

AI 実習 2024 最終課題レポート

各回のグループ討議の要約と個人ワーク

[第 13 回～第 15 回]

最終課題

- 課題期間： 2024/7/18(木)
- 課題提出期限： 2024/8/7(水) 18:00
- ファイル名は、 AI-2024A-Final-Report.md （すべて半角英数記号）とすること
- 稽に、ファイル名にマルチバイト日本語を入れて Github に File Upload が失敗するケースがあるため

クラス	学籍番号	氏名
A	20122077	Roger Marvin

レポートの作成手順 (マークダウン記法)

- マークダウン記法[^1][^2][^3][^4]でレポートを作成する
- テキストエディタ+機能拡張をインストール：
- VSCode <https://code.visualstudio.com/download#>
- 機能拡張(Markdown PDF, Markdown Preview Enhanced) セットアップ
- ファイル名は、 **AI実習2024A課題レポート(学籍番号)(学生氏名).md**
- 作成したら PDF ファイル、HTML ファイルを変換生成する

マークダウン記法についてわかりやすい説明、Web 情報

マークダウン記法とは？ Markdown 記法～基礎編～ マークダウン記法一覧 マークダウンの書き方

[^2]: (<https://qiita.com/miriwo/items/28d80f46c857de49f34b>) Markdown 記法～基礎編～ [^3]: (<https://www.sejuku.net/blog/77398>) マークダウン記法一覧 [^4]: (<https://backlog.com/ja/blog/how-to-write-markdown/>) マークダウンの書き方

外部ツール画面の図式引用

- 本様式をひな型とする
- 様式中に、マークダウンのコメントとして **<!-- 要 記述 回答 -->** と記されている箇所は忘れずに適切な記述を加筆する
- 図やスクリーンショットを引用する場合、フォルダにまとめておく
- 1つのレポートにつき、1つのフォルダを用意する
- そのフォルダに、md, pdf, html, および、引用で使用した jpg, png 等ファイルをまとめて配置する
- \$MR^3\$で作成した RDF は、スクリーンショット画像として本文に取り込む
- \$Protege\$で作成したオントロジは、 **OWL/XML Syntax** 形式で、 **file名.owl** として保存する

- \$Protege\$で作成した LOD は、**RDF/XML Syntax** 形式で、**file名.owl** と保存する
- \$Sparql\$のソースコードは、マークダウン形式に、引用によって記述する

Sparqlのクエリコードを `` `sql` と `` ` で囲み、クエリの実行結果も `` `` `` で囲む

レポート提出方法 Github のプライベートリポジトリにアップロード

- 2 学年 4 学期の**API 実習と同じ方法**
- Github のアカウントを作成し、**Practice-AI-2024** という名称でプライベートリポジトリを作成
- そのプライベートリポジトリに、指導員の Github アカウント=**keythrive**を招待する
- Github のプライベートリポジトリに次の名前で、6 つフォルダを用意する：
 - **report1-3**
 - **report4-6**
 - **report7-9**
 - **report10-12**
 - **report13-15**は、**report-Final** に混ぜてよい
-
- Github のアカウント名、プライベートリポジトリ作成、6 つのフォルダをつくったか？招待を完了したか？について、FORMS アンケートするので必ず回答すること。
- FORMS アンケートはこちら：
 - <https://forms.office.com/r/6iMLLYjw1t>
- FORMS アンケートに未回答の場合、レポートを取得する方法が確立しないので、必ず回答のうえレポート提出可能な状態にすること
- それぞれの提出期限までに、必要なファイル一式を当該フォルダにアップロードしておく
- 切時刻を過ぎた時点で自動的に、全員の Github プライベートリポジトリから、**git clone**などでファイルを一括ダウンロードする
- 提出が遅れるとダウンロードできず、未提出と判断される
- 真に止むを得ない事由で、提出期限が遅れる場合、事前にメールにて連絡・相談すること：
- mailto: **horikawa.keitaro@kaishi-pu.ac.jp**
- 事前連絡なしに、期限を過ぎた場合、その課題レポートは未提出として採点しない
- Github のアカウント登録、プライベートリポジトリ、ファイルアップロードが不明な場合は、必ず事前に確認・相談するか、すでに出来ている友達から教えてもらうこと

課題レポートのまとめ方

- 直近のグループ実習 3 回分をまとめて 1 つのレポートを作成する
- 毎回休まずに出席して、グループ討論に積極的に参画する
- グループを代表して発表し、質疑応答、議論、メモを確実にとる作業が大切
- 自グループと他グループの発表をしっかり聴いて、議論模様を簡潔にまとめて報告する
- それぞれの回の全てのグループ発表、および、
- 学生と教員からの質疑コメントを要約する
- ここまでではグループメンバ間の協力作業で、差異化要素はほとんどないことが予想される
- 自作の成果 (RDF, オントロジ等) には極力 "FOAF, SKOS, DC" など共通語彙を適用する
- 個人の努力を差異化要素として、さらに踏み込んだ検討・実習の成果を 3 回分の個人演習について報告してよい**
- 3 回で取り組んだ内容、理解を深めたことを独自レポートとして加筆可能
- 例えば、**作成した RDF, オントロジ, 使用した LOD, 作成した Sparql クエリ, その他の AI 手法やプログラムとの連携技、それらの分析・考察・所感** など

本実習・課題レポートに取り組む意義

- 半年後、本実習を「適当にやり過ごした学生群」と、「真剣に打ち込んで突き詰めた学生群」に明確に分かれることが予想される
- 前者と後者とで、成長の差は著しく広がり、臨地実務実習 II の実習成果および企業担当者から評価が如実に変わる
- 1 年後の今頃、就職活動の内々定数（場合によっては、転職ファストパスの数）が大きく変わることが見込まれる

最終課題

- 第 14 回ー第 15 回 「情報家電」 オントロジを参考に、最終自作オントロジの作成と活用評価を報告する
- 必要な要件は下記のとおり：
- (1) 設計者（貴方）の立場： 新製品や新商品を購入検討中の顧客に対し、適切なアドバイスを提供するコンシェルジェ
- (2) 作成評価するオントロジとそのアプリの要件： 顧客からのどのような問合せ（クエリ）が考えられるか？ ユースケースを具体的に描いて、それに適切に答えられるための必要な知識・概念・語彙をオントロジとして設計する
- 例えば、白物家電（冷蔵庫、エアコン、洗濯機、掃除機、液晶 TV）、スマートフォン、PC、タブレット、電気自動車、の買い替えを想定して、さまざまなニーズに応えるための知識を設計する
- 多数の製品・商品群がある中、数多の選択肢から、顧客が十分に満足する製品をどういう条件で絞り込むか？
- (3) 購入要件を満たす製品の絞り込みを可能とする \$Sparql クエリ \$ の仕様設計
- クエリに整合した、概念・用語・語彙と、その関係整理を製品・製品バリエーション・要求仕様として表現する
- (4) 実際の具体的製品名 (LOD)、代表的な属性・共通語彙を使ってオントロジに反映する
- (5) 実際に Protege を用いたオントロジの作成、Sparql の動作を確認

- (6) オントロジの評価： 購入要件に複数のバリエーションを用意して、複数パターンのクエリと結果の実例を示すこと。そのうえで、製品購入へのアドバイスの網羅性・精度・頻度を評価考察する
 - オントロジを用いた推論は、クラスの同一性、排他性などによるクラス体系整理のレベルであるが、利用価値の高い\$Sparql\$や\$Prolog\$による述語論理の検討・提案ができれば加点要素とする
 - また、個人作業の高度化の一環で、\$Sparql\$や\$Prolog\$の論理演算を DSL として、他の言語から呼び出す形態で活用してもよい
 - 更なる加点要素：Virtuoso を用いた RDF ストア化および、Sparql エンドポイントのローカル動作確認
-

