【情報】 概論

基礎

■プログラミング

- ▶ トークン
- ▶ エスケープ文字
- ▶ スコープ
- ▶ 丸め誤差
- ▶ シャドーイング
- ▶ 大域脱出
- ▶ 同期・非同期
- ▶ 糖衣構文
- ▶ リファクタリング
- ▶ ボイラープレートコード
- ▶ TODOコメント
- ▶ オーダ
- ▶ バイナリファイル
- ▶ プロセッサ
- ▶ マシン語
- ▶ アセンブリ言語
- ▶ 高級言語
- ▶ コンパイル言語
- ▶ スクリプト言語
- ▶ コンパイラ
- ▶ アセンブラ

【情報】 概論

基礎

■プログラミング

- ▶ トークン　ソースコード上の文字列の最小単位
- ▶ エスケープ文字　それに続く文字について別の解釈をすることを示す文字　※ `\` 等
- ▶ スコープ　変数の有効範囲
- ▶ 丸め誤差　ある桁以降の値を捨ててしまうことにより生じる誤差
- ▶ シャドーイング　既存のものと同名の変数や関数を定義して、そのスコープで既存の変数や関数にアクセスできなくする機能
- ▶ 大域脱出　幾重の反復や関数呼び出しを全て終了して所定の場所へ飛ぶ
- ▶ 同期・非同期　実行結果を待つかどうか
- ▶ 糖衣構文　冗長なコードに代わる簡潔な書き方
- ▶ リファクタリング　プログラムの外部から見た動作を変えずにソースコードの内部構造を整理すること
- ▶ ボイラープレートコード　ほとんどまたは全く変化することなく複数の場所で繰り返される定型コードのセクション
- ▶ TODOコメント　あとでやりたい、やるべき作業に備忘録としてつけるコメント
- ▶ オーダ　入力データの大きさにたいするアルゴリズムの実行時間の増加量
- ▶ バイナリファイル　特定の文字コードの範囲に収まらない任意のビット列を含むバイナリデータを保存したファイル。テキストファイル以外のファイルはすべてバイナリファイル。
- ▶ プロセッサ　CPUのこと
- ▶ マシン語　プロセッサが直接実行できる、0 と 1 の並びで書かれた言語
- ▶ アセンブリ言語　人間にわかりやすくした言語。マシン語と1対1対応している
- ▶ 高級言語　人間にさらにわかりやすくした言語。読み書きもしやすい
- ▶ コンパイル言語　ソースコードを事前にコンパイルして、先にマシン語に変換しておく必要がある高級言語
- ▶ スクリプト言語　ソースコードを逐次マシン語に翻訳しつつ実行される高級言語
- ▶ コンパイラ　高級言語－(一気に)→アセンブリ言語やマシン語　にするソフト
- ▶ アセンブラ　アセンブリ言語→マシン語　にするソフト

- ▶ インタプリタ
- ▶ コマンドライン
- ▶ シェル
- ▶ ターミナル
- ▶ コンソール
- ▶ 統合開発環境 (IDE)
- ▶ ORM
- ▶ フレームワーク
- ▶ DSL
  (ドメイン固有言語)
- ▶ ※ メモ用のコメントとして `# notes:` は便利そう。

■型システム

- ▶ 静的型付け
  動的型付け
- ▶ 強い型付け
  弱い型付け
- ▶ 明示的型変換
  暗黙的型変換
- • 名前的型システム
  構造的型システム
- ▶ ダックタイピング
- ▶ ポリモーフィズム

■コーディング（コードを書く行為）

- ▶ メタ構文変数
- ▶ メソッドチェーン
- ▶ モンキーパッチ
- ▶ オーバーライド
- ▶ ※ アーリーリターンでコードをスッキリさせようとするも、結局どの分岐にも当てはまらないためにスルーさせてしまうという間違いを犯すことがあるので注意。

