サイボウズのインフラQAを1μm支えてそうな技術

～研修の一環で行うチーム体験の成果発表ドラゴン～

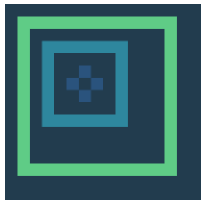
Show（@ajfAfg）

サイボウズのインフラQA を1μm支えてそうな技術

～研修の一環で行うチーム体験の成果

Show（@ajfAfg）

自己紹介



- Show（真名は佐々木勝一）
- サイボウズ24新卒の生産性向上エンジニア（明日から）
- 好きなもの
  - プログラミング言語
  - 音楽（ダンスミュージック、ゲーム音楽（、ジャズ））
  - お酒（特に、ビール、日本酒）
  - 体を動かすこと

サイボウズとは

- 「**チームワークあふれる社会を創る**」ことを目的として
  多数の**グループウェア**製品を開発・販売しているSaaS企業

スムーズな情報共有による業務効率化を狙ったソフトウェア

機能例: スケジュール管理、チャット、設備管理、…

kintone　サイボウズ Office　サイボウズ Garoon　メールワイズ Mailwise

サイボウズの開発研修

- **座学 + チーム体験 + 同期ハッカソン**

講義資料の一部は公開されているので、ぜひチェックしてね
- 2023年のエンジニア新人研修の講義資料を公開しました
- 2022年のエンジニア新人研修の講義資料を公開しました

- チーム体験では、三つのチームでそれぞれ二週間働く
- 僕はインフラQAを体験しました！

インフラQAこと「Platform QA」とは

先月まで「SET」という名称だった

- サイボウズのインフラ事情
  - 自社で契約したデータセンター上に、自社クラウドサービスの基盤を作っている（つまり**プライベートクラウド**）
  - その基盤を操作するプログラム群も自社製（もちろんOSSも活用）

Platform QAでは、上記のプログラム群や、各製品が共通利用する一部基盤サービスに関する品質保証を行う！

本題

今日話したいことはこれ！

- 解析器を作ったこと
- 言語処理系を作ったこと

今日話したいことはこれ！

- 解析器を作ったこと
- 言語処理系を作ったこと

準備: 動的解析 vs 静的解析

- 同じところ
  - プログラムが何らかの性質を満たすか調べる
- 違うところ
  - プログラム実行の有無

性質の例
- `1+1`の実行結果が`2`
- 型エラーを起こさない

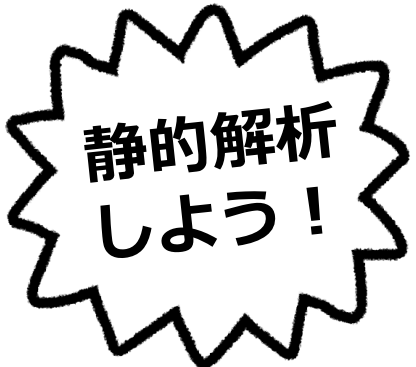
一般に、解析に使える情報は、動的解析の方が多い
（静的解析では情報が抽象化されるため（e.g. `1`を`int`と表現したり））

準備: Lumber

- 自社インフラに開発環境を作成可能な、社内Webサービス
  - お手軽に開発用kintoneを立てたりできる
- Lumberの偉いところ
  - 自社インフラを操作する低レイヤーなプログラムを頑張って使わなくても、ボタンをポチポチすれば簡単に開発環境が作れる！
  - 間違えるリスクも低い！！！

問題点: Lumber自体の設定間違えがち

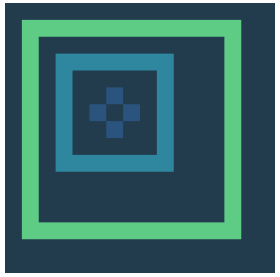
- ある設定ファイルを用いて、Lumberでの環境構築に必要な情報（e.g. VMホスト）を管理している
- **構文エラー**や**VMホストの重複**が起きがち！
  - 設定ファイルの検査方法がまだないから

