第4章

情報システムとプログラミング

第4章 情報システムとプログラミング

本単元の学習内容 【学習内容の全体像】

4 情報システムとプログラミング

ア 情報システム全体の情報の流れ

- 1 情報システムにおける情報の流れや処理の仕組み
- 2 情報システムにおける情報セキュリティを確保する方法や技術
- 3 情報システムから提供されるサービスが生活に与える効果や影響，サービスが停止した時の影響

イ 情報システムの表し方，情報システムの分割と設計

- 1 情報システムの機能や性能を明確化する要件を定義し，これを表す方法を理解する
- 2 情報システムを機能要素ごとに分割し，その関係を定義した上で，分割されたシステムの設計
- 3 全体の進行を見据えたプロジェクト・マネジメントの手法

ウ 分割したシステムの制作・統合・テスト

- 1 分割したシステムの制作に適したプログラミング言語の選択とそれを利用した制作
- 2 分割して作成したプログラムの統合
- 3 分割したプログラムのテスト及び統合した後の情報システムの動作テスト

全 体

実際に稼働している情報システムを調査する活動や情報システムを設計し制作する活動を通して，情報の科学的な見方・考え方を働かせて，情報システムの仕組み，情報セキュリティを確保する方法，情報システムを設計しプログラミングする方法を理解し，必要な技能を身に付けるようにするとともに，情報システムの制作によって課題を解決したり新たな価値を創造したりする力を養う。

学習目標

- 情報システムにおける，情報の流れや処理の仕組み，情報セキュリティを確保する方法や技術について理解するとともに，それらによって提供されるサービスについて，その在り方や社会に果たす役割と及ぼす影響について考察する。
- 情報システムの設計を表記する方法，設計，実装，テスト，運用等のソフトウェア開発のプロセスとプロジェクト・マネジメントについて理解するとともに，情報システムをいくつかの機能単位に分割して制作し統合するなど，開発の効率や運用の利便性などに配慮して設計する。
- 情報システムを構成するプログラムを制作する方法について理解し技能を身に付けるとともに，これを制作し，その過程を評価し改善する。

本単元の取扱い

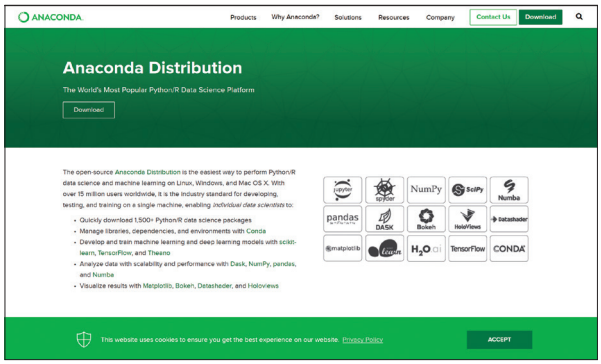
- 社会の中で実際に稼働している情報システムを取り上げ，それらの仕組みと関連させながら扱う。

情報Ⅰの学習内容との関連

- 情報システムとプログラミングについては，(3)「コンピュータとプログラミング」や，(4)「情報通信ネットワークとデータの活用」などの各項目と関連付けて扱う。

本章の学習での演習環境

- 本章の演習においては，プログラミング言語Pythonを想定している。Pythonに関しては，v3.7以上で，AnacondaのJupyter Notebook, Jupyter Labo, Spyderなどの統合開発環境での利用を想定している。
- 環境の構築については複数のライブラリやモジュールを利用し，システム構築を行うことから，オンライン実行環境と同様に，使用端末に直接Anaconda Distributionなどをインストールする方法が考えられる。また，Anaconda Distributionなどをインストールすることで，Python及び必要なライブラリをインストールすることができる。



図表1 「Anaconda Distribution」のインストール画面

URL：https://www.anaconda.com/distribution/

学習

19 情報システム全体の情報の流れ

研修内容

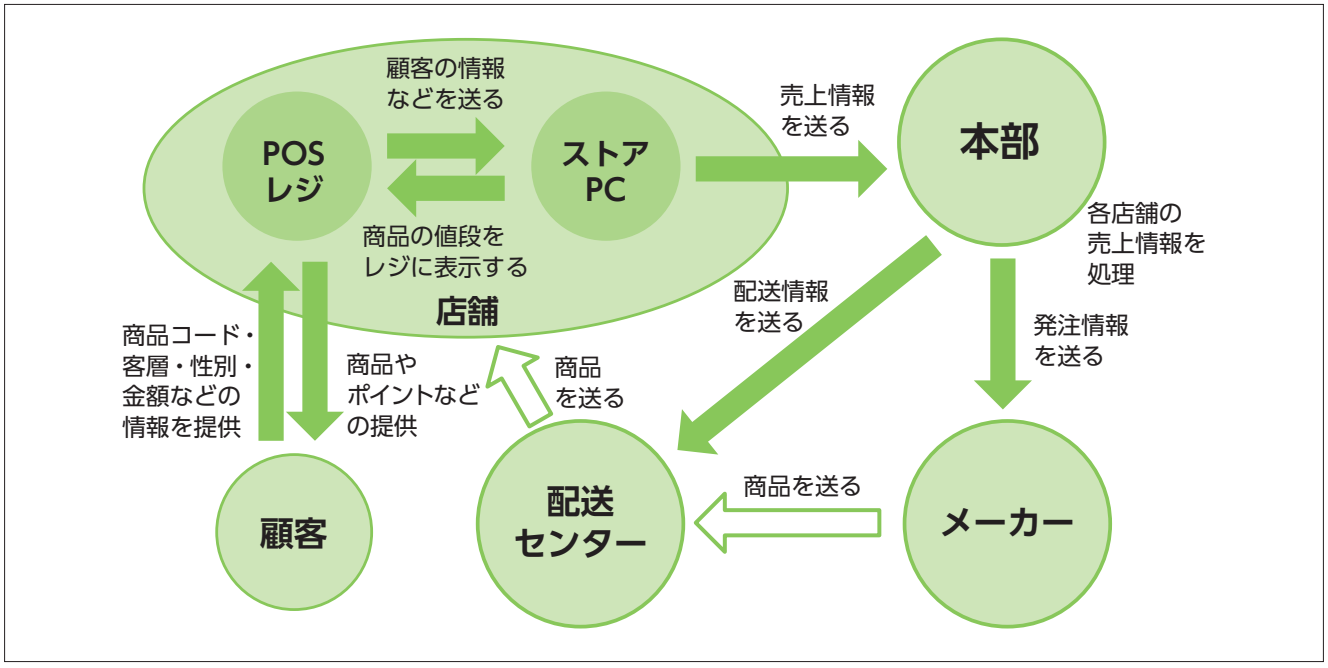
研修の目的

- 情報システムにおける，ユーザーが提供する情報と情報システムが提供する利便性を理解する。
- 情報システムにおける，情報の流れや処理の仕組みについて理解する。
- 情報システムによって提供されるサービスについて，その在り方や社会に果たす役割と及ぼす影響力について考察させる授業ができるようになる。

1 情報システムが集めるデータと生み出す利便性

情報システムとは情報通信ネットワークを活用し，様々なデータを収集・共有し，伝達することで運用されているシステムのことである。身近な情報システムとして，コンビニエンスストアなどで使用されているPOSシステム(Point Of Sales System)がある。POSシステムは，ユーザーが提供する情報を収集し，集めた情報を効果的に活用することで，商品が品切れすることなく，状況に応じて必要な数の商品が入荷され，効率よく営業できる利便性を提供することができる。

図表1にPOSシステムのデータの流れを図解化したものを示す。このように情報システムはユーザーからデータを収集し，集められたデータを効果的に処理し，そのデータを別のユーザーなどに伝達することで，図表2に示すような大きな利便性を生み出すことができる。



図表1　POSシステムのデータの流れを図解化したもの

ユーザーが提供する情報	情報システムが提供する利便性
・バーコードによる商品コード ・購入した時間 ・年齢層・性別 ・ポイントカードとのひも付け等	・在庫管理及び店舗管理の効率化 ・売り上げ分析 ・顧客分析 ・個人の購入履歴に応じたサービスの提供

図表 2　POS システムが集める情報と提供する情報

図表 3 に身近な情報システムが集める情報と提供する利便性の例を示す。情報通信ネットワークの発達により，今や私たちの生活において，情報システムは身近で利用することが当たり前になり，日常生活において必要不可欠な存在になった。そのため情報システムの利用者が増え，集められるデータの量が膨大となり，その情報をビッグデータとして活用することが近年注目されている。

その一方で，集められたビックデータをどう分析すればいいか分からない，あるいは集めたデータを適切に管理するための情報セキュリティを確保しなければならないといった課題を抱えるようになった。

情報システム	集める情報	提供する利便性
電子決済システム	購入した商品 店舗情報 購入日時 支払者の決済システムへの登録情報	支払いの簡略化
SNS	ユーザーから投稿された 文字・写真・動画	ユーザー間の交流の促進
ITS（Intelligent Transport Systems）	位置情報 目的地	最適な交通ルートの提供
e ラーニング	学習履歴 学習習熟度	次に学習する内容の提案

図表 3　情報システムが集める情報と提供する利便性の例

演習・1

EXERCISE

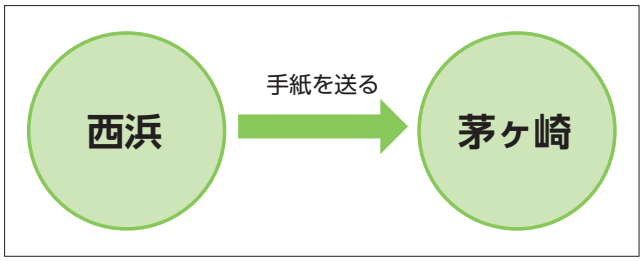
POS システムや図表 3 以外の身近な情報システムについて，情報システムにユーザーが提供する情報と，その情報システムが提供する利便性について考えてみましょう。

2 情報システムの情報の流れや処理の仕組み

このように，情報システムはユーザーから情報を集め，集めた情報を効果的に処理し，伝達することにより大きな利便性を生み出す。

こうした情報システムの情報の流れや処理の仕組みを表現する方法として，データフロー図，UML（199 ページ）といった表現方法がある。しかし，ここでは情報システムがどんなデータを集め，利便性があるかに注目するため，図解化という概念を用いて情報システムの情報の流れや処理の仕組みを表現してみよう。

図解化とは，概念を図形の組み合わせによって表すものである。例えば，「西浜は茅ヶ崎に手紙を送る」という概念を図解化して表すと図表 4 のようになる。



図表 4　図形と矢印によって概念を表す図解化の例

図解化のポイントは以下の3つである。

(1) 対象からキーワードを抽出し，図解の要素として枠で囲む。

(2) 要素同士の関係を矢印で書く。

(3) 必要に応じて矢印の上にどのような関係なのか書く。

このポイントを押さえた上で，POSシステムの情報の流れや処理の仕組みを図解化したものが図表 1 である。

情報システムの情報の流れや処理の仕組みは，資料等を目で見るだけではなかなか理解することができない。実際に自分の手で情報システムの概念を図解化してみることで，情報システムにおける情報の流れや処理の仕組みについて理解することができる。

演習 2

EXERCISE

演習 1 で調べた情報システムの情報の流れや処理の仕組みを図解化し，図解化したからこそ見えてきたことについてまとめましょう。

3 情報システムの処理形態

情報システムの情報の処理形態には様々なものがある。例えば人が仕事をする場合，一人で仕事をするときと複数人で仕事をするときでは，仕事のやり方が異なる。複数人で仕事をする場合は，事前に仕事のねらいに沿った計画を立て，人に適切な役割を割り振り，仕事量を分散して作業する必要がある。コンピュータの処理も同じで，1台で集中的に処理をするか，あるいは分散して処理するかによって処理形態が異なる。

図表 5 に示すように，処理のタイミングなどによる情報システムの処理形態としては，バッチ処理，オンライントランザクション処理，リアルタイム制御処理，対話型処理などがある。

処理名	利用されている情報システム	処理内容
バッチ処理	給与計算システム	情報を一括処理する形態。 一定期間あるいは一定量データをためてから，一括して処理をする。データを投入してから処理結果を得られるまでの処理手順が確立しており，その手順に沿って処理を行う。
オンライントランザクション処理	座席予約システム 預金システム	データを即時処理する形態。 データの発生と同時に処理するが，高い信頼性と即時性が要求され，多数の端末からの同時要求に対しても，応答時間を維持できるようにすることが重要である。
リアルタイム制御処理	自動運転システム	極めて厳しい即時性を要求される形態。カメラやセンサなどを使って常に状態を監視することが要求される。
対話型処理	プログラム開発	人の判断を加えながら情報処理を進める形態。ユーザーがディスプレイ上に表示されたアイコンなどを選択し，キーボードからコマンドを入力することによって，コンピュータとの情報のやり取りを行う。