■命名規則

- ▶ キャメルケース、ローワーキャメルケース

- ▶ インタプリタ　高級言語－(1行ずつ)→マシン語　にするソフト
- ▶ コマンドライン　文字でコマンドを打ち込んで操作するインターフェイス
- ▶ シェル　人間がOSを操作するためのインターフェイス
- ▶ ターミナル　GUIの上でCUIの操作をしたいときに使用するアプリ
- ▶ コンソール　コマンド入力を受け付ける（一般には）黒い画面。
- ▶ 統合開発環境 (IDE)　エディタ・インタプリタ・デバッガなどを１画面でできるソフト
- ▶ ORM　RDBに対するデータの操作をオブ指向で扱えるようにする手法
- ▶ フレームワーク　Webアプリ・システム開発に必要な機能が予め用意された枠組み
- ▶ DSL
  (ドメイン固有言語)　何か特別な目的を実現するために定義された、人間にとっても機械にとっても読みやすいテキストファイルの記述ルール
- ▶ ※ メモ用のコメントとして `# notes:` は便利そう。

■型システム

- ▶ 静的型付け　値やオブの型安全性を、コンパイル時に検証する
  動的型付け　〃　実行時に　〃
- ▶ 強い型付け　データ値の型を、プログラミング言語仕様内で解釈する
  弱い型付け　言語仕様外の明示的型変換と型キャストを用いて様々に解釈できる
- ▶ 明示的型変換　変数を宣言する際、その型を人間が明示的に識別する必要がある機構
  暗黙的型変換　周辺情報および文脈などから自動的に各々の型を決定する機構
- • 名前的型システム
  構造的型システム
- ▶ ダックタイピング　オブのクラスが何であれそのメソが呼べれば良しとするスタイル
- ▶ ポリモーフィズム　ある1つの関数の呼び出しに対し、オブ毎に異なる動作をする

■コーディング（コードを書く行為）

- ▶ メタ構文変数　hoge など
- ▶ メソッドチェーン　変数を介さず、関数の返り値に直接関数を呼び出すスタイル
- ▶ モンキーパッチ　既存のクラスの**メソッド**の振る舞いを変更すること
- ▶ オーバーライド　・継承した親クラスのメソッドを上書きする
  ・返り値の型や仮引数が異なる関数をつくる
- ▶ ※ アーリーリターンでコードをスッキリさせようとするも、結局どの分岐にも当てはまらないためにスルーさせてしまうという間違いを犯すことがあるので注意。

■命名規則

- ▶ キャメルケース、ローワーキャメルケース　hogeHogeHoge

- ▶ パスカルケース、アッパーキャメルケース
- ▶ スネークケース
- ▶ アッパースネークケース、コンスタントケース
- ▶ ケバブケース、チェインケース

■オブジェクト指向

- ▶ クラス
- ▶ オブジェクト（ｲﾝｽﾀﾝｽ）
- ▶ メソッド
- ▶ レシーバ
- ▶ メッセージ
- ▶ 状態（ｽﾃｰﾄ）
- ▶ 属性（ｱﾄﾘﾋﾞｭｰﾄ、ﾌﾟﾛﾊﾟﾃｨ）

- ▶ メンバ変数（ｲﾝｽﾀﾝｽ変数）
- ▶ ゲッターメソッド
- ▶ セッターメソッド
- ▶ アクセサメソッド

■UML

- ▶ ※ UML (Unified Modeling Language、統一モデリング言語) とは、システムの振る舞いや構造をオブジェクト指向で分析したり設計したりする際、図を用いることで視覚的に把握できるようになり、効果的に表現できる。しかし、その図の描き方が人によって違っては困るために、標準規格として作られた図の記法（モデリング言語）がUMLである。

構造図

- ▶ クラス図
- ▶ コンポジット構造図（複合構造図）
- ▶ コンポーネント図
- ▶ オブジェクト図
- ▶ 配置図
- ▶ パッケージ図

振舞い図

- ▶ パスカルケース、アッパーキャメルケース　HogeHogeHoge
- ▶ スネークケース　hoge_hoge_hoge
- ▶ アッパースネークケース、コンスタントケース　HOGE_HOGE_HOGE
- ▶ ケバブケース、チェインケース　hoge-hoge-hoge

■オブジェクト指向

- ▶ クラス　オブジェクトの設計図（ひな形）
- ▶ オブジェクト（ｲﾝｽﾀﾝｽ）　クラスをもとに作られたデータの集合体
- ▶ メソッド　オブジェクトがもつ動作や振る舞い
- ▶ レシーバ　とくにメソッドと比較したときのオブジェクトのこと
- ▶ メッセージ　とくにレシーバと比較したときのメソッドのこと
- ▶ 状態（ｽﾃｰﾄ）　オブジェクトごとに保持されるデータのこと
- ▶ 属性（ｱﾄﾘﾋﾞｭｰﾄ、ﾌﾟﾛﾊﾟﾃｨ）　オブジェクトから取得、あるいはそれに設定できる値