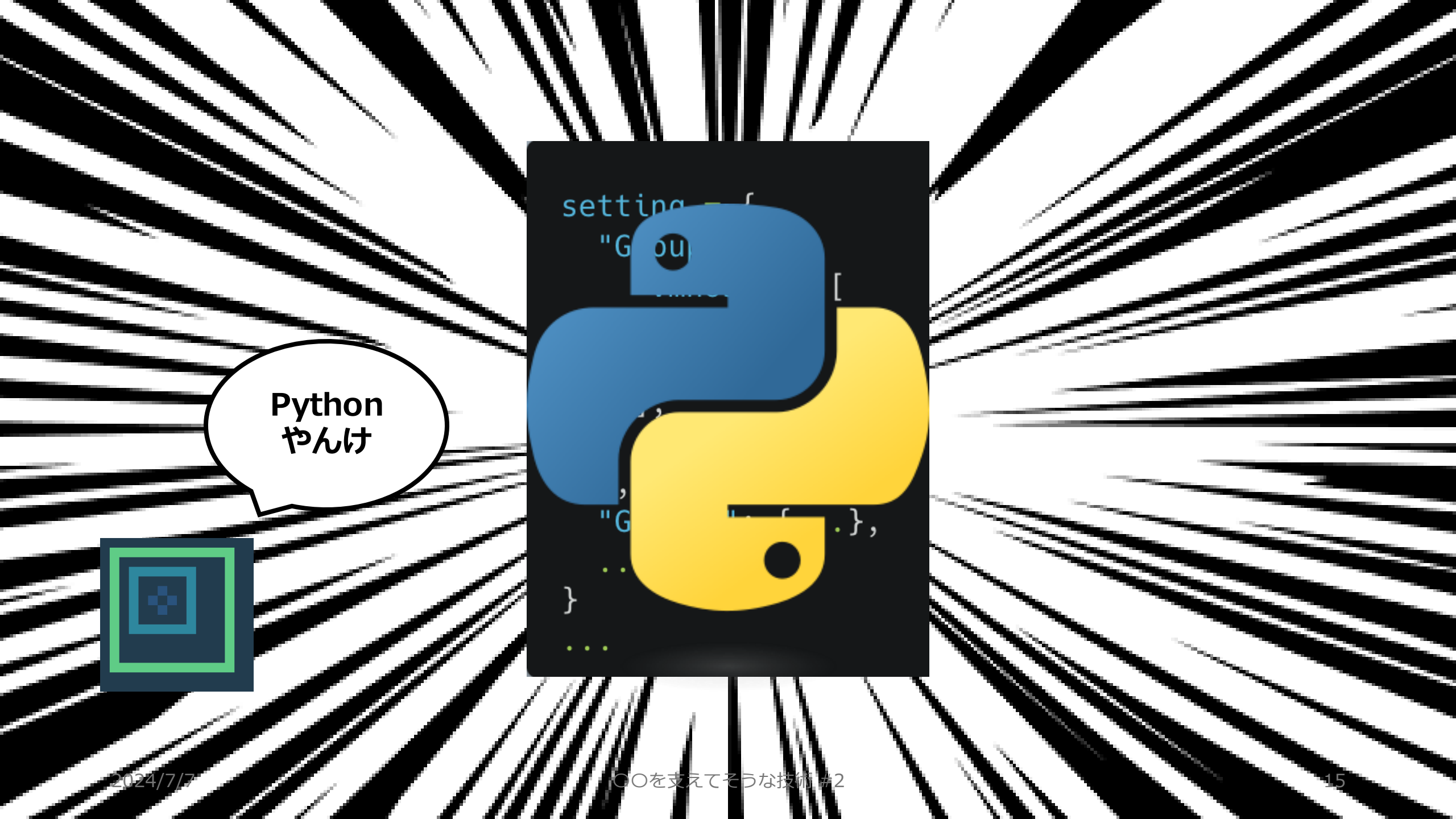



```
setting = {
  "Group1": {
    "vmhosts": [
      "host1",
      "host2"
    ],
    ...
  },
  "Group2": {...},
  ...
}
...
```


setting = {
"Group1": {
"vmhosts": [
"host1",
"host2"
],
...
},
"Group2": {...},
...
}
...