図表 5　情報システムの処理形態

演習 3

EXERCISE

演習 1 で調べた情報システムは，どの処理形態か調べてみましょう。

4 情報システムの集中処理と分散処理

情報システムにおけるネットワークの仕組みでは，集中処理と分散処理がある。拡張性の高さや障害への耐久性の面より，1台のホストコンピュータに複数の端末を接続し，ホストコンピュータに集中して処理させる集中処理から，複数台のコンピュータが処理を分担して処理を行う分散処理に移行するものもみられる。

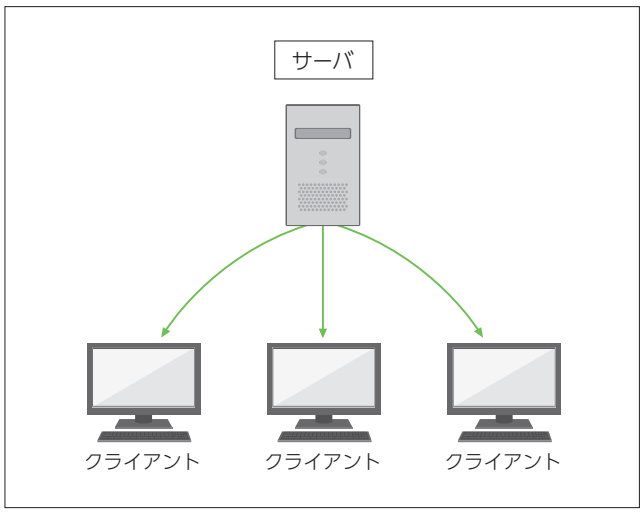
集中処理は，保守や運用管理，セキュリティの管理が容易ではあるが，ホストコンピュータが故障するとシステム全体が停止してしまうデメリットがある。

一方の分散処理システムは，1台のコンピュータが故障しても処理を継続できるため信頼性が高く，機能の拡張が容易ではあるが，保守やセキュリティ管理，運用管理が複雑になるデメリットがある。ただし近年では，インターネットを経由して利用するクラウドコンピューティングが普及しつつあり，集中処理か分散処理かを区別することの意義が薄れつつある。

5 クライアントサーバシステムとP2Pシステム

情報システムの分散型システムとして，図表 6 に示すようなクライアントサーバシステムがある。サービスを要求するクライアントと，要求されたサービスを提供するサーバで構成されている。

クライアントサーバシステムは，1つのサーバに複数の機能を持たせることも，1つの機能を複数のサーバに分散することもできる。サーバの役割として，図表 7 に示すような種類がある。



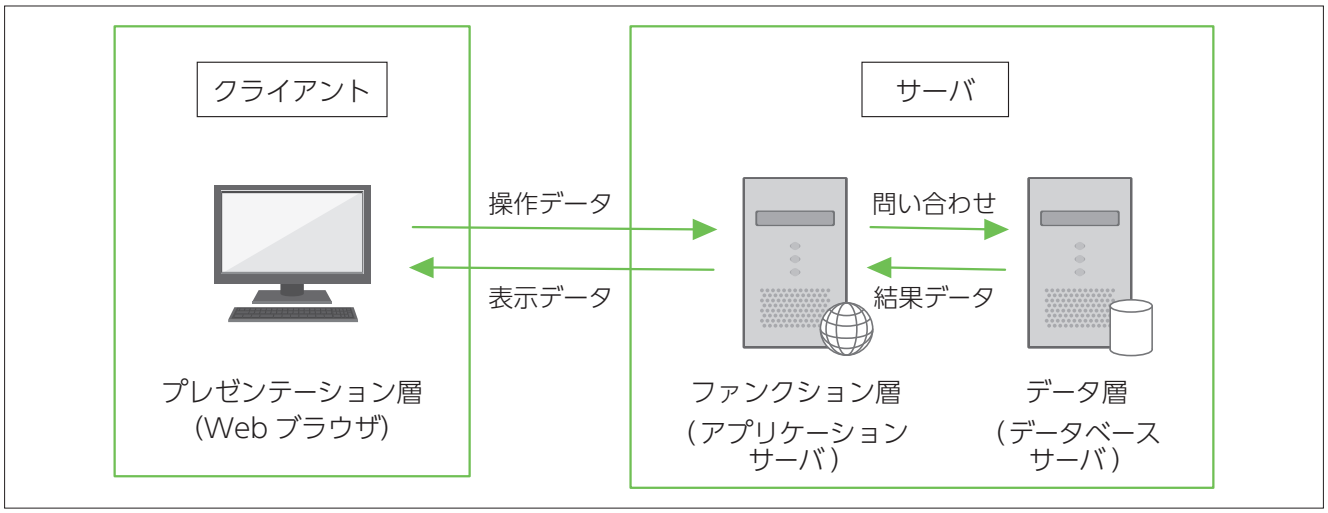
図表 6　2層クライアントサーバシステム

サーバ	役割
ファイルサーバ	ファイルを集中管理し，ユーザーのアクセス権に従ってファイルを提供する。
Web サーバ	Web ページを管理し，クライアントの要求によって情報や機能を提供する。
データベースサーバ	データベースを一元管理し，アクセス制御，データの登録・削除などを行う。
コミュニケーションサーバ	通信制御を行う。プロトコルの異なるネットワークを接続する機能を持つ。
プリントサーバ	プリンタを共有する機能を提供する。
メールサーバ	電子メールの配送を行う。

図表 7　サーバの種類

図表8に示すように，現在主流となっている3層クライアントサーバシステムは，それまでクライアントが行っていた「データの加工」をアプリケーションサーバに行わせ，クライアントでは「入力や表示」のみを行う仕組みが主流となっている。

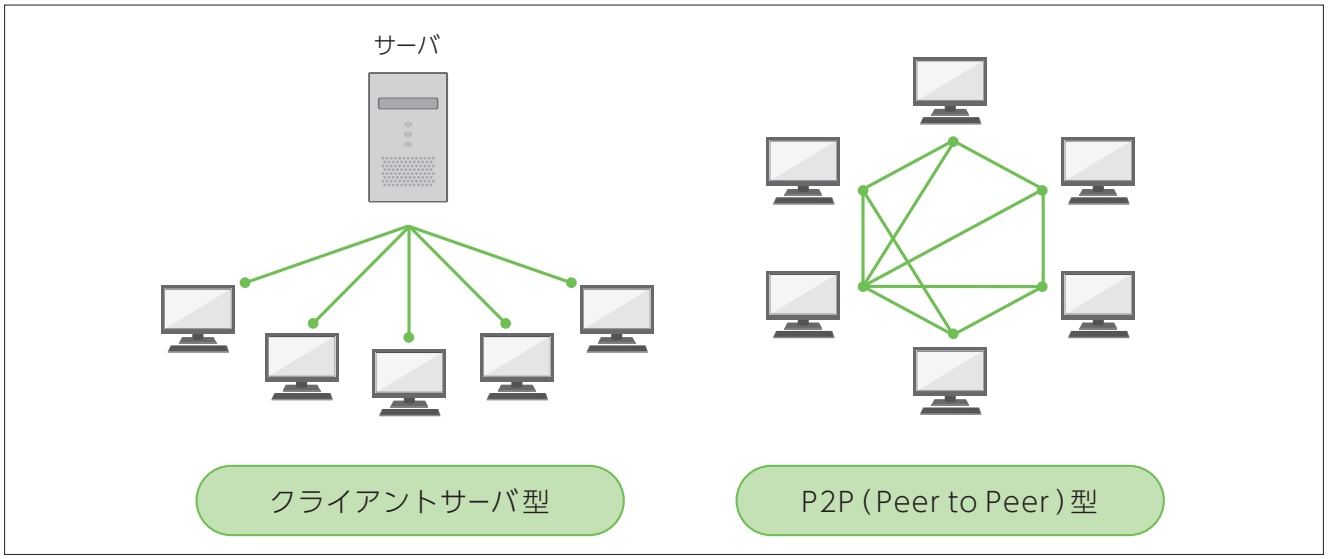
当初のクライアントサーバシステムは，クライアントのアプリケーションがサーバのデータベースにアクセスする形の2層構成であった。しかし，データベースの検索結果が全てネットワークを通じて送られるため，クライアントとサーバ間の通信負荷が問題となった。そこで，クライアントには画面表示などの機能を残し，利用頻度の高いプログラムをアプリケーションサーバに置いてデータベースにアクセスする形の3層構成が主流になり現在でも多く使われている。



図表 8　3 層クライアントサーバシステムの仕組み

情報システムにおいてクライアントサーバシステムがよく用いられているが，暗号資産のブロックチェーンや一部のSNSアプリ(電話アプリやメッセンジャーアプリなど)には，サーバを介さずに情報のやり取りをするP2P（Peer-to-Peer)システムが用いられているものもあり，IoT分野でも利用されている。

図表9に示すように，P2Pはサーバを介さない。そのためクライアントサーバシステムに対して，低コストで拡張性の高いネットワークが構築でき，またサービス提供までの時間が短縮できるメリットがある。その一方で，データが端末に分散して置かれることから，データの流出時にくい止めることが容易でないというデメリットがある。



図表 9　情報システムのネットワーク形態の比較

6 組み込みシステム

パソコンなどの汎用的なシステムとは異なり，特定の機能を実現するために，専用化されたコンピュータハードウェアと，それを制御するソフトウェアから構成されている組み込みシステムがある。図表10 に示すように，電子機器の多くは組み込みシステムと言える。組み込みシステムは，機器にコンピュータを組み込むことで，ソフトウェアを変更するだけで，製品の改良を低コストで実現できるメリットがある。

機器	ハードウェア
通信機器	携帯電話，スマートフォン，FAX
運輸機器	自動車
家電機器	テレビ，洗濯機，冷蔵庫，電子レンジ
医療機器	X 線断層撮影装置，磁気共鳴画像診断装置，陽電子放出断層撮影装置
産業機器	エレベーター，信号機，ATM

図表 10　組み込みシステムの種類の例

7 情報システムの効果的な活用方法について

情報システムの効果的な活用方法について考えるためには，これまでこの章で述べてきた，情報システムに用いられるハードウェア・ネットワーク・ソフトウェアの仕組みを理解した上で，情報システム全体の情報の流れを把握することが重要である。

ハードウェアではコンピュータやセンサ，組み込みシステムなどのハードウェアの仕組み。ネットワークでは，クライアントサーバシステムやP2Pシステムなどのネットワークデータ通信の仕組み。ソフトウェアでは，ハードウェアやネットワークを制御するソフトウェアや処理形態の仕組み。これら3つの情報技術を理解した上で，情報システム全体の情報の流れを考えることが重要である。これらを踏まえた上で，情報システムの全体の情報の流れを図解化することにより，情報システムがどのような利便性を生み，社会にどのような影響を与えるかが見えてくる。

また，GPS（Global Positioning System）とGIS（Geographic Information System）など，2つ以上の情報システムを組み合わせることで，より高度な情報システムが開発されている。こうした複数の情報システムを組み合わせることで，情報システムの仕組みは複雑にはなるが，新しい利便性を生み出すことができる。このような情報システムの効果的な活用についても考えていくことが重要である。

演習 4

EXERCISE

既存の情報システム 2 つを組み合わせることで，新しい利便性を生み出すことができそうな情報システムを提案してみましょう。

【参考文献】
●「栢木先生の基本情報技術者教室」栢木厚 著　技術評論社（2019）

学習活動と展開

学習活動の目的

情報システムにおける，情報の流れや処理の仕組みについて理解し，情報システムが提供するサービスが与える効果と影響について考えられるようになる。

学習活動とそれを促す問い

	問　い	学習活動
展開 1	身近な情報システムを挙げ，情報システムにユーザーが提供する情報と，その情報システムが提供する利便性について考えてみよう。	ユーザーが情報システムから集める情報と，情報システムが提供する利便性について考える。 情報システムのデータの流れがどのようになっているか，データの流れを把握する。
展開 2	身近な情報システムを挙げ，どのような処理形態なのか考えてみよう。	情報システムがどのような用途で利用されているか考え，どのような処理形態が適切であるか考える。
展開 3	複数の情報システムを組み合わせた，高度な情報システムについて提案してみよう。	複数の情報システムのハードウェア・ネットワーク・ソフトウェアの仕組みを理解した上で，組み合わせた情報システムが提供するサービスが与える効果と影響について考える。