最終課題の記載事項

作成したオントロジ名：対象とした新製品・新商品名

オントロジ名	対象とした新製品
あらゆる種類の機械オンライントロジー	3D プリンター、CNC マシン、レザーカッター、PCB プリンター、掘削機、グライダーラー、テーブルソー

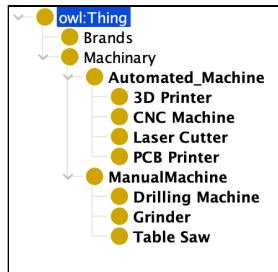
想定した顧客からの具体的質問、ユースケース事例

想定するユーザー	ユースケース事例
ある施設の管理者	働いている施設で機械を交換する際や上司に新しい機械の購入を依頼された際に、このオントロジーを使えば、検索が迅速にでき、機械の重要な情報を全て把握できます。そのため、簡単に上司に報告することができ、いちいち情報を探す手間が省けます。
ある施設の利用者	ある施設の機械を自分の家や部屋に設置したい時に、このオントロジーを使えば、自分の予算や希望する機械のスペックを迅速かつ確実に検索し、安心して購入することができます。
機械に関する初心者	機械に興味はあるが、機械に関する知識が全くない場合でも、このオントロジーを使えば、機械の重要な情報を得ることができます。そのため、後悔することなく機械を購入することができます。

Protege を用いたオントロジの図式

- **protégé**にて作成された、オントロジファイル (.owl) 形式の成果物
- 画面スナップショット (.jpg, .png)

オントロジ Class



オントロジー Individual

Anycubic_Kobra_2_Plus

Description: A 3D Printer from Anycubic.

Annotations:

- Types: 3D Printer
- Object property assertions: hasBrand Anycubic
- Data property assertions:
 - hasLayerResolution "0.1"
 - hasModel "Kobra 2 Plus"
 - hasLink "https://wwwanycubic.com/products/kobra-2-plus-3d-printer"
 - hasTechnology "FDM"
 - hasBuildVolume "320x320x400"
 - hasWattage "1000"
 - hasImage "https://m.media-amazon.com/images/I/61tfucfWIL_AC_UF894,1000_QL80.jpg"
 - hasPrice "\$2,999"
 - hasMaterialCompatibility "PLA, PETG, TPU, PVA"
 - hasVoltage "220"
 - hasWeight "13"
- Negative object property assertions: None
- Negative data property assertions: None

オントロジー Property

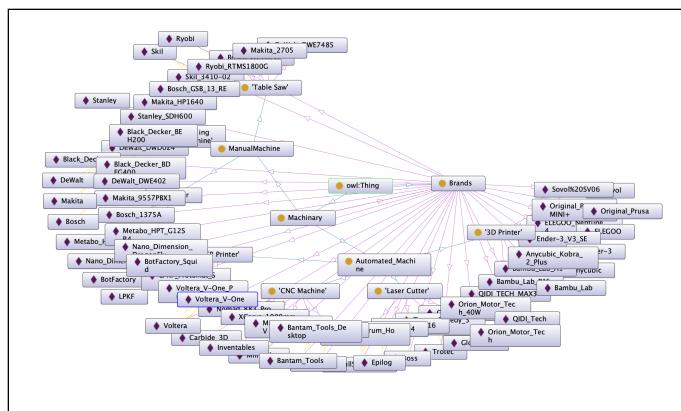
Data property hierarchy:

- owl:topDataProperty
- hasBuildVolume
- hasBuildVolume (asserted)
 - Annotations: hasBuildVolume
 - Characteristics: Functional
 - Description: The build volume of the 3D printer (e.g., 220x220x250 mm)
 - Annotations: hasBuildVolume
 - Characteristics: Functional
 - Description: The build volume of the 3D printer (e.g., 220x220x250 mm)

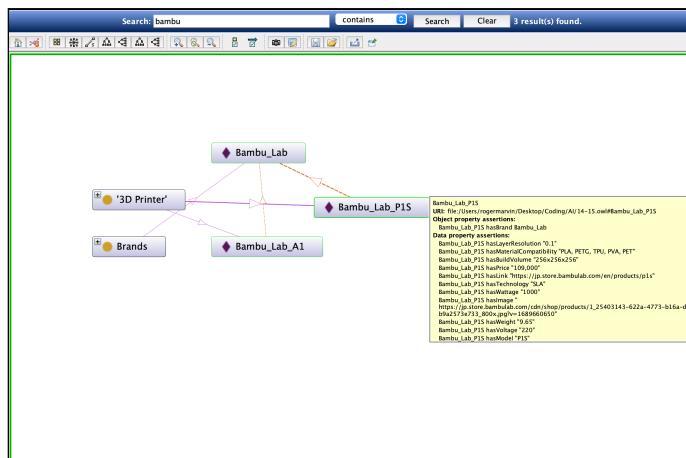
Object property hierarchy:

- owl:topObjectProperty
- hasBrand (asserted)
 - Annotations: hasBrand
 - Characteristics: Functional
 - Description: Links a machine to its brand.
 - Annotations: hasBrand
 - Characteristics: Functional
 - Description: Links a machine to its brand.

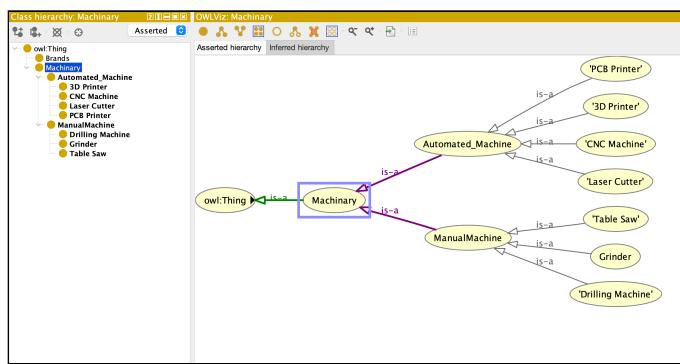
オントロジー OntoGraf



オントロジー OntoGraf で「bamboo」を検索した結果



オントロジー OwlViz



オントロジー Protege SPARQL

The screenshot shows the Protege SPARQL interface with a query for 'Brands'. The query uses PREFIX declarations for various namespaces and SELECT DISTINCT to find unique brands. It includes variables ?brand and ?brandLabel, and filters for object properties like hasBrand. The query is as follows:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX PROTEGE: <http://www.owl-ontologies.com/PROTEGE.owl#>
SELECT DISTINCT ?brand
WHERE {
    ?brand a owl:Class .
    ?brand rdfs:label ?brandLabel .
    ?brand hasBrand ?brand .
}
  
```

The screenshot shows the Protege SPARQL interface with a query for 'Individuals'. The query uses PREFIX declarations for various namespaces and SELECT DISTINCT to find unique individuals. It includes variables ?individual and ?individualLabel, and filters for object properties like hasIndividual. The query is as follows:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX PROTEGE: <http://www.owl-ontologies.com/PROTEGE.owl#>
SELECT DISTINCT ?individual
WHERE {
    ?individual a owl:NamedIndividual .
    ?individual rdfs:label ?individualLabel .
    ?individual hasIndividual ?individual .
}
  
```

Sparql の実施例とその結果 「Virtuoso 環境」

クエリ 01 「商品名（インスタンス）でインスタンスのプロパティを検索するクエリ」

3D プリンター（入力：「bamboo」）の値段が 50,000 円以上 120,000 円以下のものを検索する

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>

SELECT DISTINCT ?individual ?property ?value
WHERE {
    ?individual owl1:hasBrand ?brand .
    ?individual ?property ?value .

