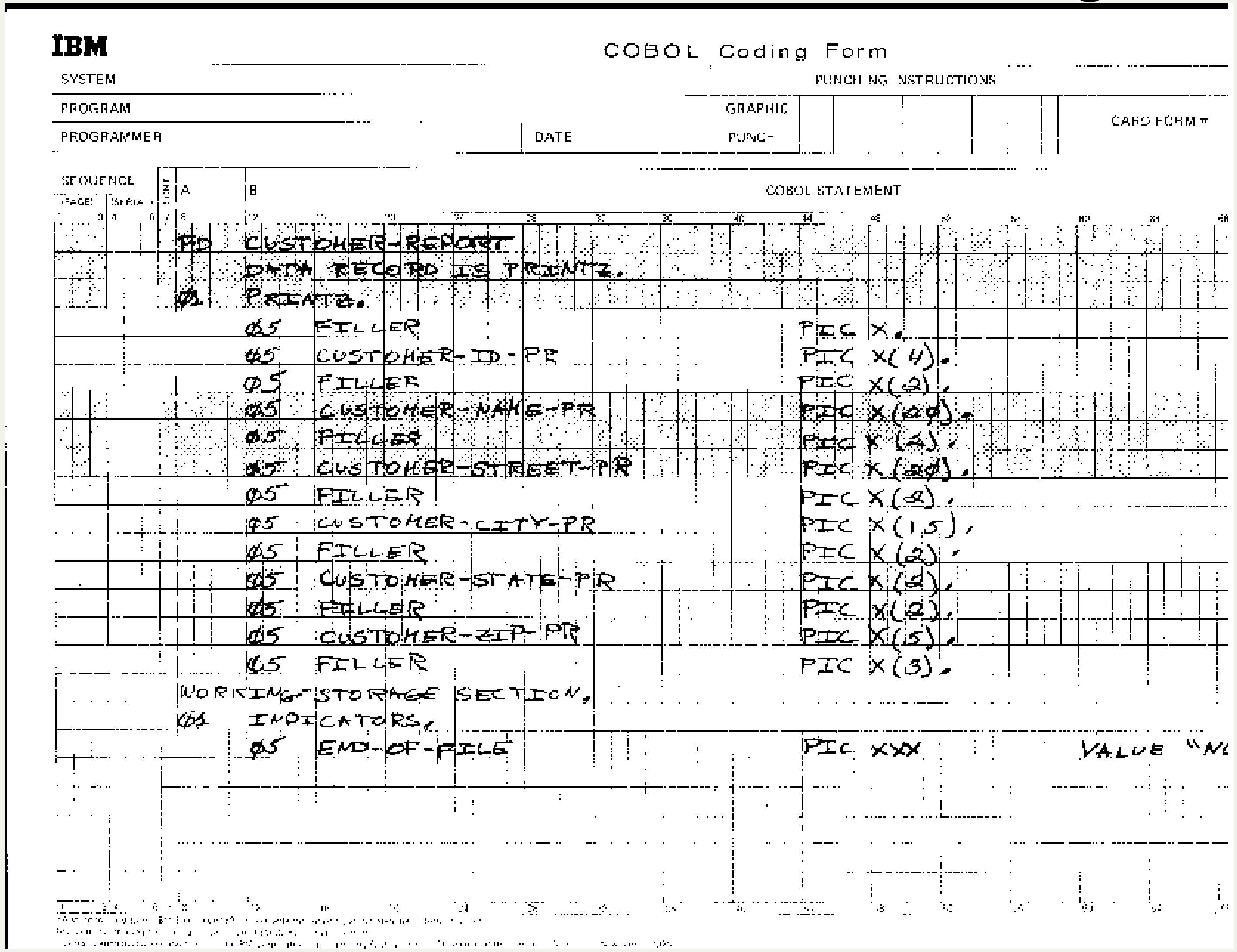
- Readability
- Changeability
- Robustness
- Correctness
- Performance
- Idiomaticness
- Conciseness
- Reuse
- Extensibility
- Usability

Styles & conventions

- Layout
  - indentation, spacing, lining
- Sorting
  - positioning, grouping
- Syntactic preference
  - lowecase vs camelCase, single vs double quotes
- Programming style
  - good and bad practices, avoiding known issues

Goncharenko, Zaytsev, *Language Design and Implementation for the Domain of Coding Conventions*, 2016

Layout conventions: history



IBM COBOL Coding Form

SYSTEM | PUNCHING INSTRUCTIONS
PROGRAM | GRAPHIC | CARD FORM #
PROGRAMMER | DATE | PUNCH

SEQUENCE (PAGE) (SERIAL) | CONT. | A | B | COBOL STATEMENT

```
FD  CUSTOMER-REPORT
    DATA RECORD IS PRINTZ.
Ø1  PRINTZ.
    Ø5  FILLER                                PIC X.
    Ø5  CUSTOMER-ID-PR                        PIC X(4).
    Ø5  FILLER                                PIC X(2).
    Ø5  CUSTOMER-NAME-PR                      PIC X(2Ø).
    Ø5  FILLER                                PIC X(2).
    Ø5  CUSTOMER-STREET-PR                    PIC X(2Ø).
    Ø5  FILLER                                PIC X(2).
    Ø5  CUSTOMER-CITY-PR                      PIC X(15).
    Ø5  FILLER                                PIC X(2).
    Ø5  CUSTOMER-STATE-PR                     PIC X(2).
    Ø5  FILLER                                PIC X(2).
    Ø5  CUSTOMER-ZIP-PR                       PIC X(5).
    Ø5  FILLER                                PIC X(3).
WORKING-STORAGE SECTION.
Ø1  INDICATORS.
    Ø5  END-OF-FILE                           PIC XXX        VALUE "N
```

COBOL Coding Form, http://www.bristol.mass.edu/pgrocer/cis12/form/codeform.html

Layout conventions

`while (x == y) {`
`    something();`
`    somethingelse();`
`}`
	`while (x == y)`
	`{`
	`    something();`
	`    somethingelse();`
	`}`
		`while (x == y)`
		`  {`
		`    something();`
		`    somethingelse();`
		`  }`