展開 1

問　い

身近な情報システムを挙げ，情報システムにユーザーが提供する情報と，その情報システムが提供する利便性について考えてみよう。

学習活動

- ユーザーが情報システムから集める情報と，情報システムが提供する利便性について考える。
- 情報システムのデータの流れがどのようになっているか，データの流れを把握する。

指導上の留意点

情報システムのデータの流れを DFD（201 ページ）で指導した場合，様々な図形がでてくるため，指導が困難になる場合がある。その場合は図解化など，データの流れを把握しやすい手法を取り入れてデータの流れを表現させる。

展開 2

問　い	身近な情報システムを挙げ，どのような処理形態なのか考えてみよう。
学習活動	情報システムがどのような用途で利用されているか考え，どのような処理形態が適切であるか考える。
指導上の留意点	処理形態にはそれぞれの形態に応じたメリットやデメリットがある，これらを整理させた上で適切な処理形態を考えられるようにする。

展開 3

問　い	複数の情報システムを組み合わせた，高度な情報システムについて提案してみよう。
学習活動	複数の情報システムのハードウェア・ネットワーク・ソフトウェアの仕組みを理解した上で，組み合わせた情報システムが提供するサービスが与える効果と影響について考える。
指導上の留意点	複数の情報システムを組み合わせると仕組みが複雑になるため，まずは２つの情報システムを組み合わせた情報システムの事例を提示する。

まとめ

まとめ	情報システムのデータの流れを図解化などで表現させ，情報システム全体の流れを把握させた上で，情報システムによって提供されるサービスについて，その在り方や社会に果たす役割と及ぼす影響力について考察できるようになる。

学習

20 情報システムの情報セキュリティ

研修内容

研修の目的

- 情報システムにおける情報セキュリティを確保する方法や技術について理解する。
- 暗号化，ファイアウォール，個人認証，アクセス制御，ネットワークのセグメント化など，セキュリティを確保する方法について理解する。
- 情報システムのサービスが停止したときや，あるいは個人情報が漏洩したときの影響力について考え，これらに対応できる力を身に付ける授業ができるようになる。
- 人間が安全かつ快適に活用できる情報システムの在り方や社会に果たす役割について考察させる授業ができるようになる。

1 情報システムにおける情報セキュリティ

情報システムにおける情報セキュリティを考える際は，組織としてどのようにシステムの安定性やデータの保守性を確保していくかという観点で考えていかなければならない。「情報Ⅰ」では個人の情報セキュリティ対策について扱ったが，「情報Ⅱ」では組織や情報システムとしての情報セキュリティ対策について扱っていく。

現代社会にとって情報システムは，日常生活において欠かせない存在になった。情報システムはユーザーから情報を収集・共有し，伝達することで大きな利便性を向上させた一方で，情報システムにトラブルが発生すると大きな事故につながる危険性を抱えている。図表1に示すような情報システムのトラブルが起こると，企業や組織の社会的な信頼を大きく失墜させてしまう可能性がある。

情報システムのトラブル	トラブルの影響
機密情報の漏洩	ウイルスへの感染や不正な情報の持ち出し，記録媒体の紛失は，組織の競争力や信頼を大きく損なう可能性がある。
個人情報の流出	保有する個人情報を流出させてしまった場合は，賠償や訴訟など大きな問題にまで発展する可能性がある。
Web ページ改ざん	組織の顔ともいえる Web ページの改ざんは，イメージ損失につながる。また，Web サーバがウイルスに感染していると，訪問者にウイルスを感染させてしまう。
システム停止	システムが停止してしまうと，最悪の場合は業務そのものが停止してしまう。
ウイルス感染	ウイルスが既に感染したパソコン内で複製され，他のパソコンに感染を広げ，上記で述べた全てのトラブルを引き起こす可能性がある。

図表1　情報システムのトラブルとその影響の例

これらのトラブルを防ぐために，情報セキュリティの3要素である，機密性(認められた者だけがアクセスできる)・完全性(情報や処理方法が正確である)・可用性(必要なときに情報にアクセスできる)を情報システムにおいて確保することが重要である。

これらの3要素は最大限確保すればいいものではなく，情報システムの運用コストや，導入コストを考慮した上で検討していかなければならない。

演習 1

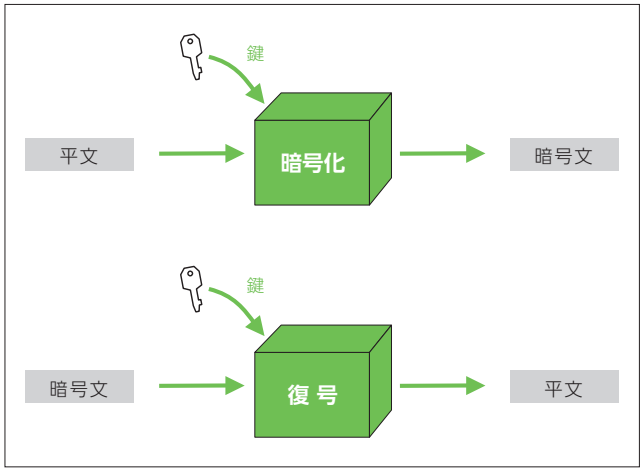
EXERCISE

図表 1 で示している情報システムのトラブルについて，実際に起こったトラブルの内容とトラブルによる社会的な影響について調べ，トラブルを防ぐためにどんな手だてを行えばいいか考えてみましょう。

2 データの暗号化による情報流出の防止

情報システムはインターネット上で，不特定多数のサーバを経由してデータのやり取りを行う。その際に送信者から受信者にデータが届くまでに，悪意のある第三者にデータを盗み見られる「盗聴」，送信者になりすまし，受信者にデータを送信する「なりすまし」，送信されたデータを不正に書き換えて受信者に送信する「改ざん」などの脅威がある。また，情報システムに保管されていたデータが外部に流出した場合，機密情報の漏洩や個人情報の流出となってしまう。

こうしたネットワーク上の脅威やデータ流出に対して考えられた技術が，データの暗号化である。暗号化とは，元のデータのビット列を，容易にその内容が分からない別のビット列に変換する技術のことである。逆に復号とは，暗号化されたデータを元のデータに戻すことである 図表 2。情報システムに保管しているデータを暗号化しておくことで，仮にデータが流出しても，暗号化されたままのデータではその内容が読めない状態となるためデータを守ることができる。データの暗号化には様々な種類がある，その一例を 図表 3 に示す。



図表 2　暗号化と復号の仕組み

出典：「安心してインターネットを使うために　国民のための情報セキュリティサイト」(総務省) をもとに作成
https://www.soumu.go.jp/main_sosiki/joho_tsusin/security/basic/structure/02.html

暗号方式	内容
共通鍵暗号方式	暗号化の鍵も復号の鍵も同じ鍵を使用する暗号方式。鍵を盗まれないように大切に管理する必要がある。代表的な方式に AES などがある。盗聴対策に効果的である。
公開鍵暗号方式	暗号化する公開鍵と復号する秘密鍵の 2 種類の鍵を使用した暗号方式。公開鍵だけを広く公開し，秘密鍵は公開しないので情報を安全にやり取りできる。不特定多数の人と情報をやり取りすることに向いており，代表的な方式に RSA や楕円曲線暗号がある。盗聴対策に効果的である。
デジタル署名	送信者の秘密鍵で電子文書の署名データを生成し，送信者の公開鍵で署名データの作成者が電子文書の送信者と同一であることを検証する仕組み。改ざん対策に効果的である。
SSL（Secure Sockets Layer）／ TLS（Transport Layer Security）	インターネット上で個人情報やパスワード，クレジットカードなどの情報を暗号化して送受信するプロトコル。SSL は，共通鍵暗号方式の利点と，公開鍵暗号方式の利点の双方を生かすような暗号方式である。SSL の仕組みは，TLS に引き継がれている。

図表 3　データの暗号化の仕組みの例

3 情報システムのファイアウォール

情報システムが外部からアクセスを受けるときに，認められた者だけがアクセスできる機密性を保持しなければならない。この機密性を確保する方法として，ファイアウォールがある。

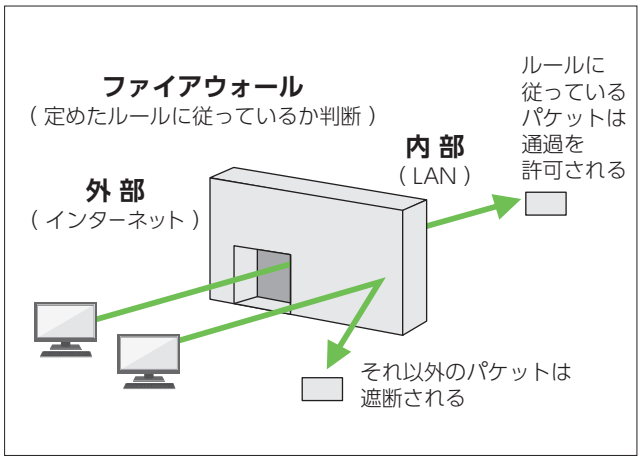
ファイアウォールは防火壁という意味があり，情報システムがある組織内ネットワークとインターネットの接続点で，パケットフィルタリング方式やアプリケーションゲートウェイ方式などを用いて，外部からの不正なアクセスを防止している。

図表4に示すパケットフィルタリング方式は，パケットのヘッダーを見て，許可されたパケットだけを通過させる仕組みである。パケットの送信元のIPアドレスとポート番号，宛先のIPアドレスとポート番号間でルールを決め，そのルールに従って通過させるかどうかを判断する仕組みである。パケットフィルタリングはルータに実装されている場合が多い。

TCP/IPネットワークでは，IPアドレスにより通信ホストを特定することができる。しかし，1台のホスト上で複数の通信サービス(アプリケーション)が動作していることがある。このアプリケーションを識別する番号をポート番号という。ポート番号には0から65535までの番号が付けてあり，そのうち0~1023まではウェルノンポート(よく知られているポート番号)と呼ばれ，特定のアプリケーションのために予約されている。ウェルノンポートの一部とそれに対応するプロトコルを図表5に示す。

ファイアウォールでは通過を許可するポート番号を設定することで，情報システムの機密性を保持することができる。

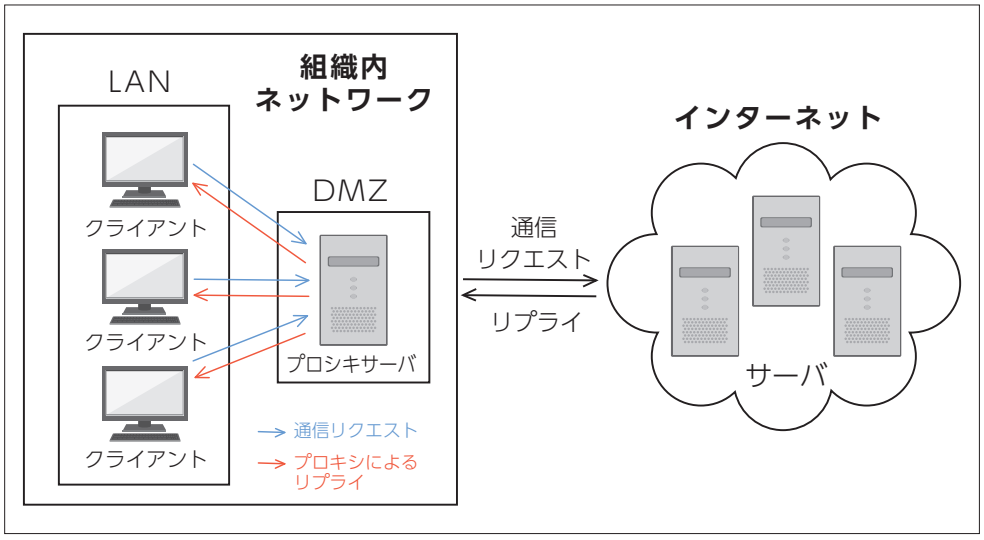
アプリケーションゲートウェイ方式は，プロキシプログラムを使用することで，組織内ネットワークとインターネットを切り離し，アプリケーションレベルでフィルタリングを行う方式である。クライアントからインターネット上のサーバにアクセスがあった場合は，プロキシプログラムがクライアントからの要求を中継し，更にサーバからクライアントの応答も中継することで組織内のクライアントを守ることができる図表6。