おや……

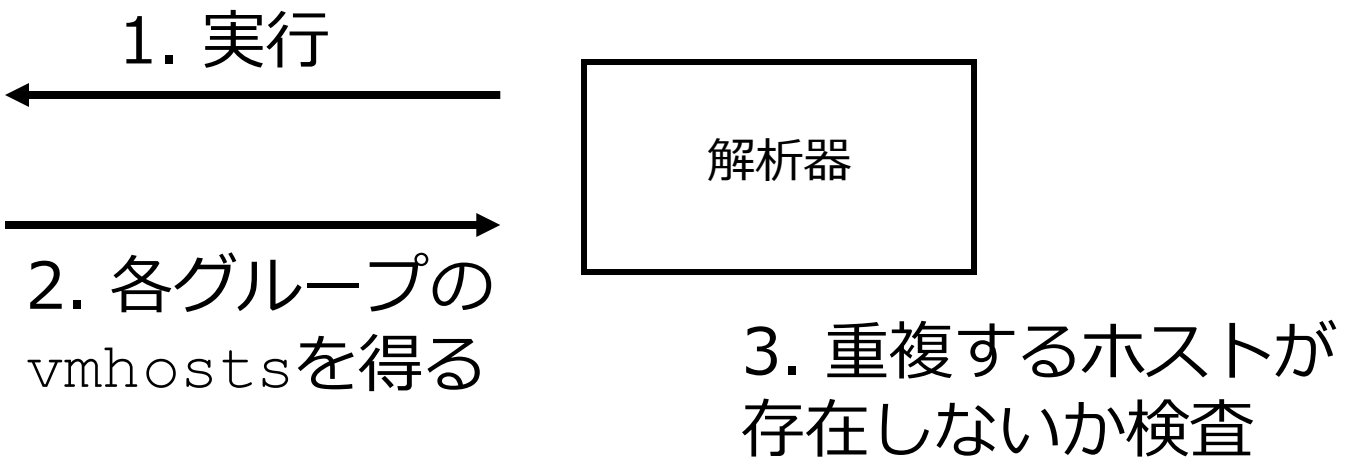



Python
やんけ

解決策: 動的解析で十分だった

- 設定ファイルの中身はPythonプログラムだから、プログラムを実行して中身を直接参照すればよかった

```
setting = {
  "Group1": {
    "vmhosts": [
      "host1",
      "host2"
    ],
    ...
  },
  "Group2": {...},
  ...
}
...
```



🤔「プログラムが停止しなかったり、I/Oを発生させる場合はどうするの？」

```python
while True:
  print("無限ループって怖くね?")
```

停止しない例

```python
import subprocess as sp
sp.run(["rm", "-rf", "/"])
```

I/Oを発生させる例

- 今回のプログラムには、変数定義とリテラル（と文の列）のみ登場すると仮定されていたため、問題にならなかった
  - つまり、プログラムは常に停止し、I/Oを発生させないと仮定できた

ここまでまとめ

- Lumberでは、環境構築に必要な情報（e.g. VMホスト）を設定ファイルで管理しているが、その記述を間違えてしまう場合が少なくなかった
- 当初は静的解析により誤りを検出する方針だったが、よく考えると動的解析で十分だった
  - 動的解析でやっていることは単純なので、僕の一番の貢献は「動的解析でやりたいことできますよ」と指摘した点（だと思ってる）