    BIND(STRAFTER(STR(?individual), "#") AS ?individualName)
    FILTER(REGEX(?individualName, "bamboo", "i"))
    FILTER(?property != rdf:type)

    ?individual owl1:hasPrice ?price .
    BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)
    FILTER(?numericPrice >= 50000 && ?numericPrice <= 120000)

    OPTIONAL {
        ?brand rdfs:label ?brandName .
    }
}
ORDER BY ?individual ?property
```

説明 01

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>
```

これらの部分は、よく使用される名前空間のプレフィックスを定義します。

```
SELECT DISTINCT ?individual ?property ?value
```

この行は、クエリによって返される変数 (?individual、?property、および ?value) を指定します。 DISTINCT は重複する結果が削除されることを保証します。

```
BIND(STRAFTER(STR(?individual), "#") AS ?individualName)
FILTER(REGEX(?individualName, "bamboo", "i"))
FILTER(?property != rdf:type)
```

- 「BIND(STRAFTER(STR(?individual), "#") AS ?individualName)」は、個体のローカル名 (# の後の部分) を抽出します。
- 「FILTER(REGEX(?individualName, "bamboo", "i"))」は、名前に "bamboo" が含まれる個人をフィルタリングします (大文字と小文字は区別されません)。
- 「FILTER(?property != rdf:type)」は、結果から rdf:type プロパティを除外します。

```
?individual owl1:hasPrice ?price .
BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)
FILTER(?numericPrice >= 50000 && ?numericPrice <= 120000)
```

- 「?individual owl1:hasPrice ?price .」は、hasPrice プロパティを持つ個体と一致します。
- 「BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)」は、価格文字列から数値を抽出します。
- 「FILTER(?numericPrice >= 50000 && ?numericPrice <= 120000)」は、価格が 50,000 から 120,000 の間にある個体をフィルタリングします。

```
OPTIONAL {
    ?brand rdfs:label ?brandName .
}
```

- 「OPTIONAL{ ?brand rdfs:label ?brandName . }」は、ブランドのラベルが存在する場合に一致を試みます。

結果 01

The screenshot shows the SPARQL Execution interface with the following details:

Query:

```
PREFIX owl1: <http://www.w3.org/ns/OWL#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?individual ?brandName
WHERE {
    ?individual owl1:hasPrice ?price .
    BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)
    FILTER(?numericPrice >= 50000 && ?numericPrice <= 120000)

    OPTIONAL {
        ?brand rdfs:label ?brandName .
    }
}
```

Results:

individual	brandName
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_A1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_B1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_C1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_D1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_E1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_F1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_G1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_H1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_I1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_J1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_K1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_L1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_M1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_N1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_O1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_P1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_Q1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_R1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_S1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_T1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_U1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_V1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_W1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_X1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_Y1	
http://www.semanticweb.org/Desktop/Codes/N14/15.owl#bamboo_Lab_Z1	

クエリ 02 「商品カテゴリ（クラス）で商品（インスタンス）のプロパティを検索するクエリ」

3D プリンター（入力：「3d」）の値段が 10,000 円以上 120,000 円以下のものを検索する

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>

SELECT DISTINCT ?individual ?property ?value
WHERE {
    ?individual rdf:type ?category .
    ?category rdfs:label ?categoryLabel .
    FILTER(CONTAINS(LCASE(STR(?categoryLabel)), LCASE("3d")))

    ?individual owl1:hasBrand ?brand .
    ?individual ?property ?value .
    FILTER(?property != rdf:type)

    ?individual owl1:hasPrice ?price .
    BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)
    FILTER(?numericPrice >= 10000 && ?numericPrice <= 120000)

    OPTIONAL {
        ?brand rdfs:label ?brandName .
    }
}
ORDER BY ?individual ?property

```

説明 02

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>

```

これらの部分は、よく使用される名前空間のプレフィックスを定義します。

```
SELECT DISTINCT ?individual ?property ?value
```

この行は、クエリによって返される変数 (?individual、?property、および ?value) を指定します。 DISTINCT は重複する結果が削除されることを保証します。

```
?individual rdf:type ?category .
?category rdfs:label ?categoryLabel .
```

```
FILTER(CONTAINS(LCASE(STR(?categoryLabel)), LCASE("3d"))))
```

- 「?individual rdf:type ?category .」は、個体が特定のカテゴリに属することを示します。
- 「?category rdfs:label ?categoryLabel .」は、カテゴリのラベルを取得します。
- 「FILTER(CONTAINS(LCASE(STR(?categoryLabel)), LCASE("3d"))))」は、カテゴリラベルに "3d" が含まれる個体をフィルタリングします (大文字と小文字は区別されません)。

```
?individual owl1:hasBrand ?brand .
?individual ?property ?value .
FILTER(?property != rdf:type)
```

- 「?individual owl1:hasBrand ?brand .」は、個体が特定のブランドを持つことを示します。
- 「?individual ?property ?value .」は、個体のプロパティとその値を取得します。
- 「FILTER(?property != rdf:type)」は、結果から rdf:type プロパティを除外します。

```
?individual owl1:hasPrice ?price .
BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)
FILTER(?numericPrice >= 10000 && ?numericPrice <= 120000)
```

- 「?individual owl1:hasPrice ?price .」は、hasPrice プロパティを持つ個体と一致します。
- 「BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?numericPrice)」は、価格文字列から数値を抽出します。
- 「FILTER(?numericPrice >= 10000 && ?numericPrice <= 120000)」は、価格が 10,000 から 120,000 の間にある個体をフィルタリングします。

```
OPTIONAL {
    ?brand rdfs:label ?brandName .
}
```

- 「OPTIONAL{ ?brand rdfs:label ?brandName . }」は、ブランドのラベルが存在する場合に一致を試みます。

結果 02

その他クエリ（クエリ実行のその他のパターン）

1. すべてのクラスを一覧表示する

- ユースケース：ユーザーが商品のカテゴリーや種類を調べたいときに使います。また、商品のメーカーリストを調べたい時に使います。

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

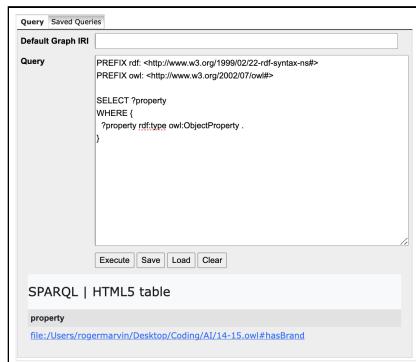
```
SELECT ?class
WHERE {
  ?class rdf:type owl:Class
}
```

2. すべてのオブジェクトプロパティを一覧表示する

- ユースケース：ユーザーが商品のスペックリスト（プロパティ）を検索するときに使います。特に商品とメーカーの関係性を調べたい時に使います。

PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>
PREFIX owl: <<http://www.w3.org/2002/07/owl#>>

```
SELECT ?property
WHERE {
    ?property rdf:type owl:ObjectProperty .
}
```



3. すべてのデータプロパティを一覧表示する

- ユースケース：ユーザーが商品のスペックリスト（データプロパティ）を検索するときに使います。商品ではどういうスペックの情報があるのかを知るためです。

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
SELECT ?property
WHERE {
    ?property rdf:type owl:DatatypeProperty .
}
```

The screenshot shows a SPARQL query interface with the following details:

- Default Graph IRI:** An empty input field.
- Query:**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?property
WHERE {
  ?property rdf:type owl:DatatypeProperty .
}
```
- Buttons:** Execute, Save, Load, Clear.
- SPARQL | HTML5 table:**

property
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasBuildVolume
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasCuttingArea
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasLaserPower
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasLayerResolution
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasMaterialCompatibility
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasModel
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasPower
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasPrice
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasPrintArea
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasPrintResolution
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasSpeed
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasSpindlePower
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasTechnology
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasWeight
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasWorkArea
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasImage
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasLink
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasVoltage
file:/Users/rogernmarvin/Desktop/Coding/AI/14-15.owl#hasWattage