図表 4　パケットフィルタリング方式

ポート番号	プロトコル
22	SSH
53	DNS
80	HTTP
123	NTP
443	HTTPS

図表 5
ポート番号とプロトコルの例



図表 6　アプリケーションゲートウェイ方式

このときにDMZ（DeMilitarized Zone）と呼ばれる，内部インターネットと外部インターネットの両方から隔離された区域を設定することで，外部から不正なアクセスがあったとしても，内部ネットワークに被害が及ばないようにしている場合が多い。

演習 2

EXERCISE

暗号化やファイアウォールを設定していなかったために起こった情報システムのトラブルについて，その内容と社会的にどのような影響を与えたのか調べ，どのような対策をすれば防げたのかを考えてみましょう。

4 情報システムの個人認証

情報システムを利用する場合は，まず利用者が使用することを許可されている本人であるかどうかを確認するユーザ認証を行う必要がある。

ユーザ認証は，利用者を一意に識別するためのユーザIDと，利用者本人であることを確認するためのパスワードを使用して行われることが多いが，情報システムではセキュリティを高めるために図表7のユーザ認証も使われている。

ユーザ認証	内容
パスワード認証	ユーザーだけが知っている ID とパスワードにより認証する方式。
ワンタイムパスワード	一度しか使用することができない使い捨てのパスワードをその都度発行する方式。
コールバック	アクセス権を持つ端末であることを確認するため，回線をいったん切り，システム側から再発信して通信を開始する方式。
バイオメトリクス認証	身体的特徴や行動的特徴を抽出して認証する方式。身体的特徴としては，指紋や静脈認証がある。行動的特徴としては，署名するときの速度や筆圧から特徴を抽出する方式などがある。
多要素認証	ユーザーだけが知っている知識情報，ユーザーだけが持っている所持情報，ユーザーだけの生体情報などのうち，2 つ以上を組み合わせて認証する方式。

図表 7　代表的なユーザ認証の例

5 アクセス制御とアクセス権

情報システムで共有するディレクトリやファイルは，許可されていない人にデータを勝手に見られたり，書き換えられたりすると大きな問題につながる。そこでディレクトリやファイルに対して，ユーザーごとに行動を制限している。アクセス制御の方法は，「認証・認可・監査」という3つのモデルから成り立っており，それぞれに役割が異なる。

認証とは，対象の正当性や真正性を確かめるものであり，コンピュータにログインするユーザーが本人かどうかを識別するユーザ認証などがある。

認可とは，管理者があらかじめ定めた条件を満たしたユーザーのみアクセスを許可する制御方法である。管理者が定めた条件をアクセスコントロール（ACL）といい，その条件を満たしたユーザーのみがアクセスすることができる。

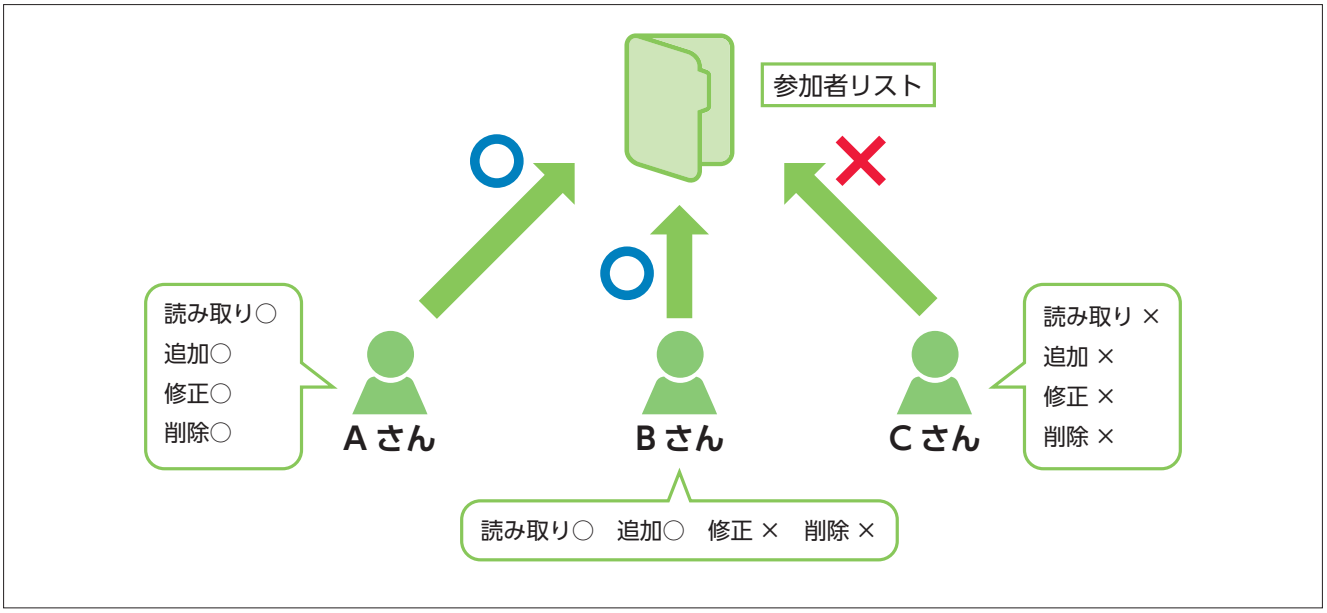
監査とは，認証や認可でとった処理の過程で発生したログを記録し，検証した結果，条件を見直すことによってより高い精度のアクセス制御を行う制御方法である。アクセス制御方式の例を図表8に示す。

モデル方式	内容
任意アクセス制御	最も一般的なアクセス制御。ユーザーが自分で情報システムのアクセス制御を行う方式。ユーザーの自由度は高く，管理者の負担は少ないが，ルールの統一が難しくセキュリティ面で問題がある。
強制アクセス制御	管理者がルールを決め，他のユーザーがルールを変更することができない方式。任意アクセス制御よりも一般的にセキュリティが高くなる。
役割ベースのアクセス制御	管理者の持つ権限の一部を特定のユーザーに与える方式。権限を与えられたユーザーは，その範囲内で業務上必要な担当者にアクセス権を付与したりすることができる。

図表 8　アクセス制御方式の例

また実際にユーザーが情報システムにアクセスできたとしても，ファイルやフォルダの読み取りや書き込みのアクセス権(許可，拒否)を設定することで，情報システムの機密性をより高めることができる。アクセス権は，フォルダやファイルに対して，ユーザーごとに「読み取り」「追加」「修正」「削除」などの権限を設定することができ，また複数ユーザーをまとめたグループごとにアクセス権の設定を行うことができる。

図表 9 に示す参加者リストのフォルダのアクセス権の設定をした場合，Aさんは参加者リストのフォルダの全ての権限を有しており，Bさんはフォルダのデータを見る，あるいは新しいデータを追加することはできるが，既存のデータを修正したり，削除したりすることができない。Cさんは参加者リストのフォルダの存在すら見ることができないようになっている。



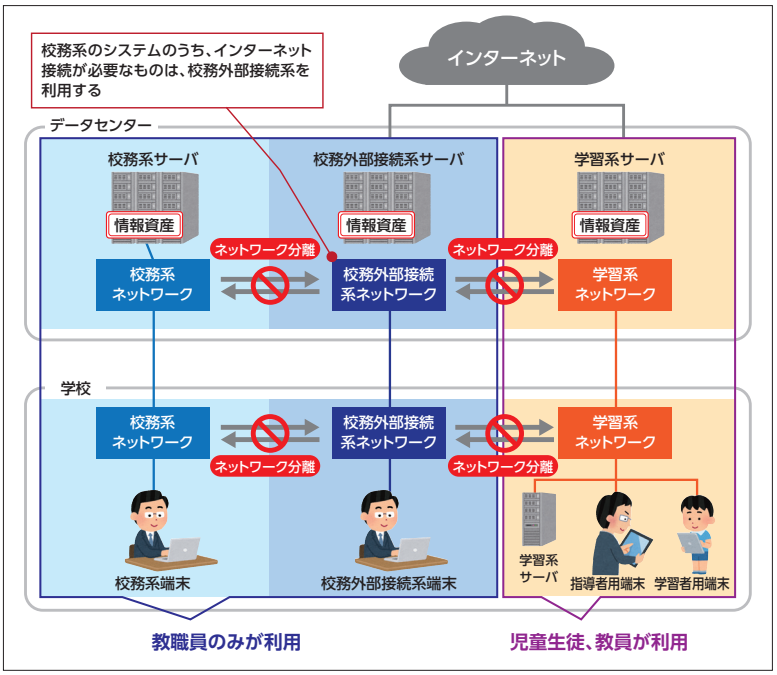
図表 9　アクセス権の設定例

6 ネットワークのセグメント化

ネットワークで機密性の度合いが異なる情報ごとに，セグメントを設定することをネットワークのセグメント化という。教育現場においては，文部科学省より「教育情報セキュリティポリシーに関するガイドライン」(令和元年12月版)が公開されており，セキュリティ対策の例として，図表10に示すように教員が使用するデータを扱う校務系ネットワークと生徒が使用するデータを扱う学習系ネットワークを別のセグメントに分けてネットワークを構築することを示している。

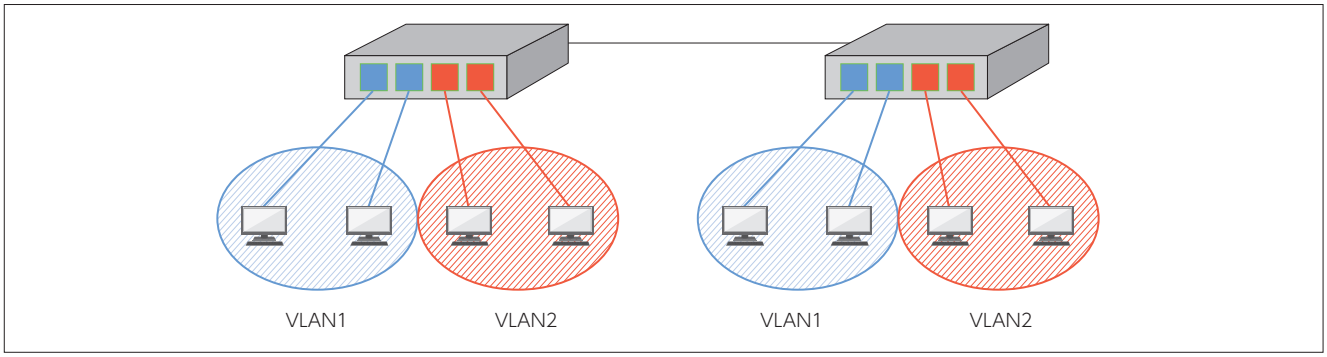
こうしてセグメントを分割することで，学習者用端末や指導者用端末から，校務系ネットワークや校務外部接続系ネットワークへのアクセスを防ぐことができる。

物理的な接続形態とは独立して仮想的なLANセグメントを作るVLAN（Virtual LAN）と呼ばれる技術で，情報セキュリティの機密性を高めることもできる。VLANはその機能を持つLANスイッチを使用し，接続された機器を管理者の設定した任意のグループに分けることで，ネットワークを分割することができる。図表11では，1つのネットワーク内で直接通信できる範囲（ブロードキャストドメイン）を，VLANを使って2つに分割した様子を示している。物理的な1つのネットワークを仮想的に2つのネットワークに分割することで，柔軟にネットワークを構成・変更することができ，組織内における情報セキュリティの機密性を向上させることができる。



図表 10　教育現場におけるセグメント化

出典：『教育情報セキュリティポリシーに関するガイドライン』ハンドブック」文部科学省
https://www.mext.go.jp/content/20200225-mxt_jogai02-100003157_003.pdf



図表 11　VALN による仮想的なセグメント化

演習 3

EXERCISE

アクセス制御やネットワークのセグメント化を設定していなかったために起こった情報システムの事故について，その内容と社会的にどのような影響を与えたのか調べ，どのような対策をすれば防げたのかを考えてみましょう。

演習 4

EXERCISE

これまで調べてきた事例をもとに，人間が安全かつ快適に活用できる情報システムの在り方や社会に果たす役割について考えてみましょう。

【参考文献・参考サイト】

- 「栢木先生の IT パスポート教室」栢木厚 著　技術評論社（2019）
- 「栢木先生の基本情報技術者教室」栢木厚 著　技術評論社（2019）
- 「安心してインターネットを使うために　国民のための情報セキュリティサイト」総務省
  https://www.soumu.go.jp/main_sosiki/joho_tsusin/security/business/executive/01.html