ちょっと休憩

生産性向上チームの
マスコットキャラクター
「セイサンシャイン君」

もうちょっと休憩

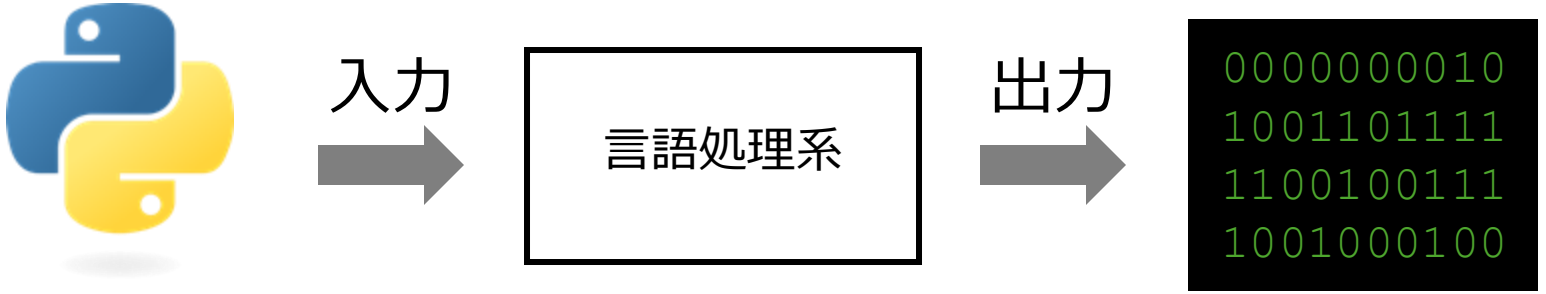
https://zenn.dev/p/cybozu_ept

今日話したいことはこれ！

- 解析器を作ったこと
- 言語処理系を作ったこと

準備: 言語処理系

- プログラムを機械の上で実行するための系（システム）
  - コンパイラとかインタプリターみたいなやつ



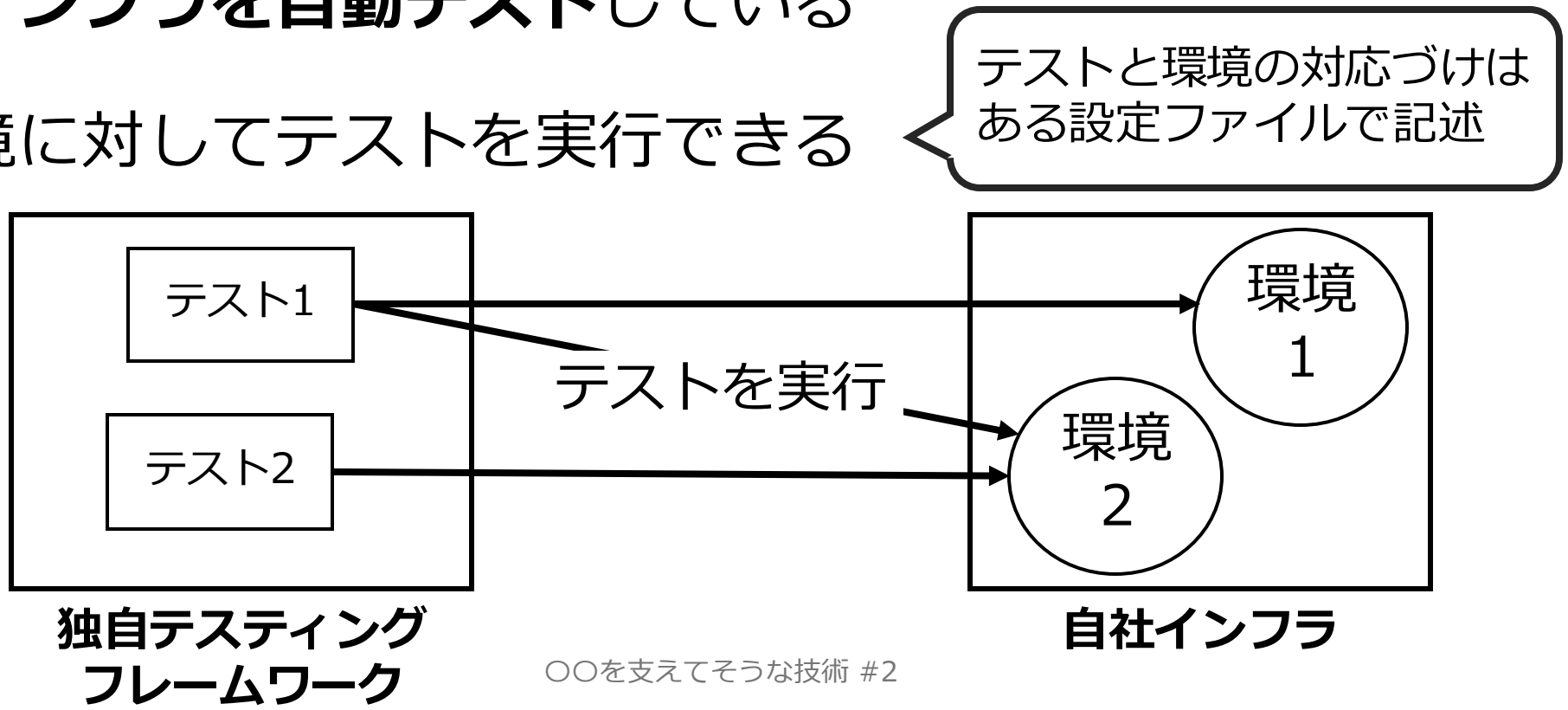
- なお、プログラミング言語は、構文論と意味論で定義される
  - 構文論：どんな形式の文字列か
  - 意味論：その構文はどんな意味か