4. すべての Annotation プロパティを一覧表示する

- ユースケース：ユーザーが商品のスペックリスト（データプロパティ）を検索するときに使います。商品ではどういう情報があるのかを知るためです。

The screenshot shows a SPARQL query interface with the following details:

- PREFIX:**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```
- Query:**

```
SELECT ?property
WHERE {
  ?property rdf:type owl:AnnotationProperty .
}
```

The screenshot shows a SPARQL query editor interface. The query pane contains the following SPARQL code:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?property
WHERE {
  ?property rdf:type owl:AnnotationProperty .
}

```

The results pane displays a list of RDF properties from the W3C schema namespace:

- <http://www.w3.org/2000/01/rdf-schema#label>
- <http://www.w3.org/2000/01/rdf-schema#domain>
- <http://www.w3.org/2000/01/rdf-schema#isDefinedBy>
- <http://www.w3.org/2000/01/rdf-schema#comment>
- <http://www.w3.org/2002/07/owl#versionInfo>
- <http://www.w3.org/2000/01/rdf-schema#seeAlso>
- <http://www.w3.org/2002/07/hasBrand>
- <http://www.w3.org/2002/07/hasBuildVolume>
- <http://www.w3.org/2002/07/hasCuttingArea>
- <http://www.w3.org/2002/07/hasLaserPower>
- <http://www.w3.org/2002/07/hasLayerResolution>
- <http://www.w3.org/2002/07/hasMaterialCompatibility>
- <http://www.w3.org/2002/07/hasModel>
- <http://www.w3.org/2002/07/hasPrice>
- <http://www.w3.org/2002/07/hasPrintArea>
- <http://www.w3.org/2002/07/hasPrintResolution>
- <http://www.w3.org/2002/07/hasSpindlePower>
- <http://www.w3.org/2002/07/hasTechnology>
- <http://www.w3.org/2002/07/hasWeight>

5. すべてのインスタンスをリストする

- ユースケース：ユーザーが商品のリストを検索するために使用しています。

The screenshot shows a SPARQL query editor interface. The query pane contains the following SPARQL code:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?individual
WHERE {
  ?individual rdf:type owl:NamedIndividual .
}

```

The screenshot shows a SPARQL query editor interface. The query pane contains the following SPARQL code:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?individual
WHERE {
  ?individual rdf:type owl:NamedIndividual .
}

```

The results pane displays a list of individual instances found in the dataset:

- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#ELEGOO_Neptune_4
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Anycubic_Kobra_2_Plus
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bambu_Lab_A1
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bambu_Lab_P1S
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bantam_Tools_Desktop
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bosch_1375A
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bosch_GSB_13_E
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bosch_GTS1021
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Boss_L5_1416
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#BotFactory_Squid
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#DeWalt_DWD024
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#DeWalt_DWE402
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#DeWalt_DWE405
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Ender-3_V3_SE
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Fillog_Zing_24
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#FullSpectrum_Hobby
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Glowforge_Pro
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#LPKF_ProtoMat_S104
- file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Makita_2705

6. 特定の個体のすべてのプロパティを一覧表示する「Anycubic Kobra 2 Plus の場合」

- ユースケース：「Anycubic Kobra 2 Plus」を調べて、商品のスペック（データやオブジェクトプロパティ）を調べるために使用しています。

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
SELECT ?property ?value
WHERE {
  <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Anycubic_Kobra_2_Plus> ?property ?value .
}
```

The screenshot shows a SPARQL query interface with the following details:

Query (SQL View)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?property ?value
WHERE {
  <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Anycubic_Kobra_2_Plus> ?property ?value
}
```

SPARQL | HTML5 table

property	value
http://www.w3.org/1999/02/22-rdf-syntax-ns#label	http://www.w3.org/2002/07/owl#Label
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://UserRequirement/DesktopCode#A14-15.owl#headband	http://UserRequirement/DesktopCode#A14-15.owl#Announces
http://UserRequirement/DesktopCode#A14-15.owl#body	"720x360MM"
http://UserRequirement/DesktopCode#A14-15.owl#material	"PLA, PETG, TPU, PVH"
http://UserRequirement/DesktopCode#A14-15.owl#extruderCompatibility	"Multi 2 Extr."
http://UserRequirement/DesktopCode#A14-15.owl#extruderType	"T2, T90"
http://UserRequirement/DesktopCode#A14-15.owl#heatBedType	"V6"
http://UserRequirement/DesktopCode#A14-15.owl#heatBedSize	"13"
http://UserRequirement/DesktopCode#A14-15.owl#featureImage	https://www.3dprintable.com/thumb/2/9337/1400x1400_3d_m_3d_14_15_anycubic_kobra_2_plus_3d_printer_720x360mm_v6_t2_t90_v6_1400x1400.jpg
http://UserRequirement/DesktopCode#A14-15.owl#hasImage	"720"

7. 特定のクラスのすべての個体をリストします「3DPrinter の場合」

- ユースケース：「3DPrinter」というカテゴリー（クラス）から調べて、商品のリストを調べるために使用しています。

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
SELECT ?individual
WHERE {
  ?individual rdf:type <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#3DPrinter> .
```

The screenshot shows a SPARQL query interface with the following details:

- Query**: SPARQL query for individuals.
- Default Graph IRI**: [http://www.w3.org/1999/02/22-rdf-syntax-ns#](#)
- Results**: An HTML5 table with the following data:

URI	Label
http://www.w3.org/2002/07/owl#owlIf	owlIf
http://www.w3.org/2002/07/owl#3DPrinter	3DPrinter
http://www.w3.org/2002/07/owl#Bamboo_Lab	Bamboo_Lab
http://www.w3.org/2002/07/owl#Original_Prusa_MINI	Original_Prusa_MINI
http://www.w3.org/2002/07/owl#QIDI_TECH_MAX3	QIDI_TECH_MAX3
http://www.w3.org/2002/07/owl#Savor_20SV06	Savor_20SV06
- Buttons**: Execute, Save, Load, Clear.

8. すべてのクラスとそのサブクラスを一覧表示します

- ユースケース：商品のカテゴリーのカテゴリーとサブカテゴリーを調べるために使用しています。

PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>
PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

```
SELECT ?class ?subclass  
WHERE {  
    ?subclass rdfs:subClassOf ?class  
}
```

Detail Graph IR	
Query	PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> SELECT ?name ?success WHERE { ?name rdfs:label ?label ?label xsd:type xsd:integer ?label rdfs:isDefinedBy ?url ?url rdfs:label ?success }.
Execute	Save
Load	Clear

SPARQL | HTML5 table

class	subClass
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property	http://www.w3.org/2001/07/rdf-schema#Property http://www.w3.org/2001/07/rdf-schema#Datatype http://www.w3.org/2001/07/rdf-schema#Container http://www.w3.org/2001/07/rdf-schema#List http://www.w3.org/2001/07/rdf-schema#Resource http://www.w3.org/2001/07/rdf-schema#Seq http://www.w3.org/2001/07/rdf-schema#Set http://www.w3.org/2001/07/rdf-schema#Bag http://www.w3.org/2001/07/rdf-schema#Alt http://www.w3.org/2001/07/rdf-schema#Class http://www.w3.org/2001/07/rdf-schema#List http://www.w3.org/2001/07/rdf-schema#Resource http://www.w3.org/2001/07/rdf-schema#Seq http://www.w3.org/2001/07/rdf-schema#Set http://www.w3.org/2001/07/rdf-schema#Bag http://www.w3.org/2001/07/rdf-schema#Alt
http://www.w3.org/1999/02/22-rdf-syntax-ns#NodeValue	http://www.w3.org/2001/07/rdf-schema#Boolean http://www.w3.org/2001/07/rdf-schema#Integer http://www.w3.org/2001/07/rdf-schema#Float http://www.w3.org/2001/07/rdf-schema#Double http://www.w3.org/2001/07/rdf-schema#String http://www.w3.org/2001/07/rdf-schema#Binary http://www.w3.org/2001/07/rdf-schema#Date http://www.w3.org/2001/07/rdf-schema#Time http://www.w3.org/2001/07/rdf-schema#DateTime http://www.w3.org/2001/07/rdf-schema#XML http://www.w3.org/2001/07/rdf-schema#URI
http://www.w3.org/1999/02/22-rdf-syntax-ns#BlankNode	http://www.w3.org/2001/07/rdf-schema#BlankNode
http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal	http://www.w3.org/2001/07/rdf-schema#Literal
http://www.w3.org/1999/02/22-rdf-syntax-ns#Container	http://www.w3.org/2001/07/rdf-schema#Container
http://www.w3.org/1999/02/22-rdf-syntax-ns#ContainerContent	http://www.w3.org/2001/07/rdf-schema#ContainerContent
http://www.co-ode.org/ns/packagelock.owl#Automated_Machine	http://www.co-ode.org/ns/packagelock.owl#Automated_Machine http://www.co-ode.org/ns/packagelock.owl#UserInteractive_Machine http://www.co-ode.org/ns/packagelock.owl#UserInteractive_Machine
http://www.co-ode.org/ns/packagelock.owl#UserInteractive_Machine	http://www.co-ode.org/ns/packagelock.owl#UserInteractive_Machine