学習活動と展開

学習活動の目的

組織や情報システムとしての情報セキュリティ対策を実行し，人間が安全かつ快適に利用できるための情報システムについて考えられるようになる。

学習活動とそれを促す問い

	問　い	学習活動
展開1	既存の情報システムにトラブルが発生した場合，どのような影響があるか考えてみよう。	過去の情報システムに起こったトラブルについて調べ，どのような社会的な影響があったのか考えさせる。
展開2	情報システムの情報の漏洩を防ぐためには，どのような対処をすればいいか考えてみよう。	暗号化やファイアウォール，個人認証などの技術を基に，情報システムの利用用途に合わせた適切な技術を考える。
展開3	人間が安全，かつ快適に活用できる情報システムの在り方について考えてみよう。	安全かつ快適に運用され，ユーザーからの評価の高い情報システムについて調べ，情報システムの在り方について考える。

展開1	
問　い	既存の情報システムにトラブルが発生した場合，どのような影響があるか考えてみよう。
▼	
学習活動	過去の情報システムに起こったトラブルについて調べ，どのような社会的な影響があったのか考えさせる。
▼	
指導上の留意点	過去の情報システムのトラブルはどのような手だてがあれば防ぐことができたのか考えさせる。

展開 2

問　い	情報システムの情報の漏洩を防ぐためには，どのような対処をすればいいか考えてみよう。
学習活動	暗号化やファイアウォール，個人認証などの技術を基に，情報システムの利用用途に合わせた適切な技術を考える。
指導上の留意点	情報セキュリティを高めることも大切だが，運用コストやユーザーの利便性についても考えた上で情報セキュリティ対策を考えるようにする。

展開 3

問　い	人間が安全，かつ快適に活用できる情報システムの在り方について考えてみよう。
学習活動	安全かつ快適に運用され，ユーザーからの評価の高い情報システムについて調べ，情報システムの在り方について考える。
指導上の留意点	その情報システムの在り方や社会的に果たす役割についても考える。

まとめ

まとめ	情報システムの果たす役割や社会的な影響を踏まえた上で，運用状況に応じた組織や情報システムとしての適切な情報セキュリティ対策が行えるようになる。

学習

21 情報システムの表し方

研修内容

研修の目的

- 情報システムの設計において要件定義が必要であることを，生徒に理解させることができるようになる。
- ユースケース図を使って利用者とシステムとのやり取りを図示し，システムのモデル化を生徒ができるようになる。
- シーケンス図を使ってオブジェクト指向の考え方を理解し，システム設計に応用することを生徒ができるようになる。
- DFDやアクティビティ図を使ってデータの流れに着目したシステムの図表化を生徒ができるようになる。

1 要件定義

「家を建てる」ときに，何も考えないで建てると後になって「このようなはずではなかった」と後悔することになるだろう。そうならないためにも事前に「どのような家を建てたいか」をじっくり考える必要がある。

情報システムにおいても同様に作成する前に，「どのような機能が必要か」「利用者にはどのような人がいるのか」をじっくり考える必要がある。このことを要件定義と呼び，業務上実現すべき要件である業務要件定義と，業務要件を実現するために必要な情報システムの機能を明らかにする機能要件定義などがある図表1。

業務要件定義	機能要件定義
・利用者がシステムを使う目的を明確にする。 ・利用者がシステムを使うことで得られる利益を明確にする。 ・システムを使うことで利益を得るためのプロセスを定義する。	・目的を達成するために，利用者がシステムを使って行う仕事を明確にする。 ・ユーザーの視点で，システムは何を行う必要があるかを定義する。
例） ・煩雑な在庫管理を情報システムに置き換えることで管理コストをカットする。 ・利用者はシステムを通して，常に正しい在庫数を確認することができる。	例） ・利用者はシステムを使って，「在庫確認」「追加発注」「在庫履歴印刷」ができる。 ・システムは在庫のリストをデータとして保持し，納品後はデータを更新し，在庫確認時は在庫の数を正しく表示する。

図表1　業務要件定義と機能要件定義の違い

演習・1

EXERCISE

「図書館の本の貸出」について情報システムに置き換える場合の要件定義をしてみましょう。

2 利用者とシステムのやり取りの図表化

機能要件定義の中で利用者がシステムを使って行う仕事を明確にする必要がある。だが，言語のみで明確になった事項を整理しようとすると，正しく意図が伝わらない場合があり，正しい情報が共有されない可能性が生じる。そこで，情報システムの開発においては要件定義等を図表化することにより，正しく情報が共有できるように工夫がなされている。ここで紹介する図表化はUML（Unified Modeling Language，統一モデリング言語）と呼ばれている。

(1)ユースケース図

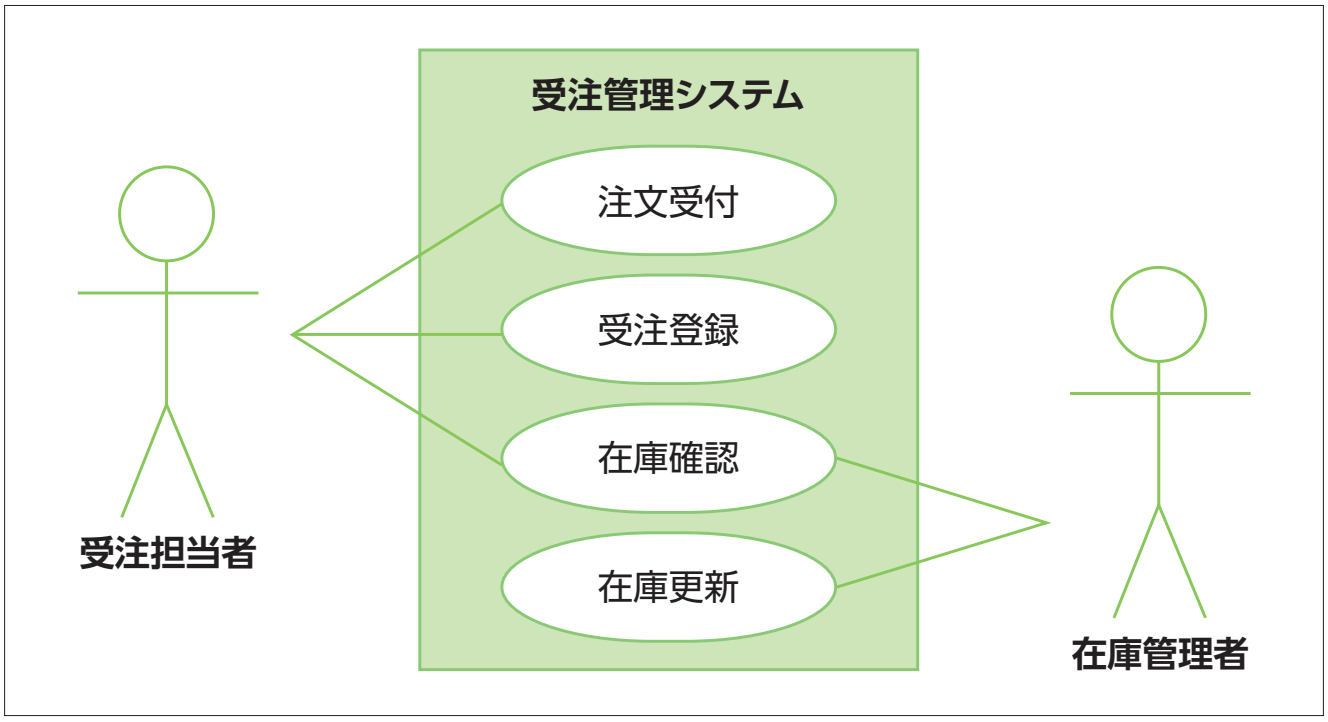
利用者がシステムを使って行う仕事を明確にする場合に使用される図には，ユースケース図などがある。この図は利用者とシステムとのやり取りを表したものであり，名のとおり「システムの使用例」という意味がある。

ここでは一つの例として商品の受注管理システムを開発することを考える。機能要件定義から図表2の項目を作った。

受注担当者が受注管理システムで行うこと	・システムで注文の連絡を受け付ける ・システムに注文を登録する ・システムで在庫を確認する
在庫管理者が受注管理システムで行うこと	・システムで在庫を確認する ・システムで在庫を更新する

図表 2　機能要件定義から洗い出した項目例

上で示した要件定義をユースケース図で表現したものは下の図である図表3。



図表 3　ユースケース図のイメージ

また，ユースケース図の記号には以下の意味がある図表４。

アクター	システムを操作する対象。人型で描き，システム境界の外部に記述する。
ユースケース	システムの機能。機能名を丸で囲み，システム境界の内部に記述する。
システム	複数のユースケースから構成される。アクターとの境界を表すために線で囲む。システム名を上部に記述する。

図表４　ユースケース図の記号

演習 2

EXERCISE

「図書館の本の貸出システム」と「利用者」「司書」を例に，ユースケース図を作図してみましょう。

(2) シーケンス図

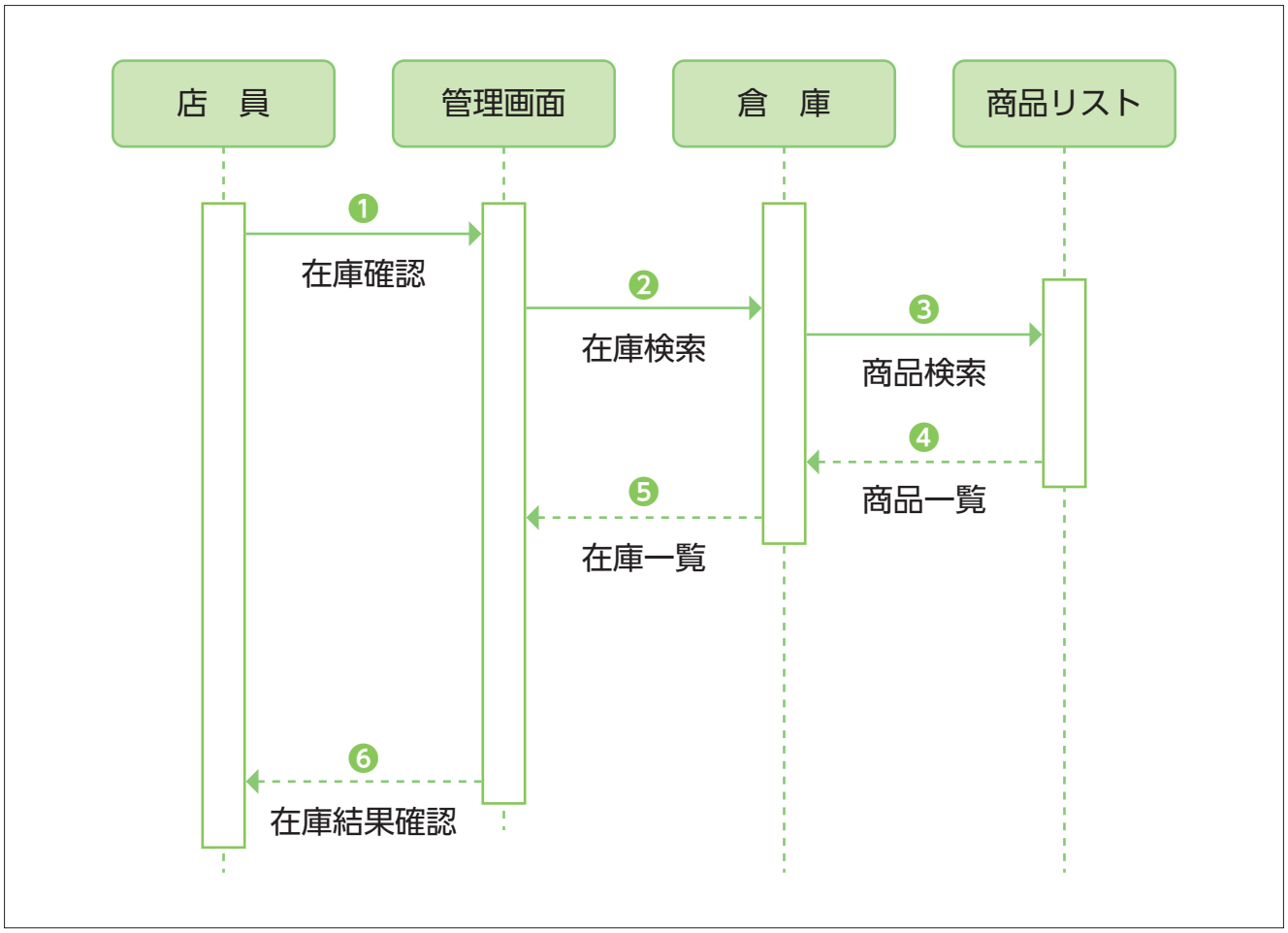
シーケンス図はユースケース図と同じように利用者とシステムのやり取りを表現した図である。ユースケース図と異なる点は，利用者とシステムのやり取りを時間軸に沿って表現する点であり，システムを実現する各要素が時間軸に沿ってどのように関係しているかが分かる。この図では，メッセージと呼ばれる矢印で各要素間の応答を表し，縦軸(上から下)を時系列として応答の順序を表現している。