準備: 自動テスト

- テストとは、プログラムが期待する挙動をするか、**プログラムを実際に実行して確認**すること
  - e.g. `1+1=2` であることを確認するため、`1+1`の実行結果と`2`を比較
- 自動テストで自動化するのは、テストの実行であることが多い
  - そのため、実行するテストは人間が書く必要がある
  - （テストの自動生成も盛んに研究されてはいる）

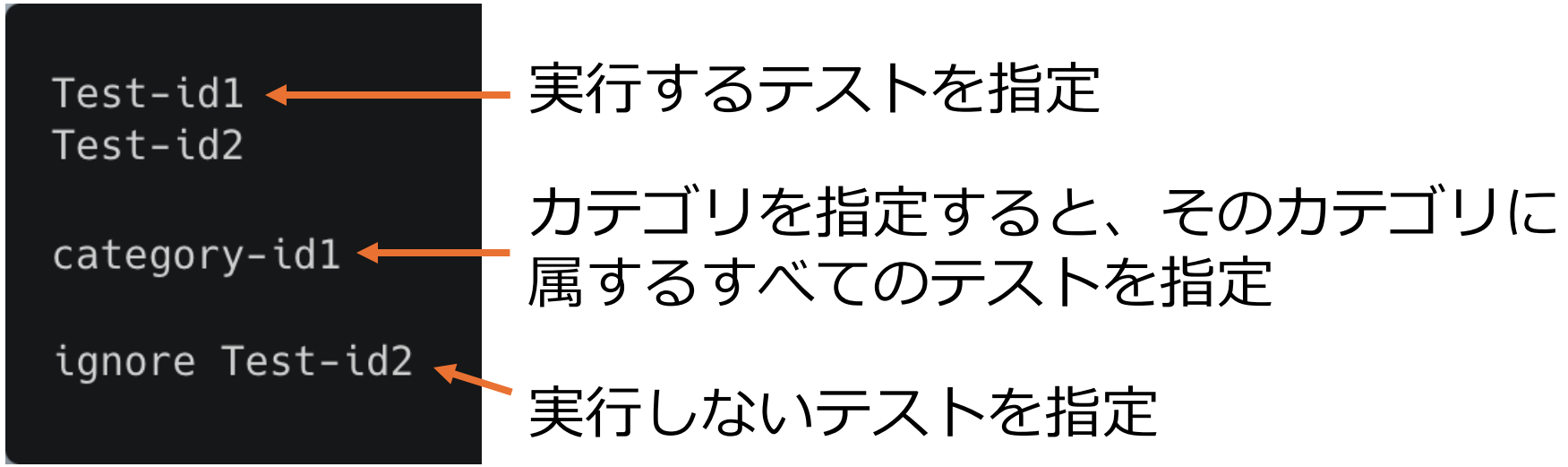
準備:
独自テスティングフレームワーク（1/2）

- Platform QAでは、独自テスティングフレームワークを用いて
  自社インフラを自動テストしている
- 各環境に対してテストを実行できる



準備:
独自テスティングフレームワーク（2/2）

- 1つのファイルで、ある環境に対して実行するテストを記述
  - ここで、カテゴリとはテストの集合（任意のテストは何らかのカテゴリに属している）



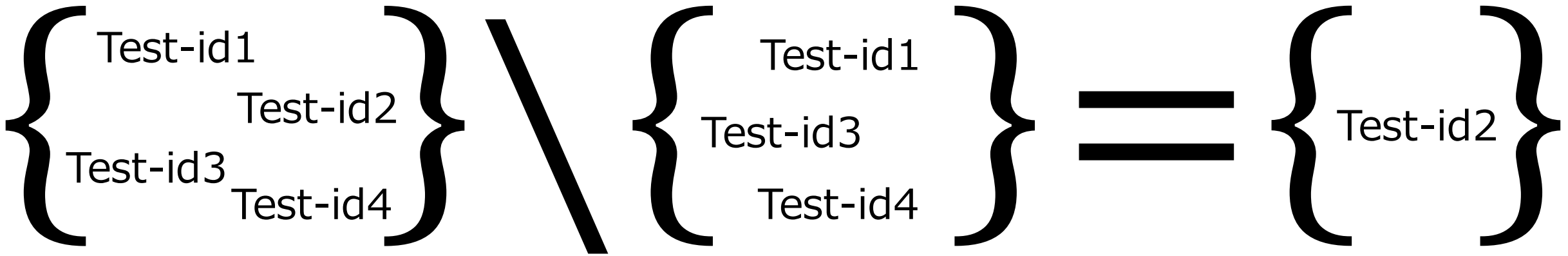
設定ファイル

問題点: テストの実行漏れが起こりがち

- テストの実装と、実際に実行するテストを分けて書くので
  実行されないテストが発生しうる

申し訳程度の七夕要素→

解決策: テストの実装の集合と実行するテストの集合の差集合を取る

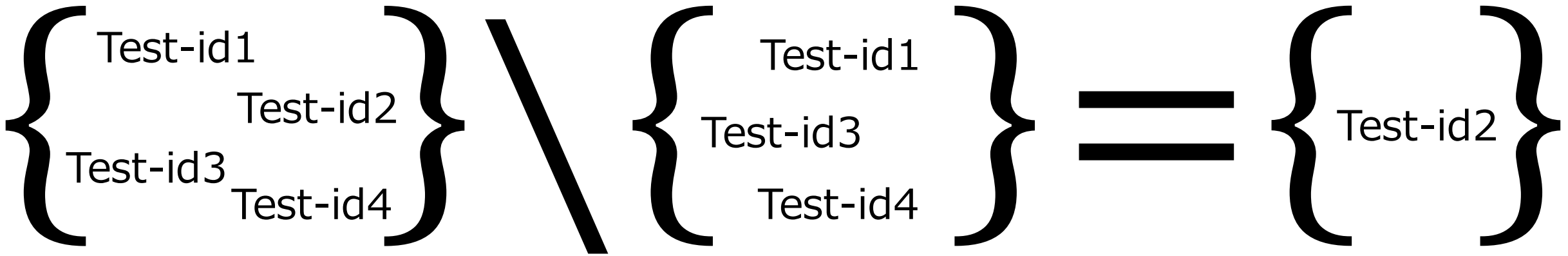


テストの実装の集合

実行するテストの集合

実行されないテストの集合

解決策: テストの実装の集合と実行するテストの集合の差集合を取る



テストの実装の集合

実行するテストの集合

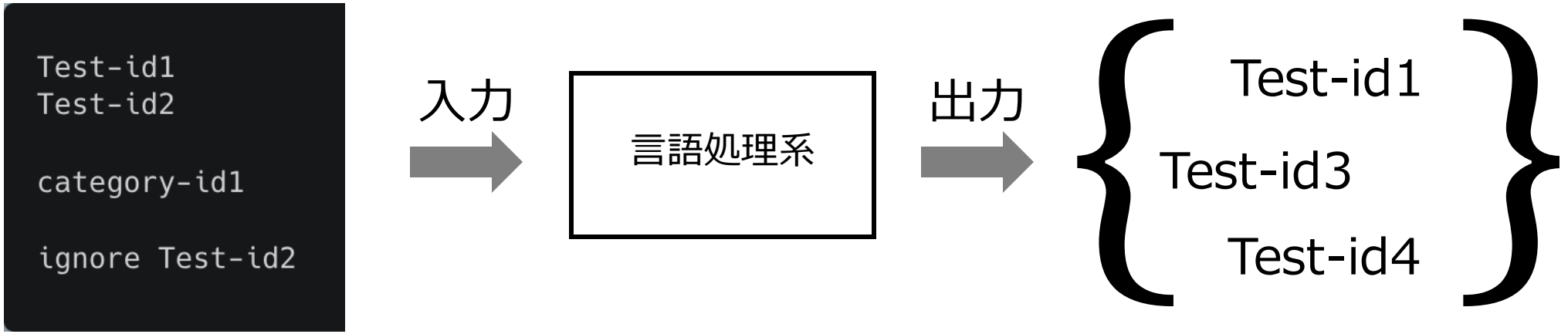
実行されないテストの集合