9. ブランドが「bamboo」の機械を検索する

- ユースケース：ユーザーが機械のブランドを調べるために使用しています。

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
SELECT ?individual ?brand
WHERE {
    ?individual rdf:type owl:NamedIndividual .
    ?individual owl:hasBrand ?brand
```

```

FILTER regex(str(?brand), "bamboo", "i")
}

```

The screenshot shows a SPARQL query editor interface. The query text is:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?individual ?brand
WHERE {
?individual rdf:type owl:NamedIndividual .
?individual owl1:hasBrand ?brand .

FILTER regex(str(?brand), "bamboo", "i")
}

```

Below the query text are four buttons: Execute, Save, Load, and Clear.

Underneath the query editor is a table titled "SPARQL | HTML5 table". The table has two columns: "individual" and "brand". The data rows are:

individual	brand
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab_A1	file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab_P1S	file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab

10. 重量の範囲により機械を検索する

- ユースケース：ユーザーが機械の重量の範囲を調べるために使用しています。

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?individual ?weight
WHERE {
?individual rdf:type owl:NamedIndividual .
?individual owl1:hasWeight ?weight .

FILTER (xsd:decimal(?weight) >= 5 && xsd:decimal(?weight) <= 15)
}

```

The screenshot shows a SPARQL query editor interface. The query text is:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?individual ?weight
WHERE {
?individual rdf:type owl:NamedIndividual .
?individual owl1:hasWeight ?weight .

FILTER (xsd:decimal(?weight) >= 5 && xsd:decimal(?weight) <= 15)
}

```

Below the query text are four buttons: Execute, Save, Load, and Clear.

Underneath the query editor is a table titled "SPARQL | HTML5 table". The table has two columns: "individual" and "weight". The data rows are:

individual	weight
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#ELEGOO_Neptune_4	"8.3"
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Anycubic_Kobra_2_Plus	"13"
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab_A1	"8.3"
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Bamboo_Lab_P1S	"9.65"
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Ender_3_V3_SE	"7.34"
file:/Users/rogermarvin/Desktop/Coding/AI/14-15.owl#Sovol%20SV05	"10.5"

作成したオントロジの規模、その活用規模を表す数値を列挙する

- 設計したクラスの数
- 設計したインスタンスの数

- 設計したプロパティの数
- .owl ファイルの行数 (XML ファイルの行数を wc でカウント、テキストエディタで確認)
- Sparql のクエリ数 (クエリ実行のパターン数)
- トータルの Sparql 行数

定量パラメタ	数値
クラス数	11
インスタンス数	69
プロパティ数	21
.owl 行数	1454
Sparql クエリ数	12
Sparql 行数	108

分析・評価・考察

- 選定理由、なぜこのテーマを選定したか？

実際に大学の施設で使用する機械のことに関してもっと深く知りたいと思い、このテーマを選定しました。
また、その施設の管理者が機械の情報を加速で検索出来れば、機械の管理がしやすくなると思い、このテーマを選定しました。

- 深く専門的な知識の所在

開志ラボの施設の機械を利用しながら、機械の知識をもらいました。もちろん、開志ラボの施設の管理者と協力して、情報をえります。
また、今回のオントロジーでは2年生の知識表現の講義で学んだもので作って、API実習やSQLの講義で学んだものを活用して、今回のSPARQLクエリを作って、実際にアプリケーションしました。

- ユーザへの寄与・貢献、ユーザにどのように活用してほしいか

このオントロジーを利用してすることで、施設の管理者や利用者が機械の情報を迅速に検索できるようになります。
特に、機械の購入やメンテナンスに関する情報を簡単に取得できるため、管理業務の効率化が期待されます。
また、機械に詳しくないユーザーでも直感的に情報を検索できるように設計されています。

- 差異化要素、強みと特徴点

このオントロジーの強みは、詳細な機械情報を網羅している点と、検索のが早いです。
特に、機械のブランド、モデル、技術仕様、価格などの情報を一元管理しているため、ユーザーは必

必要な情報を迅速に取得できます。

また、SPARQLクエリを用いた検索機能により、複雑な条件でもすぐに検索結果を得ることができます。

- 共通語彙、プロパティの使い方で自ら発見した知見

共通語彙を使用することで、異なるデータソース間での情報統合が容易になることを発見しました。また、データプロパティとオブジェクトプロパティの違いを理解でき、どっちはどっちに使うのかを理解できました。

データは単体のインスタンスの情報をつけるためです。

一方、オブジェクトは一つのインスタンスと別のインスタンスの関係性を表すためのものです。

- クラスとインスタンスの違いで発見した知見

2年生の時に、クラスを勘違いして、インスタンスをクラスにする時があります。

それを3年生でたくさんオントロジーを作ったおかげで、自分がどっちはクラスなのか、どっちはインスタンスなのかを理解できました。

もちろん、今回で作ったオントロジーでもうまく書きました。

- オントロジ設計で貰ったポリシー

オントロジ設計では、ユーザーが直感的に情報を検索できるようにすることを重視しました。

また、データの一貫性と再利用性を確保するために、共通語彙と標準プロパティを使用しました。

さらに、検索の高速性を確保するために、SPARQLクエリの最適化にも注力しました。

- 創意工夫のポイント

機械の詳細情報を網羅するために、各プロパティの定義を細かく行いました。

また、ユーザーが直感的に情報を検索できるように、検索するアプリケーションを開発でき、実際にSPARQLのAPIを利用でき、連携できました。

さらに、SPARQLクエリを用いた検索機能を実装することで、複雑な条件でもすぐに検索結果を得られるようになりました。

- オントロジエディタの使い方で習得した技法

オントロジエディタを使用して、クラスやプロパティの定義、インスタンスの作成を効率的に行う技法を習得しました。

また、SPARQLクエリの作成と実行を通じて、データの検索と取得の方法を学びました。

さらに、Protegeだけではなくて、VirtuosoやWEB Protegeなどを利用でき、事情により利用できました。

これにより、オントロジの設計と実装のスキルを向上させることができました。

- 作業を通じて得られた知見

オントロジーを用いたデータ管理の有効性を実感しました。特に、データの一貫性と再利用性が向上し、検索の高速性が確保できることを確認しました。また、共通語彙を使用することで、異なるデータソース間での情報統合が容易になることを学びました。これにより、今後のデータ管理や検索システムの設計に役立つ知見を得ることができました。また、実際にオントロジーと他のサービスと連携して、機械の検索サービスを作成しました。その際に、オントロジーの API を取得し、連携する方法や、SPARQL クエリを用いた検索機能の実装方法を学びました。