ここでは一つの例として商品の注文管理システムを開発することを考える。次の処理を例に取り上げる。

店員が管理画面を使って在庫の商品を調べる処理

❶店員は管理画面を使って在庫確認を行おうとする。
❷管理画面の操作によりシステムは倉庫に在庫検索をかける。
❸倉庫はその中にあるデータベース「商品リスト」に商品が存在するか検索をかける。
❹「商品リスト」は商品一覧を倉庫に渡す。
❺倉庫は商品一覧にある在庫を在庫一覧として管理画面に渡す。
❻管理画面は在庫一覧から在庫結果確認として店員に表示する。

次の図では在庫管理システムの一機能をシーケンス図で表現したものである。システムは，「店員」「管理画面」「倉庫」「商品リスト」などの要素から構成され，店員が在庫管理画面から在庫一覧を確認する動作を表現している図表５。



図表 5　シーケンス図

演習 3

EXERCISE

「図書館の蔵書システム」を使って「司書」が問い合わせのあった本について現在図書館にあるかを確認する処理を例に，シーケンス図を作図してみましょう。

3 データの流れに着目した図表化

利用者とシステムとのやり取りを整理すると同時に，システム内をデータがどのように流れているかに着目した表があると，違った視点からシステムを見つめ直すことができる。ここではDFD（Data Flow Diagram）とアクティビティ図を紹介する。

(1) DFD

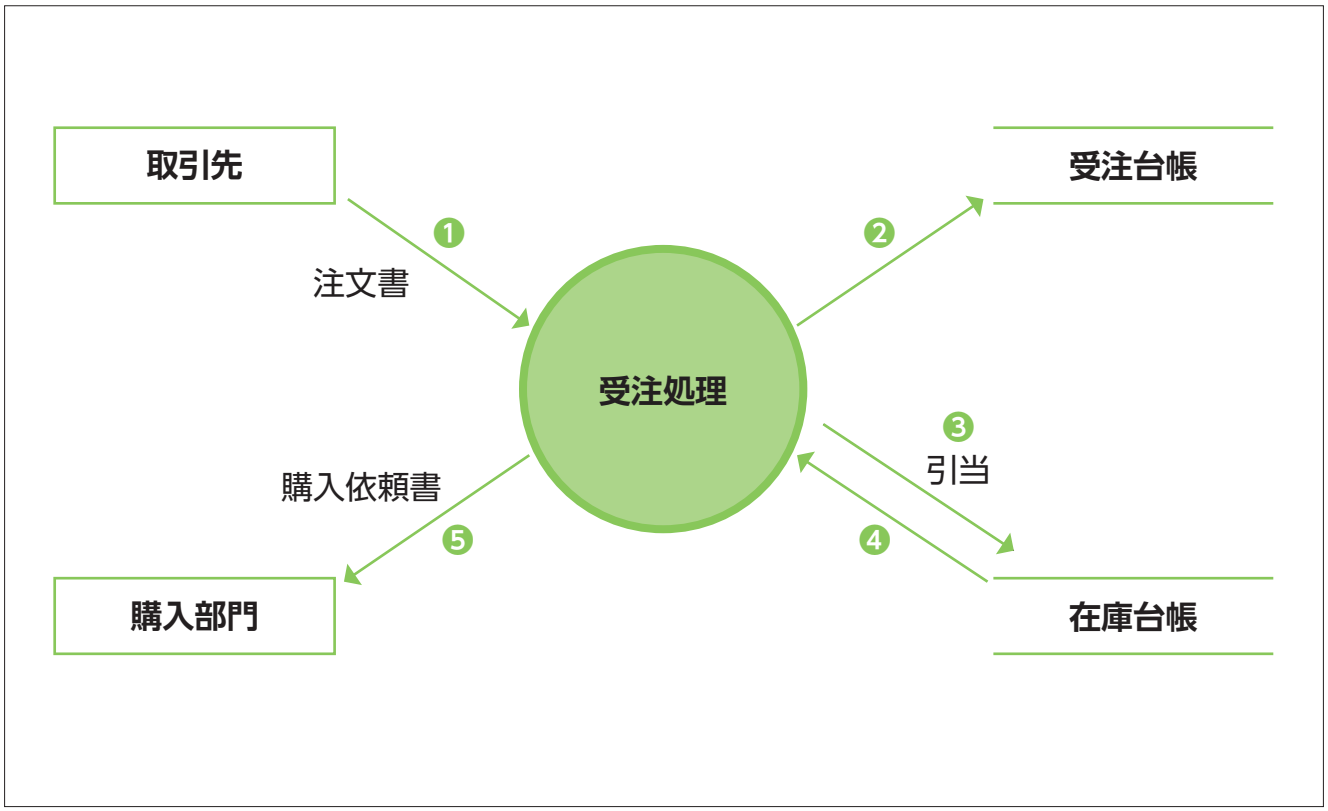
DFDとは，対象となる活動のデータの流れと処理の関連に注目して視覚的に表したものである。具体的には「データがどこから発生して，どのように処理され，どこで吸収されるか」を，次のような記号を用いて表す図表 6。

記号	名称	意味
→	データフロー	データの流れを表す
○	プロセス（処理）	データの処理を表す
＝	データストア（ファイル）	ファイルを表す
□	データの源泉と吸収	データの始まりと終わりを表す

図表 6　DFD の記号の一部

下の図では注文を受けて，在庫が変化していく処理を表現している図表 7。

❶取引先から注文書がくる。
❷注文を受注台帳に登録する。
❸在庫台帳を参照して在庫の引き当てを行う。
❹在庫の引き当てができたときには，在庫台帳が更新される。
❺在庫の引き当てができないときには，購入部門に購入を依頼する。
※ここでいう「引き当て」とは注文分の在庫を確保すること。



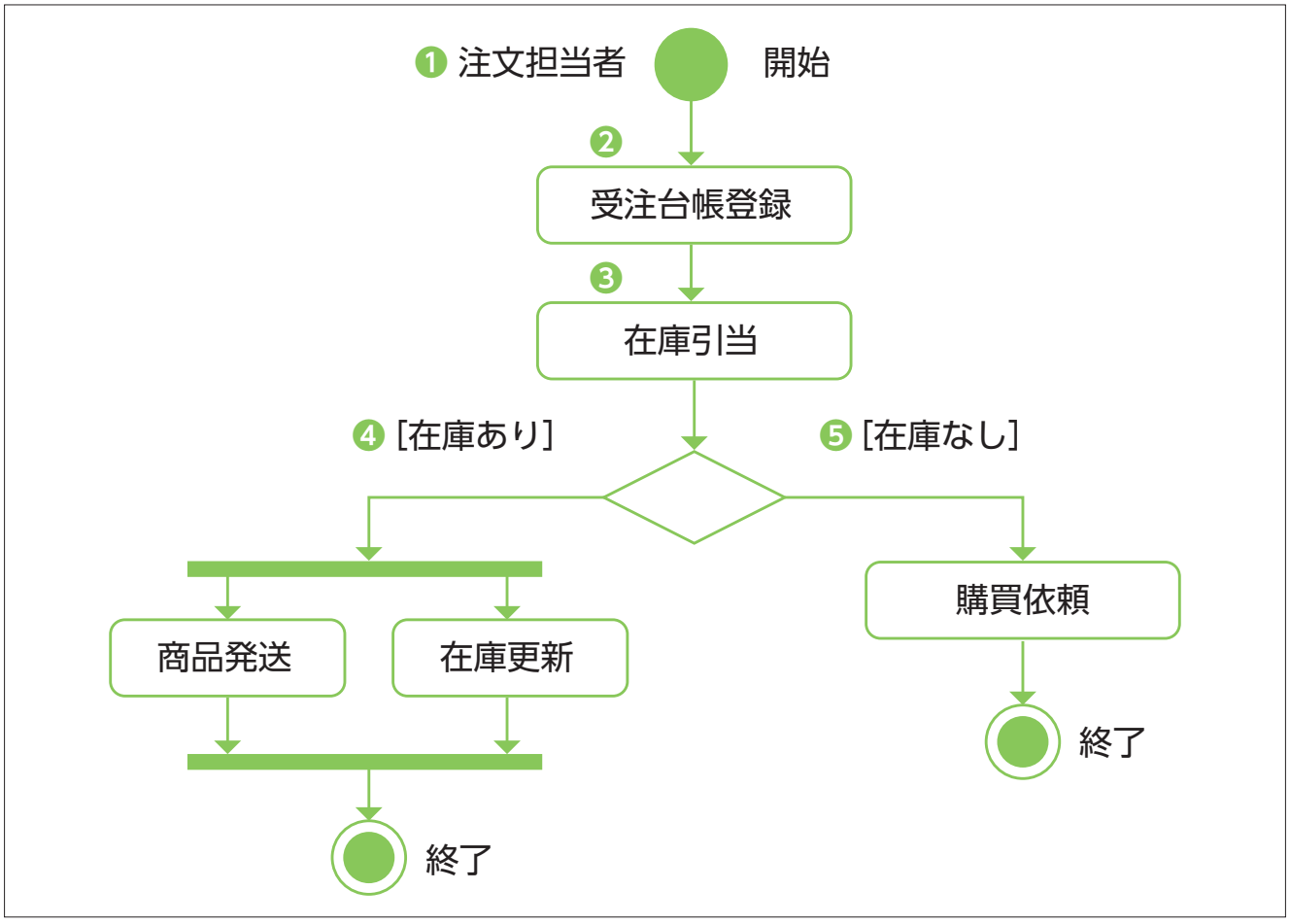
図表 7　DFD の例

(2) アクティビティ図

アクティビティ図は，実行の順序や条件分岐，並行処理など，制御の流れを記述する。よって，アクティビティ図はDFDと異なり，処理の分岐を明確に記述することができる。

下の図では注文を受けて，在庫の変化していく処理を表現している図表8。

❶取引先から注文書がくる。
❷注文を受注台帳に登録する。
❸在庫台帳を参照して在庫の引き当てを行う。
❹在庫の引き当てができたときには，在庫台帳が更新され，商品発送手続きを行う。
❺在庫の引き当てができないときには，購入部門に購入を依頼する。



図表8　アクティビティ図

演習 4

EXERCISE

「司書」が「本を貸し出す」ときの処理についてアクティビティ図を用いて作図してみましょう。

【参考文献・参考サイト】
●「栢木先生の基本情報処理技術者教室」栢木厚 著　技術評論社(2019)
●「IT 専科」http://www.itsenka.com

学習活動と展開

学習活動の目的

洗い出した要件定義に基づいてシステムの図表化ができるようになる。

学習活動とそれを促す問い

	問　い	学習活動
展開 1	「図書館の本貸出システム」を利用者とシステムがどのようなデータのやり取りをしているか図表化しよう。	利用者及び司書の立場になって「図書館の本貸出システム」にはどのような機能が必要か考えて，ユースケース図で図表化する。
展開 2	「図書館の本貸出システム」にて利用者が操作したらデータにアクセスするまでにどのような順番で処理されているか図表化しよう。	利用者がシステムを使って，結果が返ってくるまでどのような処理がなされているか，シーケンス図で図表化する。
展開 3	「図書館の本貸出システム」は状況によって行う処理が変わる。どのように変わるか図表化しよう。	分岐を考えながら，どのような処理がなされているか，アクティビティ図で図表化する。

展開 1

問　い

- 「図書館の本の貸出システム」にはどのような機能が必要かを考えよう。
- 考えた機能をユースケース図で図表化しよう。

▼

学習活動

- 必要な機能を考えることで要求定義の大切さを確認する。
- ユースケース図で表現する。

▼

指導上の留意点

- クラス内で複数人が必要と答えている機能は大半の利用者も必要と感じている機能であることを確認させる。
- 利用者と司書の立場では扱う処理が変わることを確認させる。