得るのは簡単（実装のファイル名がIDだから、それを集めるだけ）

得るのは**簡単ではない**（カテゴリ単位でテストを指定できるため、設定ファイルをただ読むだけでは実行するテストがわからない）→**解釈が必要**

解決策: 設定ファイルを解釈する言語処理系を作りました！

- この言語処理系は、設定ファイルを入力すると
  実行するテストの集合を出力する



設定ファイル

実行するテストの集合

形式的な定義もバッチリ

```
<category-id> ::= {<any-character>}+

<test-id>     ::= SRETEST-{<any-character>}+

<statement>   ::= <category-id>              // カテゴリ
                | <test-id>                  // テスト
                | "ignore" <test-id>         // テストの無視

<statements>  ::= ・                        // 空の列
                | <statement> <statements>   // 文の列
```

構文論

$$\frac{testids(c) = ids}{T \vdash c \Downarrow T \cup ids}\ \text{(E-CategoryId)}$$

$$T \vdash t \Downarrow T \cup \{t\}\ \text{(E-TestId)} \qquad T \vdash \texttt{ignore}\ t \Downarrow T \setminus \{t\}\ \text{(E-Ignore)}$$

$$T \vdash \cdot \Downarrow T\ \text{(E-EmptyStatements)}$$

$$\frac{T \vdash s \Downarrow T' \qquad T' \vdash ss \Downarrow T''}{T \vdash s\ ss \Downarrow T''}\ \text{(E-Statements)}$$

意味論

🤔「既存の言語処理系はなかったの？」

- Yes
- 独自テスティングフレームワークの中では、設定ファイルの解釈とテストの実行をひとまとめに行なっており、再利用性が低かった

ここまでまとめ2

- 独自テスティングフレームワークでは、テストの実装と実行するテストを分けて書くため、実行されないテストが起こりえた
- 設定ファイルではカテゴリ単位でテストを指定できるため、設定ファイルをただ読むだけでは実行するテストがわからない
- 設定ファイルから実行するテストの集合に変換する言語処理系を作った

Platform QAチーム体験の感想

- これまで趣味で取り組んできたことを業務に活かせて嬉しい！
  - チーム体験で取り組んだタスクの中で、今日話したタスクが一番大変だった（三日でこなす必要があったから）が、一番楽しくもあった
- SRE見習い見習いになれた
  - 今日は話さなかったが、インフラにより近いタスクもあったから
    - 自社インフラの低レイヤーな操作を課題形式で覚えたり、自社インフラ内の開発環境を利用可能なアカウントを作ったり

まとめ

- サイボウズの開発研修の一環として、Platform QAというインフラQA職の業務に取り組みました
- その中で、動的解析器や言語処理系を作りました
- 趣味で取り組んできたことを業務に活かせてHappy