- ▶ メンバ変数（ｲﾝｽﾀﾝｽ変数）　個々のインスタンスごとに固有の変数
- ▶ ゲッターメソッド　メンバ変数を読み取るメソッド
- ▶ セッターメソッド　メンバ変数を書き換えるメソッド
- ▶ アクセサメソッド　ゲッターメソッド、セッターメソッドを併せてこう呼ぶ

■UML

- ▶ ※ UML (Unified Modeling Language、統一モデリング言語) とは、システムの振る舞いや構造をオブジェクト指向で分析したり設計したりする際、図を用いることで視覚的に把握できるようになり、効果的に表現できる。しかし、その図の描き方が人によって違っては困るために、標準規格として作られた図の記法（モデリング言語）がUMLである。

構造図

- ▶ クラス図　クラス間の関係を表現
- ▶ コンポジット構造図（複合構造図）　複数のクラスを包括するようなクラスやコンポーネントにおいて、その内部要素の構造や相互作用を表現
- ▶ コンポーネント図　コンポーネントの内部構造やそれら同士の相互作用を表現
- ▶ オブジェクト図　オブジェクト（インスタンス）間の関係を表現
- ▶ 配置図　システムの物理的側面を表現
- ▶ パッケージ図　モデリングされたクラスなどをグループ化し関連付け

振舞い図

- ▶ アクティビティ図
- ▶ ユースケース図
- ▶ ステートマシン図
  （状態遷移図）
- ▶ シーケンス図
- ▶ コミュニケーション図
- ▶ インタラクション
  オーバービュー図
  （相互作用概要図）
- ▶ タイミング図

■その他の設計図

- ▶ E-R図
- ▶ DFD (データフロー図)

■ソフトウェア設計

- ▶ MVC
- ▶ CRUD
- ▶ リグレッション

■通信

- ▶ ICT
- ▶ 輻輳(ふくそう)
- ▶ ☆ ポート番号
- ▶ ☆ アクセスポイント
- ▶ ☆ HTTPステータスコード

■Web

- ▶ 自分の PC の IPアドレスの調べ方
- ▶ ※ localhost のIPアドレスは `127.0.0.1` 。
- ▶ DNSとは
- ▶ ドメインに対応するIPアドレスを調べる
- ▶ ﾈｯﾄﾜｰｸ上のパケットの経路を調べる
- ▶ パケット・パケット通信とは

- ▶ アクティビティ図　ある振る舞いから次のふるまいへの制御の流れを表現
- ▶ ユースケース図　システムが外部に提供する機能と、
  それを利用する人や外部システムとの関係を表現
- ▶ ステートマシン図
  （状態遷移図）　1つのオブジェクトの状態がイベントの発生や
  時間の経過とともにどのように変化するかを表現
- ▶ シーケンス図　オブジェクト間の相互作用を時系列で表現
- ▶ コミュニケーション図　オブジェクト間の相互作用と応答を表現
- ▶ インタラクション
  オーバービュー図
  （相互作用概要図）　機能ごとに記述された相互作用図が、
  より広域のシステム構成から見たとき、
  それぞれがどのように連携しているのかを表現
- ▶ タイミング図　クラスやオブジェクトの状態を時系列で表現

■その他の設計図

- ▶ E-R図　データの関係性を実体、関連の2つの概念を用いて表現
- ▶ DFD (データフロー図)　データの流れを表現

■ソフトウェア設計

- ▶ MVC　機能を Model、View、Controller に分けて設計
- ▶ CRUD　永続的にデータを扱うソフトウェアに要求される4つの基本機能
- ▶ リグレッション　ある変更により混入された新たなバグ

■通信

- ▶ ICT　情報通信技術（Information and Communication Technology）
- ▶ 輻輳(ふくそう)　インターネット回線や電話回線にアクセスが集中すること
- ▶ ☆ ポート番号
- ▶ ☆ アクセスポイント
- ▶ ☆ HTTPステータスコード

■Web

- ▶ 自分の PC の IPアドレスの調べ方　ここにアクセス　※ `ipconfig` と結果違う…
- ▶ ※ localhost のIPアドレスは `127.0.0.1` 。
- ▶ DNSとは　IPアドレスとドメインを対応させるシステム
- ▶ ドメインに対応するIPアドレスを調べる　cmdにて `> nslookup ドメイン` と打つ
- ▶ ﾈｯﾄﾜｰｸ上のパケットの経路を調べる　cmdにて `> tracert ドメイン` と打つ
- ▶ パケット・パケット通信とは　通信上、分割されたデータの単位