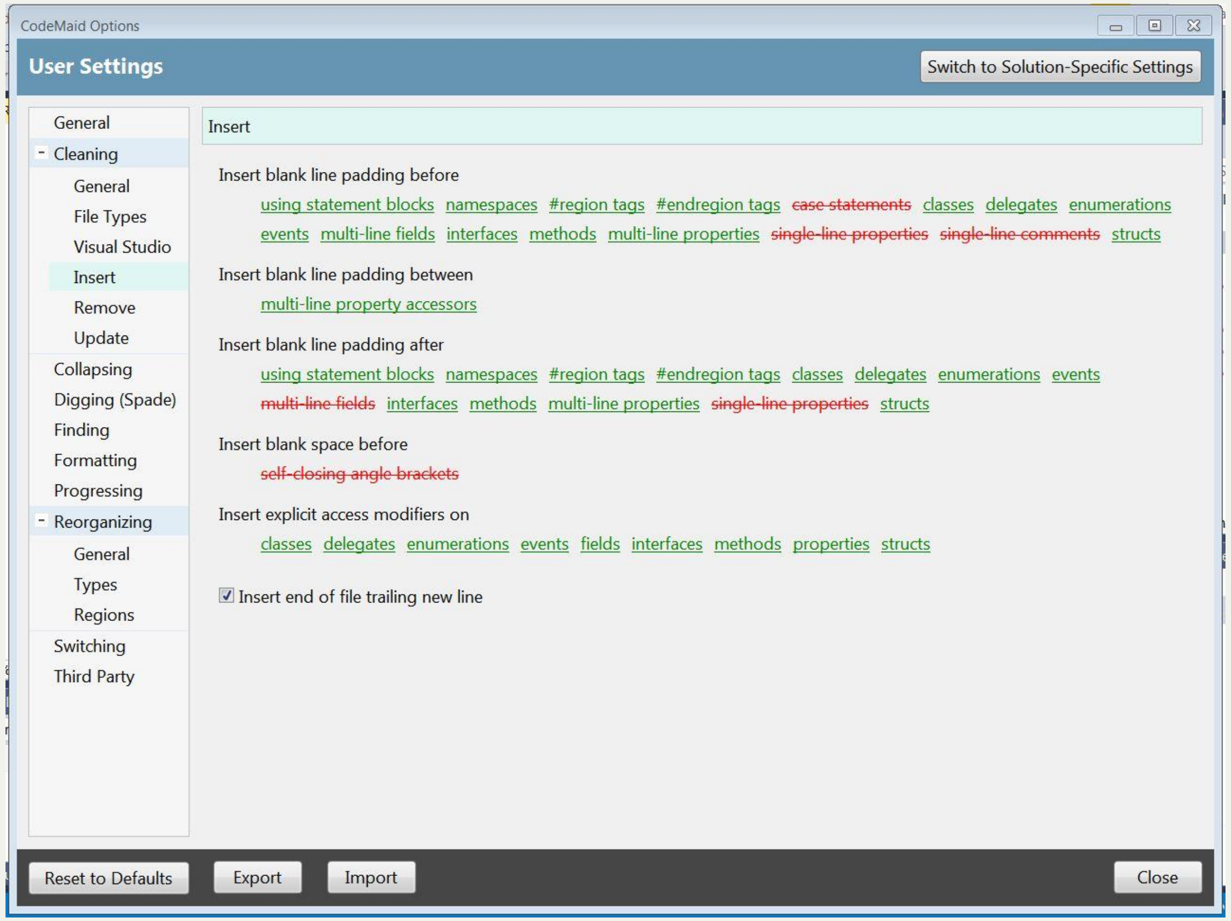
`while (x == y)`
`    {`
`    something();`
`    somethingelse();`
`    }`
	?	`while (x == y) {`
		`    something();`
		`    somethingelse();`
		`    }`

`while (x == y)`
`{   something();`
`    somethingelse();`
`}`
	`while (x == y)`
	`{   something();`
	`    somethingelse(); }`
		`while (x == y) {`
		`    something();`
		`    somethingelse(); }`

https://en.wikipedia.org/wiki/Indentation_style

raincode LABS
compiler experts

Modern IDEs are configurable



Insert

Insert blank line padding before

using statement blocks namespaces #region tags #endregion tags ~~case statements~~ classes delegates enumerations events multi-line fields interfaces methods multi-line properties ~~single-line properties~~ ~~single-line comments~~ structs

Insert blank line padding between

multi-line property accessors

Insert blank line padding after

using statement blocks namespaces #region tags #endregion tags classes delegates enumerations events ~~multi-line fields~~ interfaces methods multi-line properties ~~single-line properties~~ structs

Insert blank space before

~~self-closing angle brackets~~

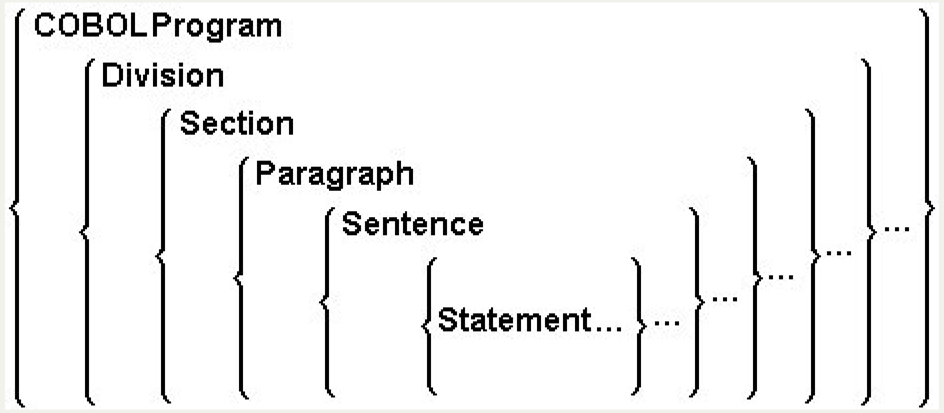
Insert explicit access modifiers on

classes delegates enumerations events fields interfaces methods properties structs

[✓] Insert end of file trailing new line

raincode LABS
compiler experts

Ordering



- Constants / fields / properties / constructors / methods
- One public class per file
- Multiple files per class
- Start file / sheet with a comment

Naming

- BumpyCaps / CamelCase / mixedCase / WikiWord
- ALL_CAPS / ALLCAPS / MACRO_CASE
- snake_case / asdfghjkl
- Space-separated (FORTRAN, ALGOL-58)
- Dash-separated (COBOL, LISP, CSS, Forth) aka Train-Case
- "OF" separated (RETURN_CODE OF HPS_GET_MINMAX)
- Hungarian notation
  - `lpszName, rgfpList, fnStart, chStart, bFlag, cX, wX, nX, lX`

Vincent Ramos, Arch.of.Titus-Inscription, CC-BY-SA 2005.

raincode LABS
compiler experts

How To Write Good

- Avoid clichés like the plague. (They're old hat.)
- Employ the vernacular.
- Foreign words and phrases are not apropos.
- Comparisons are as bad as clichés.
- Don’t be redundant; don’t use more words than necessary; it’s highly superfluous.
- Be more or less specific.
- Understatement is always best.
- Exaggeration is a billion times worse than understatement.
- Analogies in writing are like feathers on a snake.
- Even if a mixed metaphor sings, it should be derailed.

