INIAD CS Essentials 2

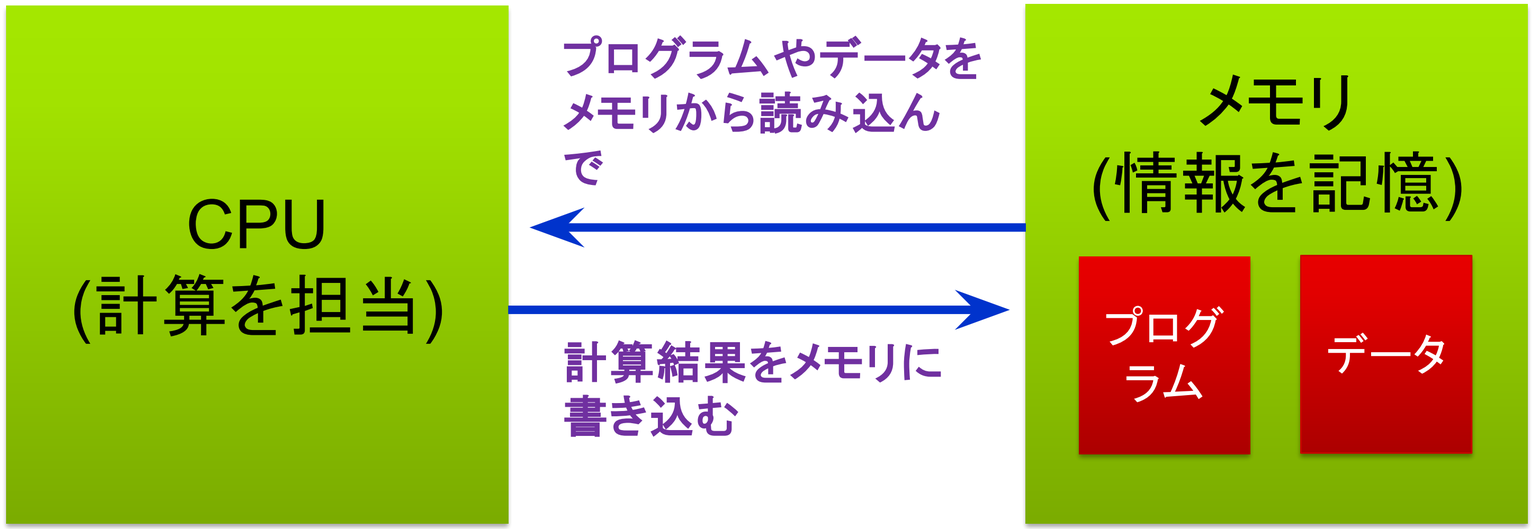
4-1：リストの仕組み

① メモリの補足

リストの学習を始める前に、コンピュータにおけるメモリについて、少しだけ補足をします。

（復習）ノイマン型コンピュータの仕組み

- コンピュータでは「メモリ」に情報を記憶
  - **CPU**は計算を担当する部品
  - CPUの実行機能だけではコンピュータは動作しないので、**メモリ(memory)** と呼ばれる場所にプログラムやデータを記憶させ、内容を見たり書き換えたりしながら動作



（復習） CPUとメモリ

- CPU (Central Processing Unit)：中央処理装置）
  - プログラムやデータをメモリから読み込み、プログラムに書かれている通りにデータ処理し、その結果をメモリに書き込む

インテルのCPU

- メモリ：プログラムやデータを記憶する装置
  - ROM (Read Only Memory)
    - プログラム自体など、プログラム実行後も消さずに保存するデータを記憶するメモリ
  - RAM (Random Access Memory）
    - プログラムの変数データなど、プログラムの実行中に読み書きするデータを記憶するメモリ

メモリとメモリアドレス

- メモリにはデータを保存することができますが、どこにどのデータを格納したかを区別する必要があります
- そこで、メモリには番地（アドレス）がついています
  - 「メモリアドレス」と言います
  - メモリアドレスは、正の整数が割り当てられています
- メモリ（RAM）では、指定したアドレスのデータを、瞬時に読み書きすることができます
  - 「ランダムアクセス」という名称の由来です
  - 実際には、キャッシュなどの機構があるため常に一定というわけではありませんが、計算時間を見積もる際には、通常メモリへのアクセスは一定時間 O(1) で完了するとみなします

（参考）物理アドレスと論理アドレス

- 実際には、現在のコンピュータでは「仮想記憶」という仕組みがあり、プログラムから見える「アドレス」と、メモリにとっての「アドレス」は異なっています
  - 物理アドレス：実際のハードウェアとしてのメモリ上のアドレスです
  - 論理アドレス：プログラム内で見えるメモリ上のアドレスです
- プログラムが「論理アドレス」にアクセスしようとすると、実際にはそのアドレスに対応する「物理アドレス」にアクセスするように、OSなどが変換をしています
  - このことで、プログラムが他のプログラムのデータを不用意に書き換えないようになっています
  - ※それ以外にも様々な役割がありますが、今回は触れません

Pythonのオブジェクトとメモリ

- コンピュータ・サイエンス概論Iで学習したように、Pythonでは全てのデータは「オブジェクト」です
- Pythonのオブジェクトも、実際にはメモリ上に格納されるため、アドレスをもちます
  - 標準的なPythonの実装では、オブジェクトのアドレスは `id()` メソッドで確認することができます

```
lst1 = [1,2,3]
lst2 = [1,2,3]
print(id(lst1))
print(id(lst2))
```

異なるオブジェクトなので、別のアドレスが表示されるはずです

```
lst1 = [1,2,3]
lst2 = lst1
print(id(lst1))
print(id(lst2))
```

同じオブジェクトなので、同じアドレスが表示されるはずです

データ構造とメモリ

- データをどのようにコンピュータ上で表すか（＝データ構造）は、アルゴリズムと並ぶ重要なトピックです
  - ※「アルゴリズム」と「データ構造」をセットで学習するのが一般的です
- 何故かというと、データをどのようにコンピュータ上に格納するかにより、処理によって効率に大きな違いが出るためです
  - 今回学習するように、リストを1つとっても、様々な実現方法があります
- 普段メモリアドレスを直接的に意識することはあまりないと思いますが、Pythonにおいてもオブジェクトはメモリ上に格納されていることを理解した上で、この先の学習を進めてください