展開 2

問　い

- 利用者がシステムを操作してから結果が表示されるまでどのような順番で処理が進んでいるかを考えよう。
- 考えた処理の順番をシーケンス図で図表化しよう。

学習活動

- 時系列で処理の順番を考えることで必要となる処理を考えさせる。
- シーケンス図で表現する。

指導上の留意点

処理の順番を考えさせるときに必要でない処理を取り入れていないかよく考えさせる。

展開 3

問　い

- 図書館の本の貸出システムにおいて処理の分岐があることを確認しよう。
- 処理の分岐を踏まえた上でアクティビティ図で図表化しよう。

学習活動

- どのような場面で処理の分岐が起こるか，ケーススタディから考えさせ列挙させる。
- アクティビティ図で表現する。

指導上の留意点

- あまりに複雑な分岐になった場合は，分岐を減らすなど単純化を進めさせる。
- シーケンス図とアクティビティ図の違いをよく意識させる。

まとめ

まとめ

- 情報システムの開発ではシステムの対象についてよく考察を行うことにより，必要となる処理が明らかになることを認識させる。
- 言語だけでは整理しきれないことを３つの図表化で認識させる。
- ここで整理した図表を使って，システムをどのように構築するか考えると，内部設計につながることを説明する。

学習

22 情報システムの分割と設計

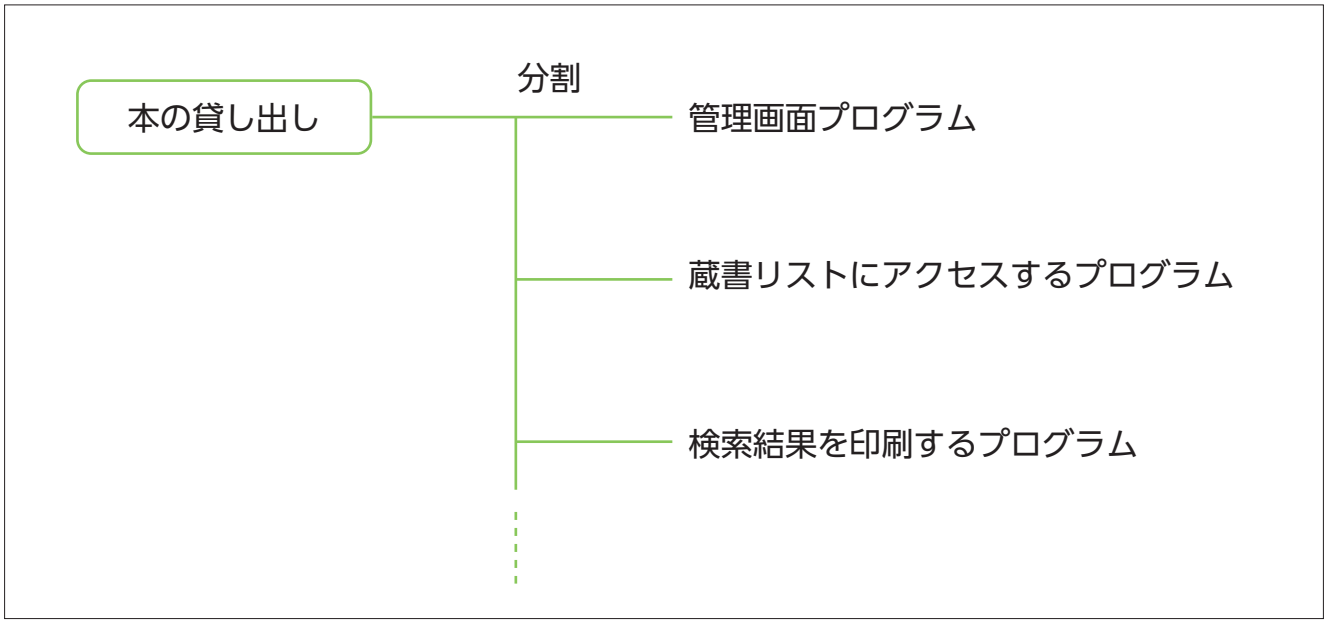
研修内容

研修の目的

- システムの内部を考えるソフトウェア方式設計の重要性を生徒に理解させることができるようになる。
- ソフトウェア方式設計から更に詳細に分割・階層構造にするモジュール分割を行うメリットと必要性を生徒に理解させることができるようになる。
- モジュール分割技法の違いを理解させ，生徒がモジュール分割を行えるような授業ができるようになる。
- 適切にモジュールに分割できるようになるためにモジュール結合度を生徒が理解できるようになる。
- 生徒がカプセル化の概念を理解し，より効果的にモジュールに分割できるようになる。

1 ソフトウェア方式設計

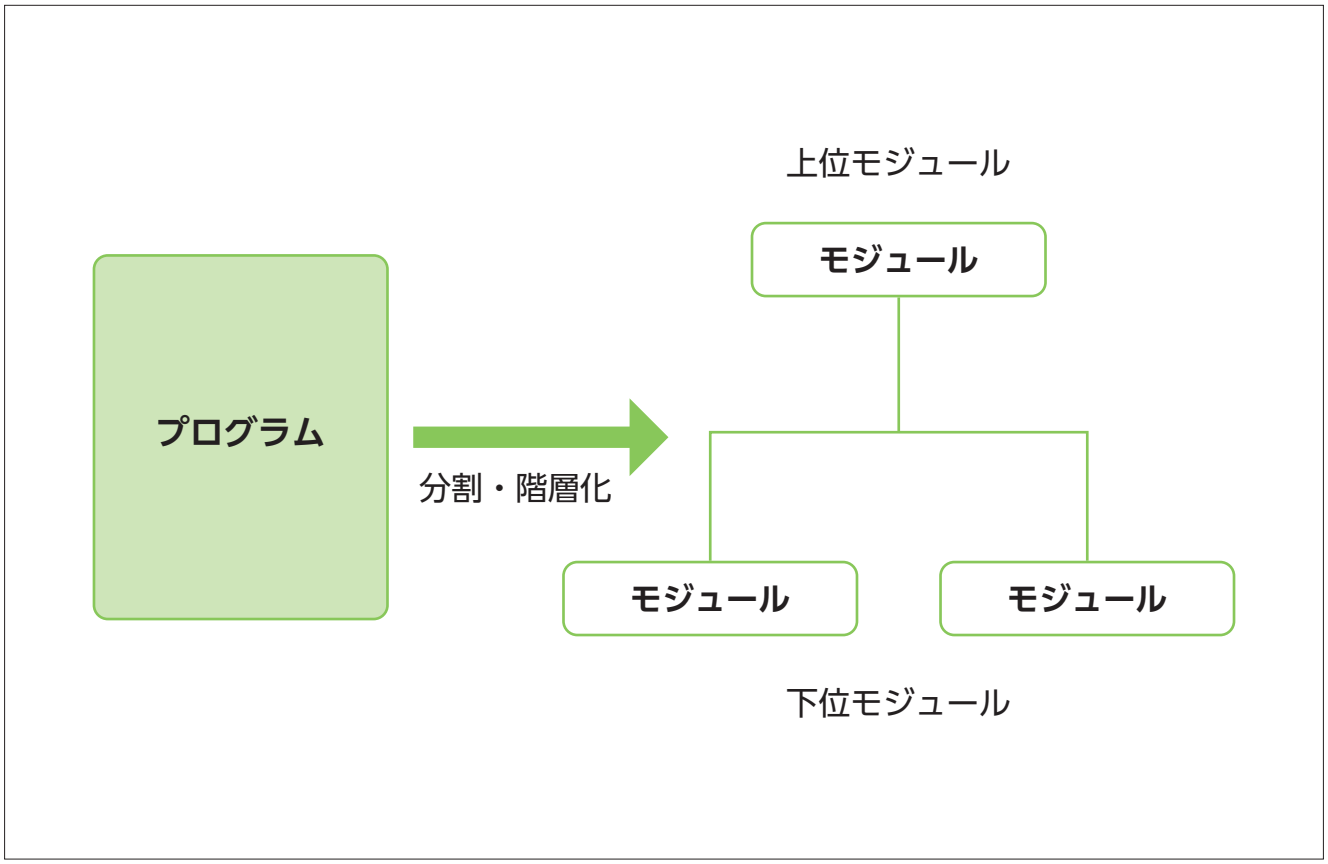
ユースケース図やシーケンス図などでは利用者の視点からシステムに何が求められるかを整理した。これを外部設計と呼び，次に行うべき設計は整理した内容をどのようにプログラムとして記述するかを考えるものである。これを内部設計と呼ぶ。はじめに，システムで提供する予定の機能をどのように分割してプログラムとして記述するかを要件定義と矛盾がないように設計を行う。この設計をソフトウェア方式設計と呼ぶ。図表 1。



図表1　ソフトウェア方式設計のイメージ

2 ソフトウェア詳細設計

ソフトウェア方式設計ではシステムをプログラム単位に分割するが，ソフトウェア詳細設計ではプログラムを更に細かく，モジュールと呼ばれる単位に分割・階層化し，モジュール間のインタフェースを明確にする。いわばモジュールは，ある機能を実現するソフトウェアの部品であり，ソフトウェアユニットとも呼ばれる図表2。



図表2　階層化による分割のイメージ

モジュール単位に分割すると，モジュールごとに設計・作成・テストができるので作業を分担し，並行して進めることができる。また，問題が起こってもモジュール単位でデバッグ（218ページ）できる。1つのモジュールを複数箇所で再利用することも可能になり、開発の効率化につながる。

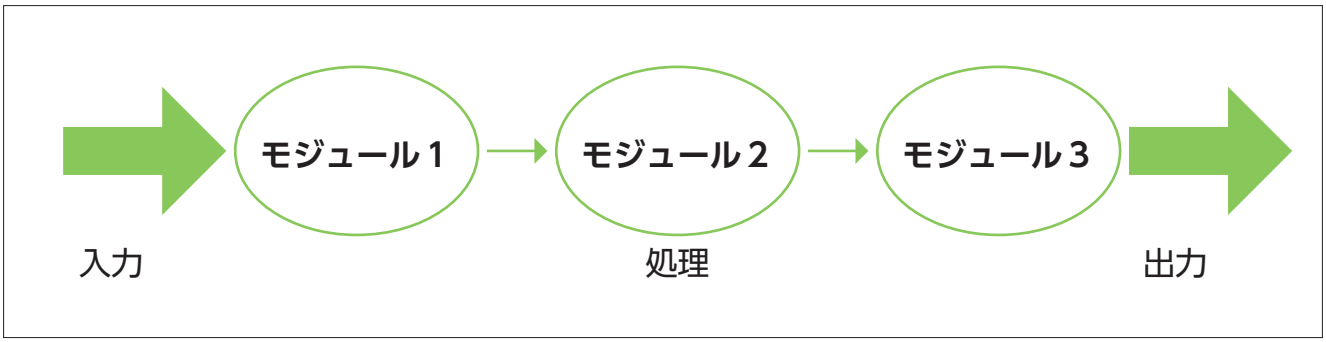
3 モジュール分割技法

システム開発ではプログラムをモジュールに分割していくが，見通しを持たずにモジュールに分割すると，開発が難しくなることが予想される。現在ではモジュールの分割技法がいくつか開発されている。