Frank L. Visco, How To Write Good, Writers’ Digest 1986.

How To Write Good

- Avoid alliteration. Always.
- Prepositions are not words to end sentences with.
- Avoid clichés like the plague. (They're old hat.)
- Employ the vernacular.
- Eschew ampersands & abbreviations, etc.
- Parenthetical remarks (however relevant) are unnecessary.
- It is wrong to ever split an infinitive.
- Contractions aren’t necessary.
- Foreign words and phrases are not apropos.
- One should never generalize.
- Eliminate quotations. As Ralph Waldo Emerson once said: "I hate quotations. Tell me what you know."
- Comparisons are as bad as clichés.
- Don't be redundant; don’t use more words than necessary; it's highly superfluous.
- Profanity sucks.
- Be more or less specific.
- Understatement is always best.
- Exaggeration is a billion times worse than understatement.
- One-word sentences? Eliminate.
- Analogies in writing are like feathers on a snake.
- The passive voice is to be avoided.
- Go around the barn at high noon to avoid colloquialisms.
- Even if a mixed metaphor sings, it should be derailed.
- Who needs rhetorical questions?

Frank L. Visco, How To Write Good, Writers’ Digest 1986.

raincode LABS
compiler experts

Syntactic preference

```
List<string> xs = new List<string>();
```

```
for (int i = 0; i < xs.Count; i++)
    if (xs[i] == s)
        return true;
```

```
for (int i = xs.Count; i >=0; i--)
    if (xs[i] == s)
        return true;
```

```
foreach (string x in xs)
    if (x == s)
        return true;
```

```
return xs.Exists(x => x == s);
```

- Use em instead of px
- Prefer rgba() to #hex
- No units for zeros
- Ban words
  - left, top, head, blue, …
- .x and #x must not meet
- Semicolon at the end
- Fewer than 4 selectors

raincode LABS
compiler experts

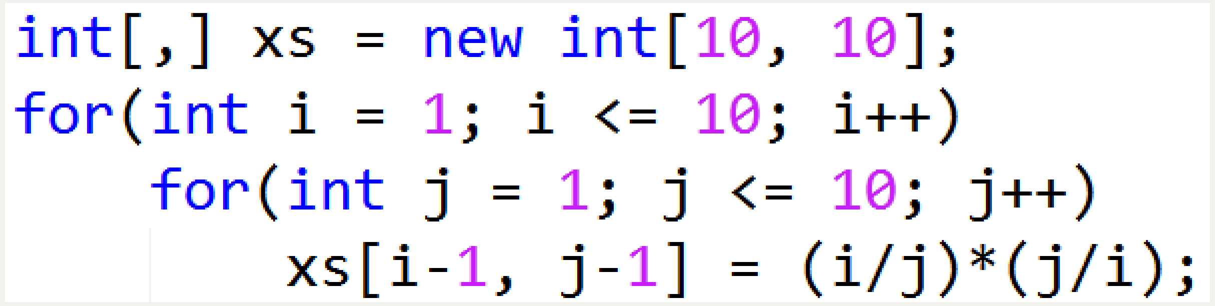
Style preference

- Avoid !important
- Do not use #id selectors
- Disallow universal * selector
- Require properties for display
- Do not use negative indent
- No shorthands but border

- Use em instead of px
- Prefer rgba() to #hex
- No units for zeros
- Ban words
  - left, top, head, blue, …
- .x and #x must not meet
- Semicolon at the end
- Fewer than 4 selectors

Goncharenko, Zaytsev, *Language Design and Implementation for the Domain of Coding Conventions*, 2016

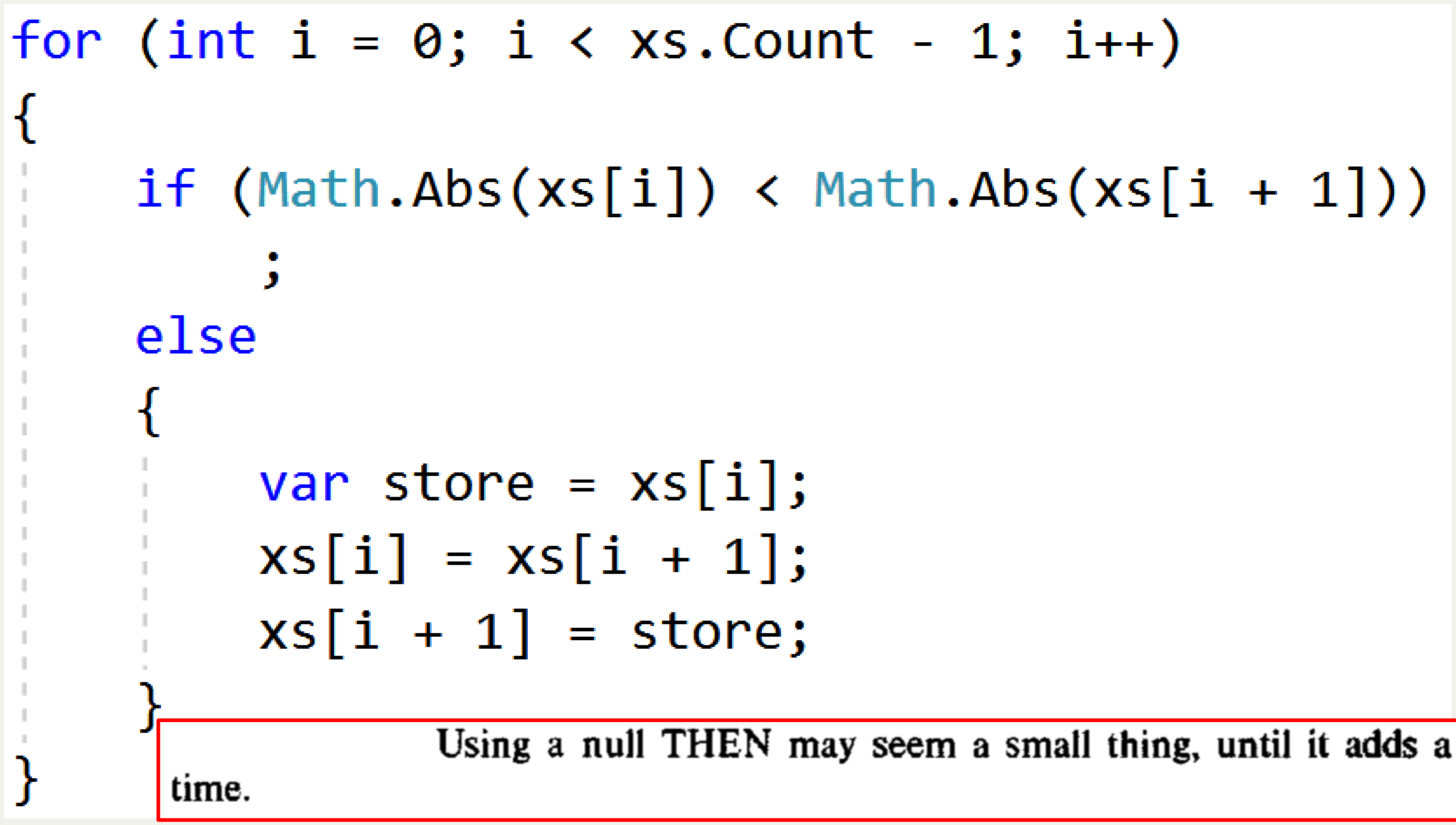
What does this code do?