- ▶ URL
- ▶ URN
- ▶ URI
- ▶ URLにおけるスキーム
- ▶ URLにおけるオーソリティ
- ▶ URLにおけるパス
- ▶ URLにおけるクエリ
- ▶ URLにおけるフラグメント
- ▶ ☆ サーバーにアップロードしたのにページが更新されない場合、閲覧用ブラウザにキャッシュが残っていることを疑おう。　→　スーパーリロード
- ▶ ルーティング
- ▶ 検索エンジン
- ▶ SEO

■文字コード

- ▶ ※ `Shift-JIS` と `CP932` (= `MS932` = `Windows-31J`) はとてもよく似ている。
- ▶ ☆ 改行コードについて

■正規表現

- ▶ ☆ 正規表現の規格（参考）
- ▶ ☆ マッチする文字列を視覚的に確認するためのサイト

メタ文字

- ▶ 任意の1文字
- ▶ 文字群のいずれか1文字
- ▶ 文字群のいずれでもない1文字
- ▶ 数字1文字・英字1文字
- ▶ 数字以外・英字以外の1文字
- ▶ 行の先頭
- ▶ 行の末尾
- ▶ 直前のパターンを0回か1回繰り返す
- ▶ 直前のパターンを丁度*n*回くり返す
- ▶ 直前のパターンを0回以上くり返す

- ▶ URL　リソースのネット上での所在を表す書式
- ▶ URN　リソースを**その存在位置によらずに**一意に識別するための符号を与える書式
- ▶ URI　URL と URN の総称
- ▶ URLにおけるスキーム　http://example.com/main/index?q=python#lead なら http
- ▶ URLにおけるオーソリティ　〃 なら example.com
- ▶ URLにおけるパス　〃 なら /main/index
- ▶ URLにおけるクエリ　〃 なら q=python
- ▶ URLにおけるフラグメント　〃 の lead
- ▶ ☆ サーバーにアップロードしたのにページが更新されない場合、閲覧用ブラウザにキャッシュが残っていることを疑おう。　→　スーパーリロード
- ▶ ルーティング　ネットワーク上でデータを転送する際、その経路を導き出すこと
- ▶ 検索エンジン　ネット上にある情報を検索する機能、およびそのプログラム
- ▶ SEO　検索エンジンの検索結果上位にサイトが表示されるための様々な工夫

■文字コード

- ▶ ※ `Shift-JIS` と `CP932` (= `MS932` = `Windows-31J`) はとてもよく似ている。
- ▶ ☆ 改行コードについて

■正規表現

- ▶ ☆ 正規表現の規格（参考）
- ▶ ☆ マッチする文字列を視覚的に確認するためのサイト

メタ文字

- ▶ 任意の1文字　(.|\n)　※ `.` だけだと `\n` が含まれない
- ▶ 文字群のいずれか1文字　[*characters*]　[*charA-charZ*]
- ▶ 文字群のいずれでもない1文字　[^*charcters*]
- ▶ 数字1文字・英字1文字　\d　・　\w
- ▶ 数字以外・英字以外の1文字　\D　・　\W
- ▶ 行の先頭　^
- ▶ 行の末尾　$
- ▶ 直前のパターンを0回か1回繰り返す　?
- ▶ 直前のパターンを丁度*n*回くり返す　{*n*}
- ▶ 直前のパターンを0回以上くり返す　*（欲張り）　*?（控えめ）

- ▶ 直前のパターンを1回以上くり返す
- ▶ 直前のパターンを*n*回以上くり返す
- ▶ 直前のパターンを*m*回以下くり返す
- ▶ 直前のパターンを*n*～*m*回くり返す
- ▶ 囲まれたパターンをグループ化
- ▶ 区切られたパターンのいずれか
- ▶ キャプチャで抜き出したい部分
- ▶ キャプチャを回避してグループ化
- ▶ 名前をつけてキャプチャ
- ▶ *next*が続いている*prev*
- ▶ *next*が続いていない*prev*
- ▶ *prev*が前にある*next*
- ▶ *prev*が前にない*next*
- ▶ メタ文字をエスケープ
- ▶ ☆ 主なメタ文字のまとめ（具体例つき）