評価の観点

- 独創性と丁寧な設計
- 差異化性（既存と異なる付加価値）
- 深さ（表面的でなく、知識に深みがある）
- 設計思想（どのような思いでデザインしたか）
- 見通しの良さ、設計のわかりやすさ
- is-a, has-a, 主語・述語・目的語の簡明さ
- 共通語彙の活用度
- Sparql の量・質・複雑さ・多様さ

AI 実習全体を振り返り、考察・分析・展望

- AI 実習（知識工学系）で得た知見を今後どのように活用するか
- 具体的な知識データの活用方法
- 機械学習、生成系 AI との効果的な連携方法

目次	説明
AI 実習（知識工学系）で得た知見を今後どのように活用するか	これから様々なハッカソンや開発に参加する予定です。その際に、この講義で学んだ内容を実際の開発に活かし、オントロジーを利用することで効果的に取り組みたいと思います。
具体的な知識データの活用方法	例えば、最終レポートでは、自分は実際にただオントロジーまでじゃなくて、実際に目に見やすくフロントを開発しました。それは機械の検索するサービスです。他の案は、実際に自分の AI を作ることや多言語の辞書を作ることなどを想定しています。
機械学習、生成系 AI との効果的な連携方法	実際にオントロジーと連携した後、自分の感想は「すごい、早い」というものでした。普通の API とは異なり、オントロジーと連携すると検索速度が非常に速いです。方法としては、Virtuoso のようなサービスを利用して、自分のオントロジーの API を取得し、連携しました。

第 13 回グループ課題の要約・整理

回数	グループ名	発表者	発表内容	発表への質疑・コメント
13	青雲の志	竹田勇斗	家電に関するオントロジを構築。扇風機等をクラスとして定義し、販売元 URL も含めた。時間の制約から高度なクエリ作成には至らなかったが、2つのインスタンスのデータプロパティと概要を出力するクエリを実行。次回授業の準備も兼ねて取り組んだ。	Q: グループ内での役割分担は？ A: クエリ設計、全体統括など、メンバー間で分担して作業を進めた。
13	破竹の勢	倉石大輝	Wikidata のプログラミング言語データに対し、Python を用いてクエリを実行。ランダムに 10 個のサブジェクトを抽出し、選択したサブジェクトの述語を表示するプログラムを作成。例として「Java」の所有者を述語として表示した。	Q: 「所有者」は権利保有者を指すのか？ A: サブジェクトがランダムに選ばれるため、結果の正確性は保証できない。事実確認は行っていない。
13	飛龍在天	大竹啓之	Wikidata のポケモンデータを利用し、世代別クエリを実行。地域ごとのポケモンタイプ表示を目指したが、第 9 世代ポケモンの表示にとどまった。今後、ラベルでの検索機能実装を計画。	Q: 述語での検索が課題だったか？ A: その通りです。
13	鵬程万里	小柳怜	小柳: 地域別の山を標高順に表示。小出: Wikidata のポケモンデータで連結表示を実施。特定ポケモンの詳細表示を試みた。安達: DBpedia を用いて「研音」所屬俳優の情報を生年別に表示。主要作品情報も Wikipedia から取得して表示。	Q: 最高峰は？ A: 東北の燧ヶ岳。Q: ポケモンタイプの表示方法は？ A: タイプラベルと連結表示を使用。Q: 「テレビドラマ」表示は必要？ A: Wikipedia 由来の不要な情報。今後除外予定。
13	金剛不壞	相場陸	プログラミング言語に関するクエリを実行。2000 年以降に誕生した静的型付け言語を抽出。個別言語の詳細クエリは未完成。	Q: 抽出された言語で使用経験はあるか？ A: 「Go」言語の使用経験あり。Q: クエリの工夫点は？ A: 不要な ID の表示を除外するよう調整した。
13	磐石之固	加藤颯士	ポケモンオントロジを拡張し、「特性」と「進化方法」を追加。「しんりょく」特性を持つポケモンや、「ほのおのいし」で進化するポケモンを抽出し、進化前後の関係を一覧表示するクエリを実装。	Q: 道具進化とレベルアップ進化の両方がある場合の表示は？ A: 進化方法は一つのみなので、複数表示の必要はない。

第 14 回グループ課題の要約・整理

回数	グループ名	発表者	発表内容	発表への質疑・コメント
----	-------	-----	------	-------------

回数	グループ名	発表者	発表内容	発表への質疑・コメント
14	青雲の志			
14	破竹の勢			
14	飛龍在天	大竹啓之	<p>この班の発表内容は主にスピーカーのオントロジー。メーカーや機械の情報をクラスとして設定していると報告しました。連携機能がついているなどを今後追加していくと考えていますが、どの層に向けての商品なのかを分けて制作していきます。SPARQL の検索はスピーカーを買いに来たらイヤホンやマイクが欲しくなることを想定して情報を追加したいと思っている。</p>	<p>お風呂で使えるスピーカーの選び方についての情報を検索できるようにしたいです。また、スマートスピーカーの消費電力についての情報も記載できると助かります。</p>
14	鵬程万里			
14	金剛不壞	Roger Marvin	<p>機器のオントロジーに関して、開志ラボにある機器を自動化された機械と手動の機械に分類し、HTML や JavaScript を使ってデータを検索できるようにします</p>	<p>Q: マニュアルと自動的で分けたと合ったが、人の手で動かすものと自動的に動かす物という解釈で合ってるか。A: 実際に手を動かして行うものなので、作業自体の方法によって区別しています。Q: どういうスパークルを考えたか。A: クラスを抽出して、それに属するインスタンスや、インスタンスから関連するクラスを調査するためのスパークルクエリを考えています。</p>

回数	グループ名	発表者	発表内容	発表への質疑・コメント
14	磐石之国	五十嵐 寛人, 池田 侑哉, 駒木根道元	五十嵐: イヤホンのオントロジー。クラス: 種類、特徴、機能。スパークルでの検索は Apple の製品を検索すると価格などの情報を入れている。サイズや色などの情報も追加したい。池田: カメラのプロトタイプに向けてのオントロジー。クラス: カメラの名前、カメラの情報（メーカー、値段、ボディ色、画素数、レンズマウントなど）を含む。駒木根: 冷蔵庫について。お金持ちで奥さんが料理好きな素人を想定。クラス: メーカー、機能、色、価格。スパークルの検索は価格を無視して行うことを想定している。	五十嵐: Q: プロだったらどのようにスパークル検索をするのか。A: 動物を撮る場合だったらプレ防止などが重要なので、その段階ごとに検索をすることを想定している。駒木根: Q: 価格以外に選ぶ要素はあるのか。A: 冷蔵庫の機能で選べるようにする。Is-a、has-a 関係をしっかり記述する。

第 15 回グループ課題の要約・整理

回数	グループ名	発表者	発表内容	発表への質疑・コメント
15	青雲の志	竹田 勇斗	モニターと周辺機器のオントロジを作成。現時点ではモニター関連のみ実装。データプロパティを充実させた点が特徴。	Q: 複合検索は可能か？ A: はい、複数条件での検索を実装済み。プロパティの充実は良いアプローチだと評価された。
15	破竹の勢	倉石 大暉	キーボードに関するオントロジを構築。配列、スイッチタイプ、規格の違いなど、多様なクラスを設定。キータイプ検索機能を実装。	Q: キーキャップの位置による重さの違いは記述されているか？ A: 未実装だが、今後データプロパティを追加して対応予定。Q: 足用キーボードは考慮しているか？ A: 現時点では想定外。
15	飛龍在天	大竹 啓之, 阿部 一成, 川崎 宝	スピーカー・イヤホン、スマートフォンに関する複数のオントロジを発表。ブランドごとのインスタンス設定や、スマホ初心者向けの情報構築を目指す。	Q: スマホオントロジの重複をどう扱うか？ A: 検索対象ユーザーを分けて作成する方針。内蔵メモリやCPU情報の追加、ターゲットとオントロジの整合性向上が課題として指摘された。