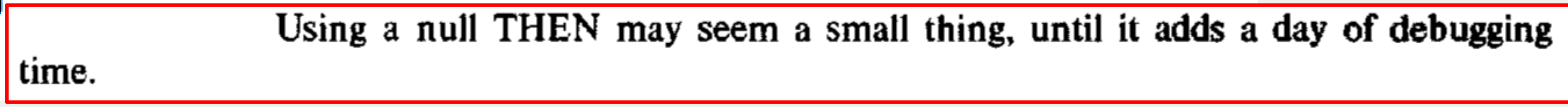
```
int[,] xs = new int[10, 10];
for(int i = 1; i <= 10; i++)
    for(int j = 1; j <= 10; j++)
        xs[i-1, j-1] = (i/j)*(j/i);
```

Kernighan, Plauger, *Programming Style: Examples and Counterexamples*, 1974.

What does this code do?



```
for (int i = 0; i < xs.Count - 1; i++)
{
    if (Math.Abs(xs[i]) < Math.Abs(xs[i + 1]))
        ;
    else
    {
        var store = xs[i];
        xs[i] = xs[i + 1];
        xs[i + 1] = store;
    }
}
```



Using a null THEN may seem a small thing, until it adds a day of debugging time.

Kernighan, Plauger, *Programming Style: Examples and Counterexamples*, 1974.

raincode LABS
compiler experts

Basic styles

- Monolith
- Cookbook
- Pipeline

Monolith

```
word_freqs = []
with open('../stop_words.txt') as f:
    stop_words = f.read().split(',')
stop_words.extend(list(string.ascii_lowercase))

for line in open(sys.argv[1]):
    start_char = None
    i = 0
    for c in line:
        if start_char == None:
            if c.isalnum():
                start_char = i
        else:
            if not c.isalnum():
                found = False
                word = line[start_char:i].lower()
                if word not in stop_words:
                    pair_index = 0
                    for pair in word_freqs:
                        if word == pair[0]:
                            pair[1] += 1
                            found = True
                            found_at = pair_index
                            break
                        pair_index += 1
                    if not found:
                        word_freqs.append([word, 1])
                    elif len(word_freqs) > 1:
                        for n in reversed(range(pair_index)):
                            if word_freqs[pair_index][1] > word_freqs[n][1]:
                                word_freqs[n], word_freqs[pair_index] = word_freqs[pair_index], word_freqs[n]
                                pair_index = n
                start_char = None
        i += 1

for tf in word_freqs[0:25]:
    print tf[0], ' - ', tf[1]
```

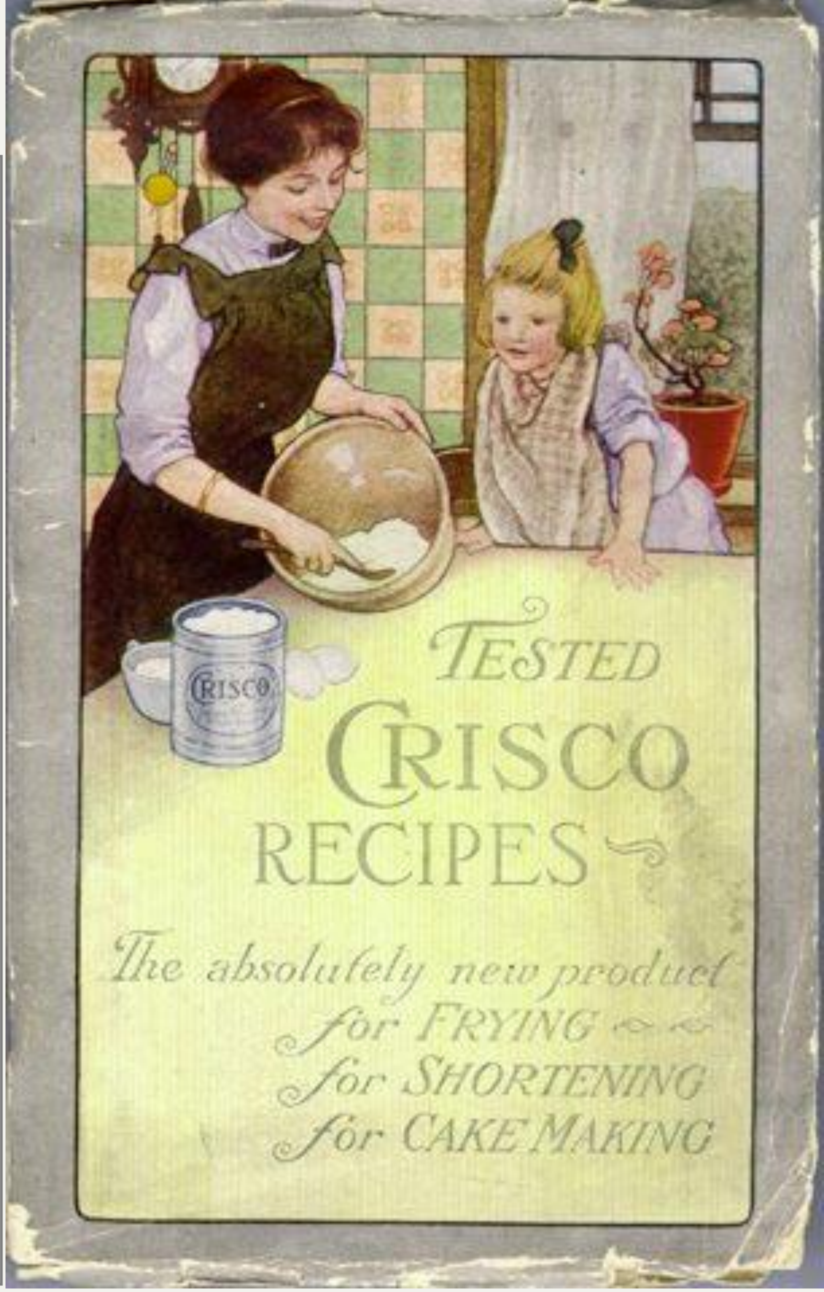
Cristina Videira Lopes,
https://github.com/crista/exercises-in-programming-style/blob/master/03-monolith/tf-03.py
Bjørn Erik Pedersen, Monolith 2 200804, CC-BY-SA, 2008