■プログラム・ソフトウェア

- ▶ メジャー・バージョン番号
- ▶ マイナー・バージョン番号
- ▶ ミドルウェア
- ▶ ※ ミドルウェアには「Webサーバー」「アプリケーションサーバー (APサーバ) 」「データベース管理サーバー (DBサーバ) 」の3種類がある。これらはWeb3層構造と呼ばれ、セキュリティの高さや管理のしやすさ、故障の際の復旧が早いことから、大規模システムで広く使用されている。
- ▶ Webサーバ
- ▶ APサーバ
- ▶ DBサーバ
- ▶ ☆ インストーラの名前にある x86_64, amd64, i386, i686, x86, x64 の意味

文化など

■文化

- ▶ 直前のパターンを1回以上くり返す　+（欲張り）　+?（控えめ）
- ▶ 直前のパターンを*n*回以上くり返す　{*n*,}（欲張り）　{*n*,}?（控えめ）
- ▶ 直前のパターンを*m*回以下くり返す　{,*m*}（欲張り）　{,*m*}?（控えめ）
- ▶ 直前のパターンを*n*～*m*回くり返す　{*n*,*m*}（欲張り）　{*n*,*m*}?（控えめ）
- ▶ 囲まれたパターンをグループ化　(*pattern*)
- ▶ 区切られたパターンのいずれか　|
- ▶ キャプチャで抜き出したい部分　(*pattern*)
- ▶ キャプチャを回避してグループ化　(?:*pattern*)
- ▶ 名前をつけてキャプチャ　(?<*name*>*pattern*)
- ▶ *next*が続いている*prev*　*prev*(?=*next*)
- ▶ *next*が続いていない*prev*　*prev*(?!*next*)
- ▶ *prev*が前にある*next*　(?<=*prev*)*next*
- ▶ *prev*が前にない*next*　(?<!*prev*)*next*
- ▶ メタ文字をエスケープ　\　※エスケープしたい文字の直前におく
- ▶ ☆ 主なメタ文字のまとめ（具体例つき）

■プログラム・ソフトウェア

- ▶ メジャー・バージョン番号　Python 3.7.1 の 3 の部分
- ▶ マイナー・バージョン番号　Python 3.7.1 の 7.1 の部分
- ▶ ミドルウェア　OSとアプリケーションの間に入り、両者の役割を補佐するソフト
- ▶ ※ ミドルウェアには「Webサーバー」「アプリケーションサーバー (APサーバ) 」「データベース管理サーバー (DBサーバ) 」の3種類がある。これらはWeb3層構造と呼ばれ、セキュリティの高さや管理のしやすさ、故障の際の復旧が早いことから、大規模システムで広く使用されている。
- ▶ Webサーバ　ブラウザから検索した検索結果を視覚的に表示させる役割を担うサーバ
- ▶ APサーバ　動的コンテンツや業務処理を行うサーバ
- ▶ DBサーバ　DBMSが動作しているサーバ
- ▶ ☆ インストーラの名前にある x86_64, amd64, i386, i686, x86, x64 の意味

文化など

■文化

- ハッカソン
- イースターエッグ

- ハッカソン　　ソフト開発関係者らが集まって集中的にソフト開発作業を行うイベント
- イースターエッグ　　ソフトなどで密かに備えられている、
  本来の機能、目的とは無関係のメッセージや画面

特定のアプリケーションについて

■VSCode

- ☆ Python ライブラリについてサジェストさせる
- ☆ PCにインストールされているコマンドラインシェルを開く (に接続する？)
- ☆ Mac OSにて、コマンドラインから呼び出せるようにする

■PowerPoint

- ※ 動画をエクスポートする際は、 [画面切り替え] > [画面切り替えのタイミング] > [自動] にチェックがついており、ちゃんと時間が設定されていることを確認しよう。

特定のアプリケーションについて

■VSCode

- ☆ Python ライブラリについてサジェストさせる
- ☆ PCにインストールされているコマンドラインシェルを開く (に接続する？)
- ☆ Mac OSにて、コマンドラインから呼び出せるようにする

■PowerPoint

- ※ 動画をエクスポートする際は、 [画面切り替え] > [画面切り替えのタイミング] > [自動] にチェックがついており、ちゃんと時間が設定されていることを確認しよう。