回数	グループ名	発表者	発表内容	発表への質疑・コメント
15	鵬程万里	安達萌衣	コスメ、特にアイシャドウに特化したオントロジを作成。肌の色に合わせた製品検索を目指す。形状ごとのクラス分けを実施。	Q: 肌の種類は4種類で十分か？ A: 暖色系・寒色系それぞれ2種類を想定。型番でのグルーピングによる情報の簡潔化が提案された。
15	金剛不壊	ロジヤー	開志ラボ用の検索エンジン「Nier」を開発。施設管理者向けに、機器の購入・利用情報を提供。自作オントロジとAPIを活用。	Q: ハンダゴテの使用可否情報は？ A: 現在は管理者向け購入情報が中心。Q: 新規購入希望商品の扱いは？ A: 購入・カテゴリボタンの追加を検討中。詳細な条件設定の難しさが課題として挙げられた。
15	加藤磐石之固	颯士五十嵐寛人	たこ焼き機とイヤホンのオントロジをそれぞれ発表。商品名、メーカー、特徴などを詳細に記述。検索機能の実装も進めている。	たこ焼き機：Q: 想定ユーザーと検索方法は？ A: 長所、サイズ、価格などで検索可能。ユーザーニーズの明確化が課題。イヤホン：価格でのフィルタリングが評価された。より複雑なクエリの実装が期待される。

[自己成長、成果、上位成績に向けて] 個人成果の報告

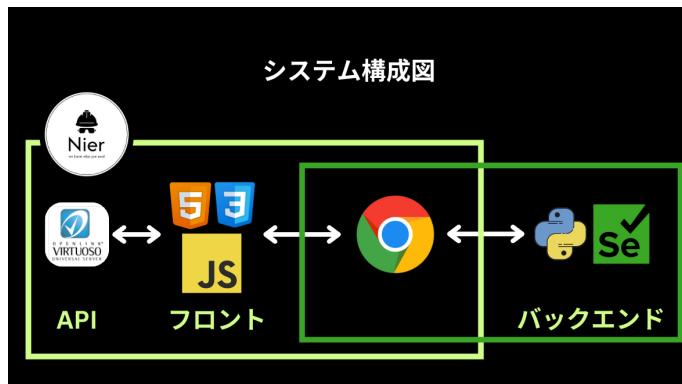
最終レポートに向けて、自分が機械の検索オントロジーを作りました。自分がただオントロジーを作るだけではなくて、実際にそのオントロジーと自分が開発するものをけいしたいと思うから、チャレンジしました。実際にできるかどうか全く知らなくて、とりあえず挑戦して、その結果、うまくできました。以下ではそのサービスの説明です。

サービスの詳細

サービス概要

目次	説明
テーマ	開志ラボ機械検索サービス
選んだ理由	開志ラボである機械を自分が書いたい時や開志ラボのような施設の管理者や利用者が機械の情報を調べるときに、一つ一つ探さなくても、このサービスを利用して、加速で機械の重要な情報を調べられるようにしたいと思ったから。
想定するユーザー	機械が好きな人、開志のような施設の利用者または管理者、機械に関して詳しくない人でも使います。

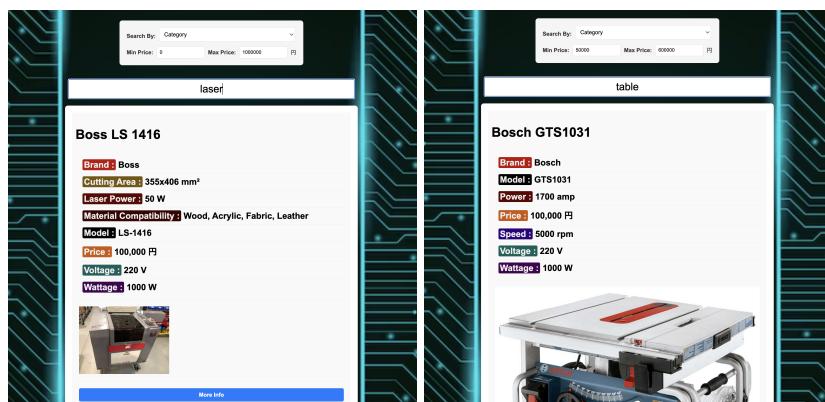
サービス説明



商品名で検索してみる



商品のカテゴリで検索してみる



ソースコード

HTML&CSS 「シンプルな HTML」

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>開志ラボ機械ガイド</title>
    <link rel="stylesheet" href="styles.css" />
    <script src="script.js" defer></script>
  </head>
  <body>

```

```

<div class="logo"></div>
<h1>検索</h1>
<div class="filter-container">
  <div class="search-by">
    <p>Search By:</p>
    <select id="searchType">
      <option value="productName">Product Name</option>
      <option value="category">Category</option>
    </select>
  </div>
  <div class="price-range">
    <label for="minPrice">Min Price:</label>
    <input
      type="number"
      id="minPrice"
      min="0"
      max="1000000"
      step="1000"
      value="0"
    />
    <label for="maxPrice">Max Price:</label>
    <input
      type="number"
      id="maxPrice"
      min="0"
      max="1000000"
      step="1000"
      value="1000000"
    />
    <span>円</span>
  </div>
</div>
<input
  type="text"
  id="individualInput"
  placeholder="Enter product name or category"
/>
<div id="results">Nothing here...</div>
</body>
</html>

```

JavaScript 「オントロジーの API と連携して、SPARQL クエリでやり取りする」

```

async function querySPARQL(searchInput, searchType, minPrice, maxPrice) {
  let query;
  if (searchType === "productName") {
    query = `
      PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
      PREFIX owl: <http://www.w3.org/2002/07/owl#>
      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
      SELECT ?product WHERE {
        ?product rdfs:label ?label .
        FILTER REGEX(?label, ${searchInput})
      }
    `;
  }
  const response = await fetch("http://ontology-api.com/query", {
    method: "POST",
    headers: {
      "Content-Type": "application/sparql-query"
    },
    body: query
  });
  const results = await response.json();
  return results;
}

```

```

PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-
15.owl#>

SELECT DISTINCT ?individual ?property ?value
WHERE {
    ?individual owl1:hasBrand ?brand .
    ?individual ?property ?value .

    BIND(STRAFTER(STR(?individual), "#") AS ?individualName)
    FILTER(REGEX(?individualName, "${searchInput}", "i"))
    FILTER(?property != rdf:type)

    ?individual owl1:hasPrice ?price .
    BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?
numericPrice)
    FILTER(?numericPrice >= ${minPrice} && ?numericPrice <=
${maxPrice})

    OPTIONAL {
        ?brand rdfs:label ?brandName .
    }
}
ORDER BY ?individual ?property
`;
}

else if (searchType === "category") {
query =
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl1: <file:/Users/rogermarvin/Desktop/Coding/AI/14-
15.owl#>

SELECT DISTINCT ?individual ?property ?value
WHERE {
    ?individual rdf:type ?category .
    ?category rdfs:label ?categoryLabel .
    FILTER(CONTAINS(LCASE(STR(?categoryLabel)),
LCASE("${searchInput}")))

    ?individual owl1:hasBrand ?brand .
    ?individual ?property ?value .
    FILTER(?property != rdf:type)

    ?individual owl1:hasPrice ?price .
    BIND(xsd:integer(REPLACE(?price, "[^0-9]", "")) AS ?
numericPrice)
    FILTER(?numericPrice >= ${minPrice} && ?numericPrice <=
${maxPrice})