raincode LABS
compiler experts

Cookbook

```
data = []
words = []
word_freqs = []
def read_file(path_to_file):
    with open(path_to_file) as f:
        data = data + list(f.read())
def filter_chars_and_normalize():
    for i in range(len(data)):
        if not data[i].isalnum():
            data[i] = ' '
        else:
            data[i] = data[i].lower()
def scan():
    data_str = ''.join(data)
    words = words + data_str.split()
def remove_stop_words():
    with open('../stop_words.txt') as f:
        stop_words = f.read().split(',')
    stop_words.extend(list(string.ascii_lowercase))
    indexes = []
    for i in range(len(words)):
        if words[i] in stop_words:
            indexes.append(i)
    for i in reversed(indexes):
        words.pop(i)
def frequencies():
    for w in words:
        keys = [wd[0] for wd in word_freqs]
        if w in keys:
            word_freqs[keys.index(w)][1] += 1
        else:
            word_freqs.append([w, 1])
def sort():
    word_freqs.sort(lambda x, y: cmp(y[1], x[1]))

read_file(sys.argv[1])
filter_chars_and_normalize()
scan()
remove_stop_words()
frequencies()
sort()
for tf in word_freqs[0:25]:
    print tf[0], ' - ', tf[1]
```



Cristina Videira Lopes,
https://github.com/crista/exercises-in-programming-style/blob/master/04-cookbook/tf-04.py
https://commons.wikimedia.org/wiki/File:Crisco_Cookbook_1912.jpg, PD

raincode LABS
compiler experts

Pipeline

```
def read_file(path_to_file):
    with open(path_to_file) as f:
        data = f.read()
    return data

def filter_chars_and_normalize(str_data):
    pattern = re.compile('[\W_]+')
    return pattern.sub(' ', str_data).lower()

def scan(str_data):
    return str_data.split()

def remove_stop_words(word_list):
    with open('../stop_words.txt') as f:
        stop_words = f.read().split(',')
    stop_words.extend(list(string.ascii_lowercase))
    return [w for w in word_list if not w in stop_words]

def frequencies(word_list):
    word_freqs = {}
    for w in word_list:
        if w in word_freqs:
            word_freqs[w] += 1
        else:
            word_freqs[w] = 1
    return word_freqs

def sort(word_freq):
    return sorted(word_freq.iteritems(), key=operator.itemgetter(1), reverse=True)

def print_all(word_freqs):
    if(len(word_freqs) > 0):
        print word_freqs[0][0], ' - ', word_freqs[0][1]
        print_all(word_freqs[1:]);

print_all(sort(frequencies(remove_stop_words(scan(filter_chars_and_normalize(read_file(sys.argv[1]))))))[0:25])
```

Cristina Videira Lopes, https://github.com/crista/exercises-in-programming-style/blob/master/05-pipeline/tf-05.py

Dominicus Johannes Bergsma, Baggertransportleiding drijft onder de Trambrug, CC-BY-SA, 2017

