見よう見まねで学ぶプログラミング

Pythonで学ぶプログラミングのきほん

株式会社インパラ　森近　真（2021/10/17）

自然言語と人工言語

プログラミング言語はコンピューターに仕事を依頼するための人工言語

なるべく人が理解しやすいように設計されている（はず）

一を聞いて十を知ることができるのが理想

規則性を見つけることで全てを覚えなくても予測できるようになる

講座の目標

プログラミングでの
データの扱い方を学ぶ

プログラミングで扱うデータとは

- プログラムとはデータを受け取って加工してデータを出力するのもの
- データが「食材」とするとプログラミングは「調理方法」
- 扱うデータが多いほどコンピュータが役に立つ
- データの種類や扱い方はプログラミング言語の中である程度共通性がある
- 講座では５種類のデータとその扱い方を学んでもらう

プログラミングを身につけるには

楽しむことが最も大事

プログラミングを楽しむには

- 自分が思いついたものを作ってみる
- 問題を自分でアレンジしてみる
- 身の回りや自分が興味のある分野で応用できないかを考える

講座の進め方

見よう見まねとは

算数の九九のように9 x 9の課題を用意

まず課題に挑戦してみる

問題文を実行して何が起きているのか観察する

どうやって動いているのかを推測する

課題の指示に従ってコードを変更したり、質問に回答する

さらに自分なりに問題をアレンジしてみる

考えただけではわからない場合

わからない言葉をGoogleなどで調べる

- 他のプログラミング 言語と区別するため「Python xxx」など

１行ずつ実行して状態がどう変化するのかを確かめる

- https://pythontutor.com/live.html#mode=edit

試験ではないのでカンニングして問題ない

何度か挑戦するうちにできるようになる（はず）

前回までのおさらい

[1の段] 文字列(string)

- プログラミングで文字列を扱うには「’」もしくは「”」で囲む
- 「**+**」で文字列同士を結合することができる
- 「**print()**」関数で文字列を表示できる
- 「**input()**」関数でキーボードから文字列を受け取れる
- 「**len()**」関数で文字列の長さを求めることができる
- 「**[n]**」に数字を入れることでその順番に当たる文字を取り出せる

[1の段] 変数(variable)

- データを入れておける箱
- データに名前をつけることができるラベル
- 自分で自由に名前をつけることができる
- 「=」で定義（代入）できる
- 中に入っているデータに対してできる操作は変数同士でも可能
- 「+=」で変数に値を追加したものを再定義できる

[2の段] 数値(number)

- 数値同士で四則演算や指数計算などができる
- 変数に代入すると変数同士で計算ができる
- 数値や文字列などデータごとに型（**class**）がある
- 「**type()**」関数で型を調べることができる
- 整数(**int**)と小数点(**float**)を含む数で型は異なるが計算は可能
- 異なる型を計算したり文字の連結をするには型を変換する必要がある

[3の段] 真偽値(boolean)

- 正しい(True)か正しくない(False)の二者一択
- 数字や文字などを比較演算子で比較した結果となる

[3の段] 比較演算子

- 「a」と「b」が等しい(**a == b**)、等しくない(**a != b**)
- 「a」より「b」が大きい(**a > b**)、小さい(**a < b**)
- 「a」が「b」以上(**a >= b**)、以下(**a <= b**)
- 「a」が「b」中に含まれている(**a in b**)、含まれていない(**a not in b**)

[3の段] 論理演算子

- 真偽値を組み合わせて評価するのに使われる演算子
- 「a」と「b」がどちらも正しい(**a and b**)
- 「a」と「b」の少なくともどちらかが正しい(**a or b**)
- 「a」の否定(**not a**)

[3の段] If文(条件文)

- 条件式（a > bなど）を評価した結果によって処理を分岐する
- 「**if ~:**」もし～の場合は（最初は必ずifで始まる）
- 「**elif ~:**」（そうでなくて）もし～の場合は（複数回使用できる）
- 「**else:**」それ以外の場合は（使用できるのは最後のみ、条件式は書かない）

[3の段] ブロックとインデント

- 一連の処理として実行されるまとまりを「（コード）**ブロック**」と呼ぶ
- Pythonでは「**インデント**（字下げ）」が重要な役割を持つ
- インデントの位置（高さ）が同じものがブロックとなる
- ブロックの前の行（ヘッダ）には最後に必ず「**:**（セミコロン）」が付く
- ブロックは入れ子にすることができる
- ブロック内で何も処理をしない場合は「**pass**」を書いておく

[4の段] While文(繰り返し)

- 「**while**」の後に続く条件が「**True**」である限り「**ブロック**」を繰り返す
- 「**break文**」に達すると繰り返しが終了する
- 「**while文**」の条件を「**False**」にすることでも繰り返しが終了する
- 「**while文**」の条件が「**True**」のままだと「**無限ループ**」になる
- 「**無限ループ**」になった場合は「**Ctrl + C**」でプログラムを終了させる

[4の段] インポート(import)

- 標準では用意されていない機能を輸入（import）するための仕組み
- 標準の状態はなるべく身軽にするために基本的な機能だけを準備している
- 「**import xxx**」とするとxxxの機能が使えるようになる
- 時間やランダムな値などは**標準ライブラリ**に含まれ、すぐにimportできる
- さらに専門的な機能は「**pip install xxx**」でインストールする必要がある

[5の段] リスト(list)

- 「**[]**」で囲み「**,**(カンマ)」で区切った複数のデータ(要素)の集まり
- 文字列、数値、真偽値など様々なデータを「要素」として持てる
- 「**[n]**」に数字(**index**)を入れることでその順番に当たる要素を取り出せる
- 「**[n]=**」でその順番の要素を変更できる
- 「**len()**」関数で要素の数を求めることができる
- データがリスト内に含まれているかどうかは「x **in** list」で判定可能

[5の段] for文(繰り返し)

- リストなどのデータを処理するのに便利な構文
- 「**for** a **in** b:」でリスト「b」の各要素を変数「a」として取り出せる
- 各要素が取り出し終えた時点でループは終了する
- While文と同様に「**break文**」を使用することができる

[6の段] リストの活用

- リストの中にリストがあるものを「**多重(多次元)リスト**」と呼ぶ
- 「**list[n][m][…]**」のように添字を並べることで値を取り出す
- 「**for文**」や「**while文**」で取り出す場合は入れ子にする
- 「**.append()**、**.remove()**、**.insert()**」で要素の追加や削除ができる
- 「**max(list)**」で最大値、「**min(list)**」で最大値が取り出せる
- 「**.sort()**」「**.reverse()**」などの関数で並び替えができる

[7の段] 辞書(dict)

- リスト(list)と同様に複数のデータをひとまとまりに扱うためのデータ型
- 「**キー(key)**」と「**値(value)**」の組み合わせを「**波括弧({})**」で囲む
- リストが添字に数字を使うのに対しキーには文字列を使う場合が多い
- リストと同じように辞書やリストのデータを入れ子にできる

『12月12日10時～11時30分』miraiデジタルカレッジ
【見よう見まねで学ぶプログラミング（Python）⑩の段】
アンケートQRコード

https://forms.gle/8eGVymnfrCGinV7T6