    OPTIONAL {
        ?brand rdfs:label ?brandName .
    }
}
}

```

```
        ORDER BY ?individual ?property
    `;
}

console.log("SPARQL Query:", query);

const encodedQuery = encodeURIComponent(query);
const targetUrl = `http://localhost:8890/sparql?
query=${encodedQuery}&format=json`;

try {
    const response = await fetch(targetUrl, {
        headers: {
            Accept: "application/sparql-results+json",
        },
    });

    if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    displayResults(data);
} catch (error) {
    console.error("Error querying SPARQL endpoint:", error);
    const resultsDiv = document.getElementById("results");
    resultsDiv.innerHTML = `Error: ${error.message}`;
}
}

function displayResults(data) {
    const resultsDiv = document.getElementById("results");
    resultsDiv.innerHTML = "";

    const propertyLabels = {
        hasBrand: "Brand",
        hasBuildVolume: "Build Volume",
        hasLayerResolution: "Layer Resolution",
        hasMaterialCompatibility: "Material Compatibility",
        hasModel: "Model",
        hasPrice: "Price",
        hasTechnology: "Technology",
        hasWeight: "Weight",
        hasCuttingArea: "Cutting Area",
        hasLaserPower: "Laser Power",
        hasPrintArea: "Print Area",
        hasPrintResolution: "Print Resolution",
        hasSpindlePower: "Spindle Power",
        hasWorkArea: "Work Area",
        hasPower: "Power",
        hasSpeed: "Speed",
        hasImage: "Image",
        hasLink: "Link",
        hasVoltage: "Voltage",
    };
}
```

```
    hasWattage: "Wattage",
};

const propertyClasses = {
  hasBrand: "label-bg-brand",
  hasPrice: "label-bg-price",
  hasWeight: "label-bg-weight",
  hasBuildVolume: "label-bg-build-volume",
  hasLayerResolution: "label-bg-layer-resolution",
  hasMaterialCompatibility: "label-bg-material-compatibility",
  hasModel: "label-bg-model",
  hasTechnology: "label-bg-technology",
  hasCuttingArea: "label-bg-cutting-area",
  hasLaserPower: "label-bg-laser-power",
  hasPrintArea: "label-bg-print-area",
  hasPrintResolution: "label-bg-print-resolution",
  hasSpindlePower: "label-bg-spindle-power",
  hasWorkArea: "label-bg-work-area",
  hasPower: "label-bg-power",
  hasSpeed: "label-bg-speed",
  hasVoltage: "label-bg-voltage",
  hasWattage: "label-bg-wattage",
};

const individuals = {};

data.results.bindings.forEach((row) => {
  const individualFullUri = row.individual.value;
  const individual = individualFullUri.split("#").pop(); // Extract the name after the #
  const property = row.property.value.split("#").pop();
  let value = row.value.value;
  const label = propertyLabels[property] || property;

  if (!individuals[individual]) {
    individuals[individual] = {};
  }

  if (!individuals[individual][property]) {
    if (property === "hasImage" || property === "hasLink") {
      try {
        new URL(value);
        individuals[individual][property] = { property, label, value };
      } catch (e) {}
    } else {
      // Format specific properties
      if (property === "hasPrice") {
        value = `${value} 円`;
      } else if (property === "hasWeight") {
        value = `${value} kg`;
      } else if (property === "hasBuildVolume") {
        value = `${value} mm³`;
      } else if (property === "hasPower") {
        value = `${value} amp`;
      }
    }
  }
});
```

```

    } else if (property === "hasLayerResolution") {
      value = `${value} µm`;
    } else if (property === "hasWorkArea") {
      value = `${value} mm²`;
    } else if (property === "hasSpindlePower") {
      value = `${value} W`;
    } else if (property === "hasLaserPower") {
      value = `${value} W`;
    } else if (property === "hasCuttingArea") {
      value = `${value} mm²`;
    } else if (property === "hasVoltage") {
      value = `${value} V`;
    } else if (property === "hasWattage") {
      value = `${value} W`;
    } else if (property === "hasPrintResolution") {
      value = `${value} µm`;
    } else if (property === "hasSpeed") {
      value = `${value} rpm`;
    } else if (property === "hasBrand") {
      value = row.brandName ? row.brandName.value :
    value.split("#").pop();
    }
    individuals[individual][property] = { property, label, value };
  }
}
});

Object.keys(individuals).forEach((individualFullUri) => {
  const individual = individualFullUri.split("#").pop(); // Extract the
name after the #
  const individualDiv = document.createElement("div");
  individualDiv.className = "individual";

  // Add a title with the individual name
  const titleDiv = document.createElement("h2");
  titleDiv.textContent = individual.replace(/_/g, " "); // Replace
underscores with spaces
  individualDiv.appendChild(titleDiv);

  const descriptionDiv = document.createElement("div");
  const imageDiv = document.createElement("div");
  const linkDiv = document.createElement("div");

  Object.values(individuals[individualFullUri]).forEach(
    ({ property, label, value }) => {
      const resultItem = document.createElement("div");

      if (property === "hasImage") {
        const img = document.createElement("img");
        img.src = value;
        img.alt = label;
        img.className = "result-image";
        imageDiv.appendChild(img);
      } else if (property === "hasLink") {
        const link = document.createElement("a");
        link.href = value;
        link.textContent = label;
        linkDiv.appendChild(link);
      } else {
        const p = document.createElement("p");
        p.textContent = `${label}: ${value}`;
        descriptionDiv.appendChild(p);
      }
      resultItem.appendChild(descriptionDiv);
      resultItem.appendChild(imageDiv);
      resultItem.appendChild(linkDiv);
      individualDiv.appendChild(resultItem);
    }
  );
}
);

```

```
        const link = document.createElement("a");
        link.href = value;
        link.textContent = "More Info";
        link.className = "result-link";
        link.target = "_blank";
        linkDiv.appendChild(link);
    } else {
        const labelSpan = document.createElement("span");
        labelSpan.className = propertyClasses[property] || "label-bg";
        labelSpan.textContent = `${label}:`;

        resultItem.appendChild(labelSpan);
        resultItem.append(` ${value}`);
        descriptionDiv.appendChild(resultItem);
    }
}

individualDiv.appendChild(descriptionDiv);
individualDiv.appendChild(imageDiv);
individualDiv.appendChild(linkDiv);
resultsDiv.appendChild(individualDiv);
});

}

function handleInput() {
    const searchInput = document
        .getElementById("individualInput")
        .value.replace(/\ /g, "_");
    const searchType = document.getElementById("searchType").value;
    const minPrice = parseInt(document.getElementById("minPrice").value);
    const maxPrice = parseInt(document.getElementById("maxPrice").value);
    querySPARQL(searchInput, searchType, minPrice, maxPrice);
}

document.addEventListener("DOMContentLoaded", () => {
    document
        .getElementById("individualInput")
        .addEventListener("input", handleInput);
    document.getElementById("searchType").addEventListener("change",
    handleInput);
    document.getElementById("minPrice").addEventListener("input",
    handleInput);
    document.getElementById("maxPrice").addEventListener("input",
    handleInput);

    // Initial search
    handleInput();
});
```

```
import os
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

PATH = "./cd/chromedriver"
service = Service(PATH)

chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("--disable-web-security")
chrome_options.add_argument("--disable-site-isolation-trials")
chrome_options.add_argument("--start-fullscreen")

driver = webdriver.Chrome(service=service, options=chrome_options)
driver.get("file:///Users/rogermarvin/Desktop/Coding/AI/AI%E6%9C%80%E7%B5%82%E8%AA%B2%E9%A1%8C/index.html")

input("Press Enter to close the browser...")
driver.quit()
```

注意事項

- 直近 3 回分の個人演習と毎回のグループ課題の実施結果について、
- 学生ごとの個人のレポートとする
- 他の学生のレポートをコピー&ペーストしたことが発覚した場合、**不正行為とみなし、規程に基づく懲罰適用の可能性があるので絶対にやってはいけない**