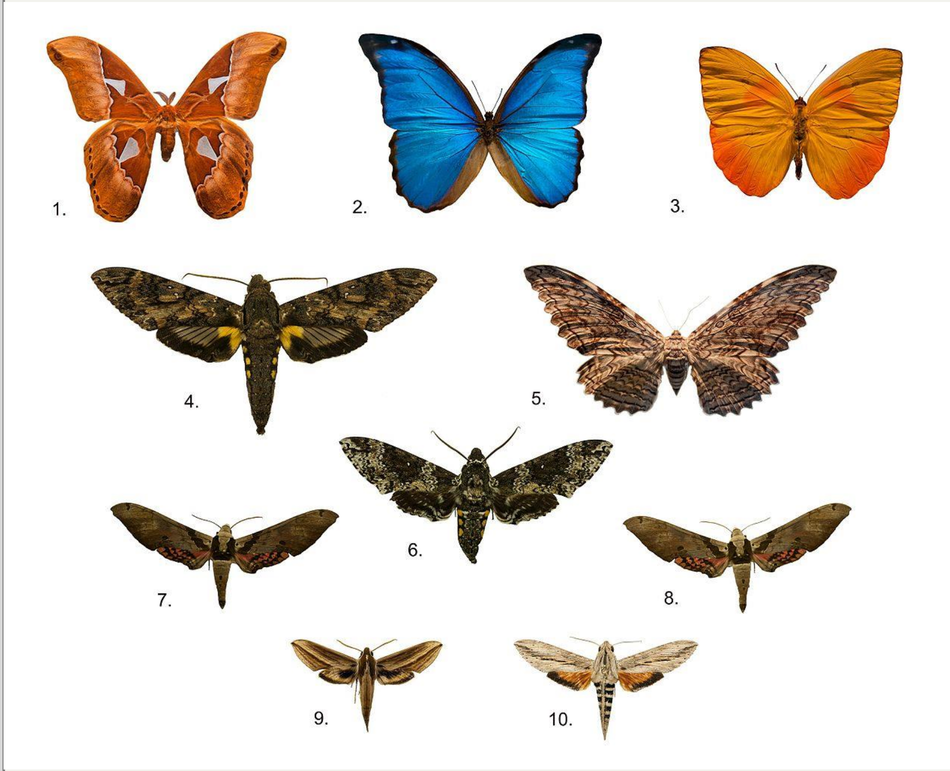
raincode LABS
compiler experts

Objects & object interaction

- Things
- Abstract Things
- Hollywood

Things (Kingdom of Nouns)

```
class TFExercise():
    __metaclass__ = ABCMeta
    def info(self):
        return self.__class__.__name__
class DataStorageManager(TFExercise):
    def __init__(self, path_to_file):
        with open(path_to_file) as f:
            self._data = f.read()
        pattern = re.compile('[\W_]+')
        self._data = pattern.sub(' ', self._data).lower()
    def words(self):
        return self._data.split()
    def info(self):
        return super(DataStorageManager, self).info() + ": My major data structure is a " + self._data.__class__.__name__
class StopWordManager(TFExercise):
    def __init__(self):
        with open('../stop_words.txt') as f:
            self._stop_words = f.read().split(',')
        # add single-letter words
        self._stop_words.extend(list(string.ascii_lowercase))
    def is_stop_word(self, word):
        return word in self._stop_words
    def info(self):
        return super(StopWordManager, self).info() + ": My major data structure is a " + self._stop_words.__class__.__name__
class WordFrequencyManager(TFExercise):
    def __init__(self):
        self._word_freqs = {}
    def increment_count(self, word):
        if word in self._word_freqs:
            self._word_freqs[word] += 1
        else:
            self._word_freqs[word] = 1
    def sorted(self):
        return sorted(self._word_freqs.iteritems(), key=operator.itemgetter(1), reverse=True)
    def info(self):
        return super(WordFrequencyManager, self).info() + ": My major data structure is a " + self._word_freqs.__class__.__name__
class WordFrequencyController(TFExercise):
    def __init__(self, path_to_file):
        self._storage_manager = DataStorageManager(path_to_file)
        self._stop_word_manager = StopWordManager()
        self._word_freq_manager = WordFrequencyManager()
    def run(self):
        for w in self._storage_manager.words():
            if not self._stop_word_manager.is_stop_word(w):
                self._word_freq_manager.increment_count(w)
        word_freqs = self._word_freq_manager.sorted()
        for (w, c) in word_freqs[0:25]:
            print w, ' - ', c
WordFrequencyController(sys.argv[1]).run()
```



Cristina Videira Lopes, https://github.com/crista/exercises-in-programming-style/blob/master/10-things/tf-10.py
The Photographer, [Brazilians butterfly collection](), CC-BY-SA, 2016.

raincode LABS
compiler experts

Abstract things

```
class IDataStorage (object):
    __metaclass__ = abc.ABCMeta
    @abc.abstractmethod
    def words(self):
        pass
class IStopWordFilter (object):
    __metaclass__ = abc.ABCMeta
    @abc.abstractmethod
    def is_stop_word(self, word):
        pass
class IWordFrequencyCounter(object):
    __metaclass__ = abc.ABCMeta
    @abc.abstractmethod
    def increment_count(self, word):
        pass
    @abc.abstractmethod
    def sorted(self):
        pass
class DataStorageManager:
    _data = ''
    def __init__(self, path_to_file):
        with open(path_to_file) as f:
            self._data = f.read()
        pattern = re.compile('[\W_]+')
        self._data = pattern.sub(' ', self._data).lower()
        self._data = ''.join(self._data).split()
    def words(self):
        return self._data
class StopWordManager:
    _stop_words = []
    def __init__(self):
        with open('../stop_words.txt') as f:
            self._stop_words = f.read().split(',')
        self._stop_words.extend(list(string.ascii_lowercase))
    def is_stop_word(self, word):
        return word in self._stop_words
class WordFrequencyManager:
    _word_freqs = {}
    def increment_count(self, word):
        if word in self._word_freqs:
            self._word_freqs[word] += 1
        else:
            self._word_freqs[word] = 1
    def sorted(self):
        return sorted(self._word_freqs.iteritems(), key=operator.itemgetter(1),
reverse=True)
IDataStorage.register(DataStorageManager)
IStopWordFilter.register(StopWordManager)
IWordFrequencyCounter.register(WordFrequencyManager)
class WordFrequencyController:
    def __init__(self, path_to_file):
        self._storage = DataStorageManager(path_to_file)
        self._stop_word_manager = StopWordManager()
        self._word_freq_counter = WordFrequencyManager()
    def run(self):
        for w in self._storage.words():
            if not self._stop_word_manager.is_stop_word(w):
                self._word_freq_counter.increment_count(w)
        word_freqs = self._word_freq_counter.sorted()
        for (w, c) in word_freqs[0:25]:
            print w, ' - ', c
WordFrequencyController(sys.argv[1]).run()
```

Cristina Videira Lopes,
https://github.com/crista/exercises-in-programming-style/blob/master/13-abstract-things/tf-13.py

Hollywood

```
class WordFrequencyFramework:
    _load_event_handlers = []
    _dowork_event_handlers = []
    _end_event_handlers = []
    def register_for_load_event(self, handler):
        self._load_event_handlers.append(handler)
    def register_for_dowork_event(self, handler):
        self._dowork_event_handlers.append(handler)
    def register_for_end_event(self, handler):
        self._end_event_handlers.append(handler)

    def run(self, path_to_file):
        for h in self._load_event_handlers:
            h(path_to_file)
        for h in self._dowork_event_handlers:
            h()
        for h in self._end_event_handlers:
            h()
class DataStorage:
    _data = ''
    _stop_word_filter = None
    _word_event_handlers = []
    def __init__(self, wfapp, stop_word_filter):
        self._stop_word_filter = stop_word_filter
        wfapp.register_for_load_event(self.__load)
        wfapp.register_for_dowork_event(self.__produce_words)
    def __load(self, path_to_file):
        with open(path_to_file) as f:
            self._data = f.read()
        pattern = re.compile('[\W_]+')
        self._data = pattern.sub(' ', self._data).lower()
    def __produce_words(self):
        data_str = ''.join(self._data)
        for w in data_str.split():
            if not self._stop_word_filter.is_stop_word(w):
                for h in self._word_event_handlers:
                    h(w)
    def register_for_word_event(self, handler):
        self._word_event_handlers.append(handler)
class StopWordFilter:
    _stop_words = []
    def __init__(self, wfapp):
        wfapp.register_for_load_event(self.__load)
    def __load(self, ignore):
        with open('../stop_words.txt') as f:
            self._stop_words = f.read().split(',')
        self._stop_words.extend(list(string.ascii_lowercase))
    def is_stop_word(self, word):
        return word in self._stop_words
class WordFrequencyCounter:
    _word_freqs = {}
    def __init__(self, wfapp, data_storage):

data_storage.register_for_word_event(self.__increment_count)
        wfapp.register_for_end_event(self.__print_freqs)
    def __increment_count(self, word):
        if word in self._word_freqs:
            self._word_freqs[word] += 1
        else:
            self._word_freqs[word] = 1
    def __print_freqs(self):
        word_freqs = sorted(self._word_freqs.iteritems(),
key=operator.itemgetter(1), reverse=True)
        for (w, c) in word_freqs[0:25]:
            print w, ' - ', c
wfapp = WordFrequencyFramework()
stop_word_filter = StopWordFilter(wfapp)
data_storage = DataStorage(wfapp, stop_word_filter)
word_freq_counter = WordFrequencyCounter(wfapp, data_storage)
wfapp.run(sys.argv[1])
```

Cristina Videira Lopes,
https://github.com/crista/exercises-in-programming-style/blob/master/14-hollywood/tf-14.py
Sten Rüdrich, Hollywood sign, CC-BY-SA, 2006.

Dealing with adversity

- Constructivism
- Tantrum
- Passive aggressive

Constructivism

```
def extract_words(path_to_file):
    if type(path_to_file) is not str or not path_to_file:
        return []
    try:
        with open(path_to_file) as f:
            str_data = f.read()
    except IOError as e:
        print "I/O error({0}) when opening {1}: {2}".format(e.errno, path_to_file, e.strerror)
        return []
    pattern = re.compile('[\W_]+')
    word_list = pattern.sub(' ', str_data).lower().split()
    return word_list
def remove_stop_words(word_list):
    if type(word_list) is not list:
        return []
    try:
        with open('../stop_words.txt') as f:
            stop_words = f.read().split(',')
    except IOError as e:
        print "I/O error({0}) when opening ../stops_words.txt: {1}".format(e.errno, e.strerror)
        return word_list
    stop_words.extend(list(string.ascii_lowercase))
    return [w for w in word_list if not w in stop_words]
def frequencies(word_list):
    if type(word_list) is not list or word_list == []:
        return {}
    word_freqs = {}
    for w in word_list:
        if w in word_freqs:
            word_freqs[w] += 1
        else:
            word_freqs[w] = 1
    return word_freqs
def sort(word_freq):
    if type(word_freq) is not dict or word_freq == {}:
        return []
    return sorted(word_freq.iteritems(), key=operator.itemgetter(1), reverse=True)
filename = sys.argv[1] if len(sys.argv) > 1 else "../input.txt"
word_freqs = sort(frequencies(remove_stop_words(extract_words(filename))))
for tf in word_freqs[0:25]:
    print tf[0], ' - ', tf[1]
```

Cristina Videira Lopes, https://github.com/crista/exercises-in-programming-style/blob/master/20-constructivist/tf-20.py
Artem Svetlov, -constructivism -sovarch -Architecture –Moscow, CC-BY, 2014.

Tantrum

```
def extract_words(path_to_file):
    assert(type(path_to_file) is str), "I need a
string!"
    assert(path_to_file), "I need a non-empty string!"
    try:
        with open(path_to_file) as f:
            str_data = f.read()
    except IOError as e:
        print "I/O error({0}) when opening {1}: {2}! I
quit!".format(e.errno, path_to_file, e.strerror)
        raise e
    pattern = re.compile('[\W_]+')
    word_list = pattern.sub(' ',
str_data).lower().split()
    return word_list
def remove_stop_words(word_list):
    assert(type(word_list) is list), "I need a list!"
    try:
        with open('../stop_words.txt') as f:
            stop_words = f.read().split(',')
    except IOError as e:
        print "I/O error({0}) when opening
../stops_words.txt: {1}! I quit!".format(e.errno,
e.strerror)
        raise e
    stop_words.extend(list(string.ascii_lowercase))
    return [w for w in word_list if not w in
stop_words]
def frequencies(word_list):
    assert(type(word_list) is list), "I need a list!"
    assert(word_list <> []), "I need a non-empty list!"
    word_freqs = {}
    for w in word_list:
        if w in word_freqs:
            word_freqs[w] += 1
        else:
            word_freqs[w] = 1
    return word_freqs
def sort(word_freq):
    assert(type(word_freq) is dict), "I need a
dictionary!"
    assert(word_freq <> {}), "I need a non-empty
dictionary!"
    try:
        return sorted(word_freq.iteritems(),
key=operator.itemgetter(1), reverse=True)
    except Exception as e:
        print "Sorted threw {0}: {1}".format(e)
        raise e
try:
    assert(len(sys.argv) > 1), "You idiot! I need an
input file!"
    word_freqs =
sort(frequencies(remove_stop_words(extract_words(sys.ar
gv[1]))))

    assert(type(word_freqs) is list), "OMG! This is not
a list!"
    assert(len(word_freqs) > 25), "SRSLY? Less than 25
words!"
    for (w, c) in word_freqs[0:25]:
        print w, ' - ', c
except Exception as e:
    print "Something wrong: {0}".format(e)
    traceback.print_exc()
```

Cristina Videira Lopes, https://github.com/crista/exercises-in-programming-style/blob/master/21-tantrum/tf-21.py

Meadows, O. A. Lawson, *The Mother's Present*, 1847, PD

raincode LABS
compiler experts

Passive aggressive

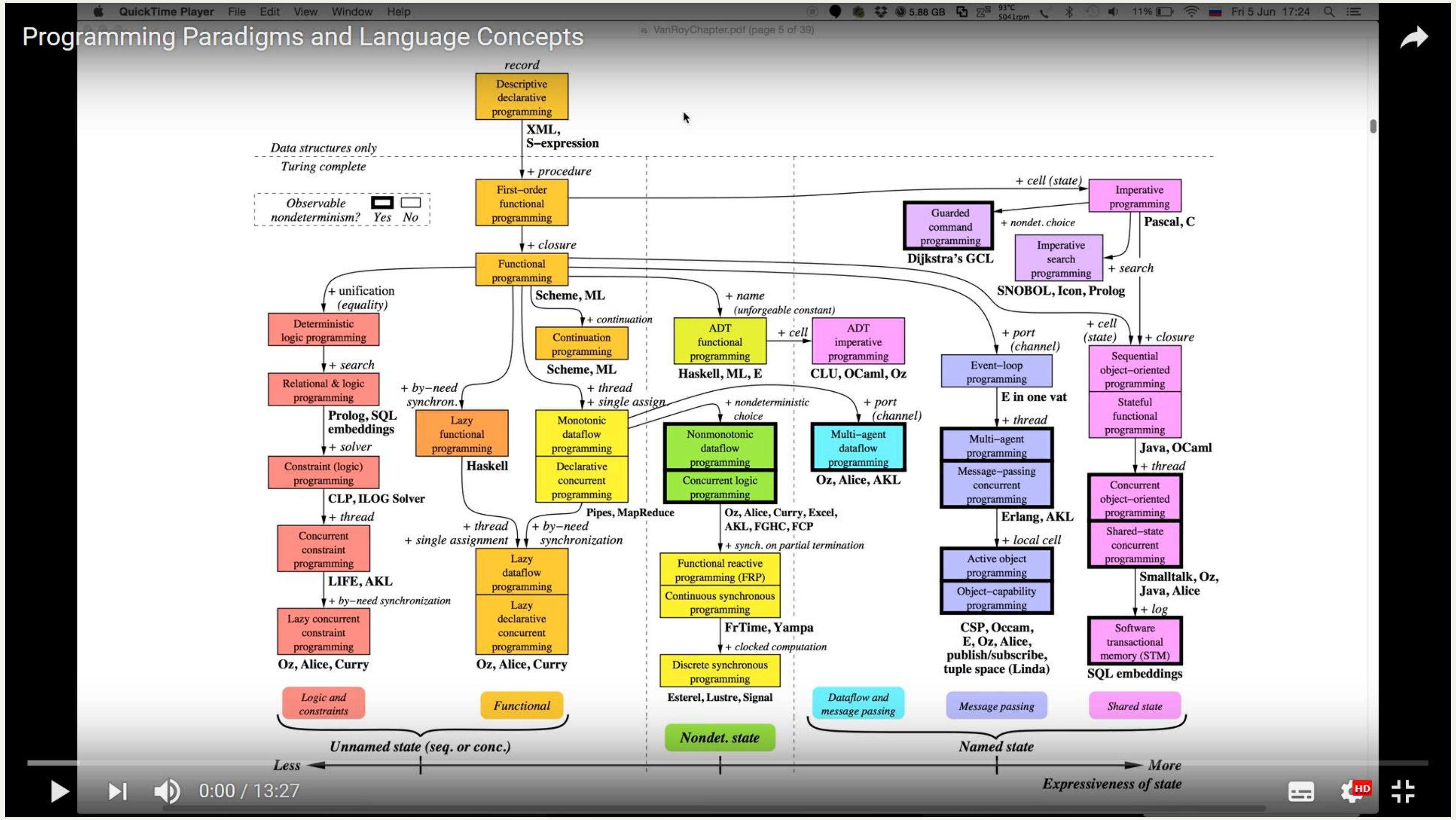
```
def extract_words(path_to_file):
    assert(type(path_to_file) is str), "I need a string! I
quit!"
    assert(path_to_file), "I need a non-empty string! I
quit!"
    with open(path_to_file) as f:
        data = f.read()
    pattern = re.compile('[\W_]+')
    word_list = pattern.sub(' ', data).lower().split()
    return word_list
def remove_stop_words(word_list):
    assert(type(word_list) is list), "I need a list! I
quit!"
    with open('../stop_words.txt') as f:
        stop_words = f.read().split(',')
    # add single-letter words
    stop_words.extend(list(string.ascii_lowercase))
    return [w for w in word_list if not w in stop_words]
def frequencies(word_list):
    assert(type(word_list) is list), "I need a list! I
quit!"
    assert(word_list <> []), "I need a non-empty list! I
quit!"
    word_freqs = {}
    for w in word_list:
        if w in word_freqs:
            word_freqs[w] += 1
        else:
            word_freqs[w] = 1
    return word_freqs
def sort(word_freqs):
    assert(type(word_freqs) is dict), "I need a dictionary!
I quit!"
    assert(word_freqs <> {}), "I need a non-empty
dictionary! I quit!"
    return sorted(word_freqs.iteritems(),
key=operator.itemgetter(1), reverse=True)
try:
    assert(len(sys.argv) > 1), "You idiot! I need an input
file! I quit!"
    word_freqs =
sort(frequencies(remove_stop_words(extract_words(sys.argv[1]
))))

    assert(len(word_freqs) > 25), "OMG! Less than 25 words!
I QUIT!"
    for tf in word_freqs[0:25]:
        print tf[0], ' - ', tf[1]
except Exception as e:
        print "Something wrong: {0}".format(e)
```

Cristina Videira Lopes,
https://github.com/crista/exercises-in-programming-style/blob/master/05-pipeline/tf-05.py

raincode LABS compiler experts

Programming paradigms


Programming Paradigms and Language Concepts
QuickTime Player File Edit View Window Help
VanRoyChapter.pdf (page 5 of 39)
record
Descriptive declarative programming
XML, S-expression
Data structures only
Turing complete
+ procedure
First-order functional programming
Observable nondeterminism? Yes No
+ closure
Functional programming
+ cell (state)
Imperative programming
Pascal, C
Guarded command programming
+ nondet. choice
Dijkstra's GCL
Imperative search programming
+ search
SNOBOL, Icon, Prolog
+ unification (equality)
Deterministic logic programming
+ search
Relational & logic programming
Prolog, SQL embeddings
+ solver
Constraint (logic) programming
CLP, ILOG Solver
+ thread
Concurrent constraint programming
LIFE, AKL
+ by-need synchronization
Lazy concurrent constraint programming
Oz, Alice, Curry
Scheme, ML
+ continuation
Continuation programming
Scheme, ML
+ by-need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Pipes, MapReduce
+ thread
+ single assignment
+ by-need synchronization
Lazy dataflow programming
Lazy declarative concurrent programming
Oz, Alice, Curry
+ name (unforgeable constant)
ADT functional programming
Haskell, ML, E
+ cell
ADT imperative programming
CLU, OCaml, Oz
+ nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
Oz, Alice, Curry, Excel, AKL, FGHC, FCP
+ synch. on partial termination
Functional reactive programming (FRP)
Continuous synchronous programming
FrTime, Yampa
+ clocked computation
Discrete synchronous programming
Esterel, Lustre, Signal
+ port (channel)
Multi-agent dataflow programming
Oz, Alice, AKL
+ port (channel)
Event-loop programming
E in one vat
+ thread
Multi-agent programming
Message-passing concurrent programming
Erlang, AKL
+ local cell
Active object programming
Object-capability programming
CSP, Occam, E, Oz, Alice, publish/subscribe, tuple space (Linda)
+ cell (state)
+ closure
Sequential object-oriented programming
Stateful functional programming
Java, OCaml
+ thread
Concurrent object-oriented programming
Shared-state concurrent programming
Smalltalk, Oz, Java, Alice
+ log
Software transactional memory (STM)
SQL embeddings
Logic and constraints
Functional
Nondet. state
Dataflow and message passing
Message passing
Shared state
Unnamed state (seq. or conc.)
Named state
Less
More
Expressiveness of state
0:00 / 13:27

Programming Paradigms and Language Concepts, PSE @ UvA, 2015, https://youtu.be/lqmMqtgWpms

Conclusion

- Conventions and styles are not tied to a language
- 9 different styles for the same program
- Design is about decisions by you!
- Different trade-offs, benefits, drawbacks
- For QL/QLS:
  - AST hierarchy, type checking, expression evaluation, rendering, event handling: ask yourselves which style is the best

raincode LABS
compiler experts