一般にモジュール分割は，次のような手順で行われる。

❶最上位のモジュールの定義
❷モジュールの機能分析
❸モジュールの分割技法の選択
❹モジュールの分割
❺インタフェースの定義
❻分割すべきほかのモジュールの検討

また，モジュール分割技法には，データの流れに着目したプロセス中心設計と，データ構造に着目したデータ中心設計がある図表3。



図表 3　データの流れに着目したモジュール分割のイメージ

演習 1

EXERCISE

図書館の本の貸出システムの「貸し出す本があるか検索するプログラム」についてモジュール分割をしてみましょう。

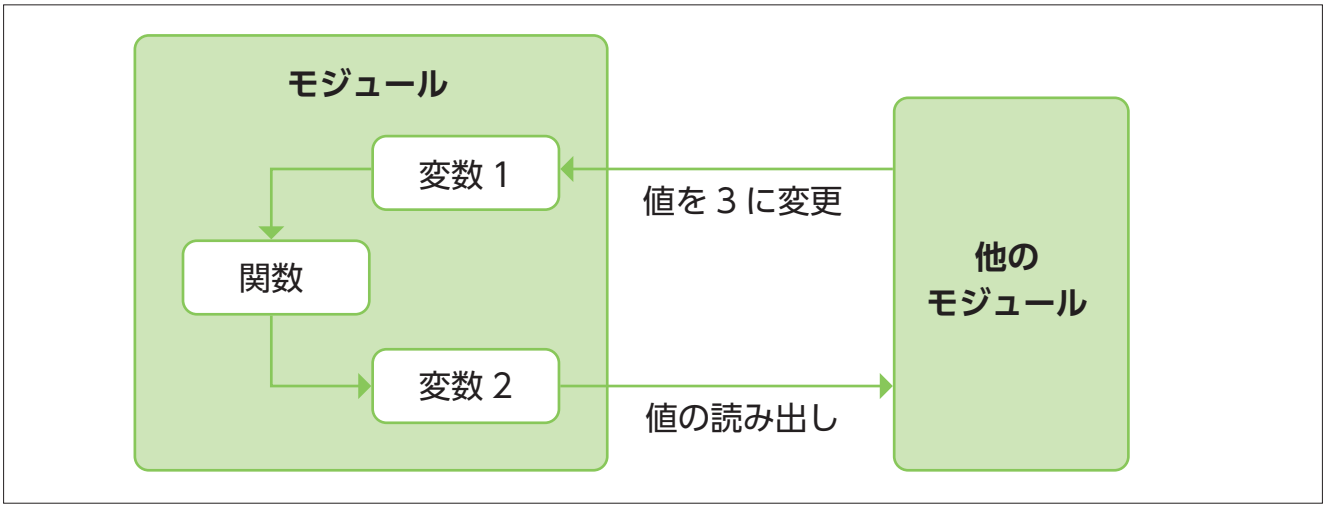
4 モジュール結合とカプセル化

モジュール分割においては，モジュール同士が適切に分割されていないと，切り離して作業を進めることができない。以下にモジュール分割の留意点を示す。

- 一つのモジュールの大きさが適切になるようにする。
- 一つのモジュールから呼び出す他のモジュールの数に制限を付ける。
- モジュールからモジュールを呼び出す階層構造があまり深くならないようにする。
- モジュール間のインタフェースが単純になるようにする。

モジュールとモジュールの結び付きをできるだけ弱くし，お互いの独立性を高めることにより分業での開発作業が行いやすくなり，モジュールの再利用も可能となるなど開発上のメリットが大きい。

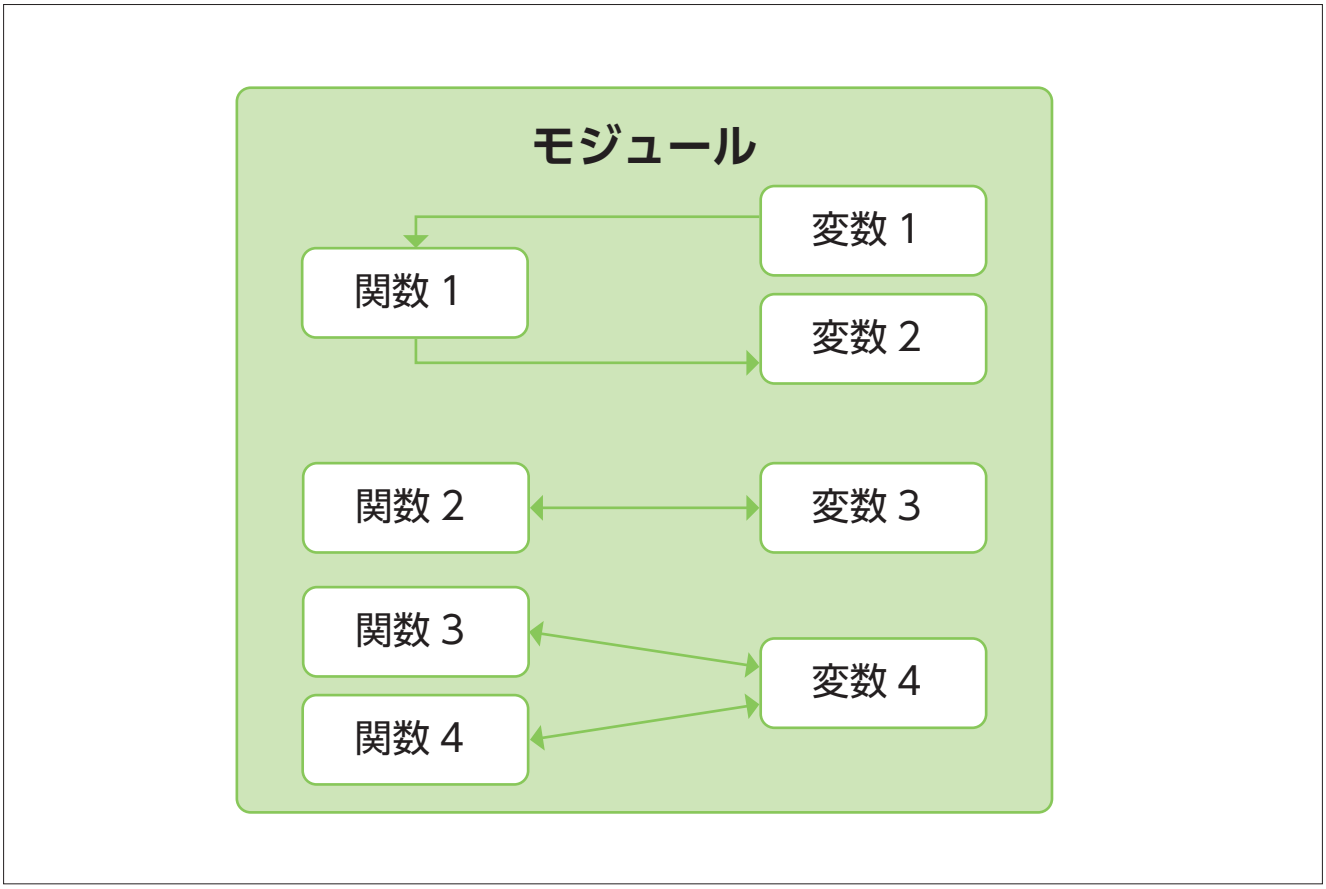
図表 4 に，独立性の低いモジュールの例を示す。モジュール内部の変数(データ)が他のモジュールから直接読み書きできてしまうため，モジュールを正しく作ったとしても他のモジュールの不具合などでデータの整合性が壊れてしまう可能性がある。



図表 4　独立性の低いモジュール

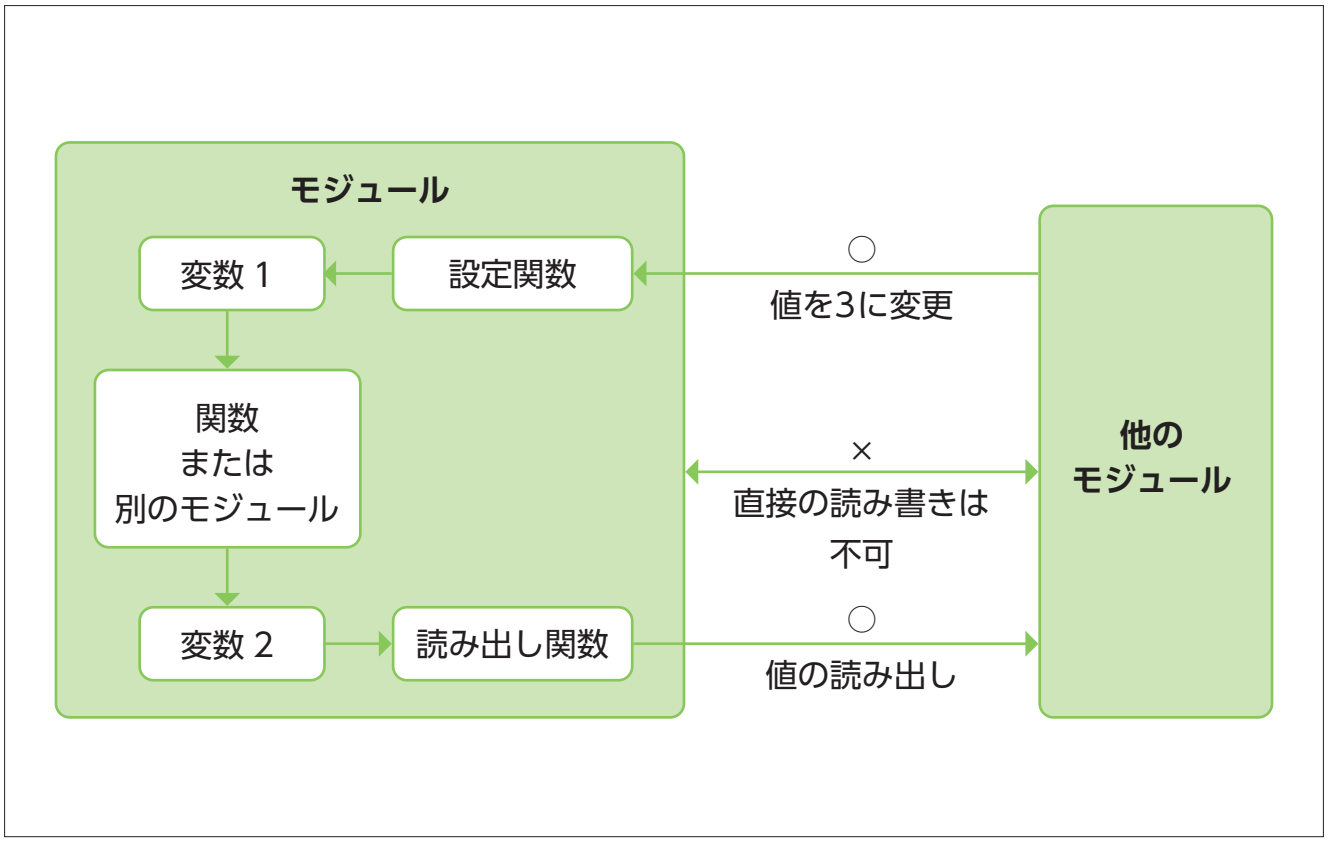
モジュールの独立性が高い状態では，モジュールの内部は他のモジュールから見ることはできず，モジュールが公開したインタフェースを通してのみ，そのモジュールにアクセスして利用することができる。このような独立性の高い状態にすることは，オブジェクト指向という設計方式ではカプセル化と呼ばれている。以下の説明では，モジュールの独立性を高める方法として，オブジェクト指向のカプセル化の考え方を紹介する。オブジェクト指向は広く使われており，「情報I」の教員研修資料で扱った言語ではPython，JavaScript，Swift，ドリトルが，その他にもC#，Ruby，Javaなど多くの言語がオブジェクト指向の考え方でプログラムを記述する。

通常のプログラムでは，プログラムを「フローチャートや，反復や条件分岐などの制御構造などの処理の流れ」で考えるが，オブジェクト指向ではプログラムを「処理の流れとデータを1つのまとまりにしたオブジェクト」で考える。そしてプログラム全体をオブジェクトの関係としてとらえて設計し，開発を進めていく。オブジェクトで見たときのモジュールは，図表5のように，データである変数と，手続きである関数で定義される。なお，オブジェクト指向では数値や文字列のような基本的なデータもオブジェクトとして扱うことがあるが，今回の説明では「本の登録モジュール」「本の貸出モジュール」のようなプログラムの機能のまとまりであるモジュールを，大きなオブジェクトと見なして説明している。



図表5　オブジェクトとしてのモジュールのイメージ

モジュールは，変数(データ)とそれに関する手続き(関数)を1つにまとめたものであり，他のプログラムからは，内部のデータを直接見たり，値の変更を行ったりすることができない。このようにモジュールやオブジェクトの内部情報を隠ぺい化することをカプセル化という図表6。カプセル化によって，外部からデータを直接アクセスすることを禁止しているが，データを読み書きするための関数を用意することができる。これはモジュールのインタフェースに相当する。図表6のモジュールでは，変数に値を書くための設定用の関数と，変数の値を参照するための読み出し用の関数を用意している。



図表 6　カプセル化のイメージ

カプセル化の効果として，モジュールの内部データ構造やメソッドの実装を変更しても，その影響をほかのモジュールに及ぼしにくくなることがある。カプセル化をモジュールの設計に利用した場合，他のモジュールがデータの値に直接アクセスすることがないので，独立性が高まり，モジュールを再利用しやすくなるなどメリットが大きい。

演習 2

EXERCISE

先の演習で図書館の本の貸出システムの「貸し出す本があるかを検索するプログラム」についてモジュール分割を行いましたが，これらのモジュールをカプセル化できるかどうかを検討し，可能な限りカプセル化を行いましょう。

【参考文献・参考サイト】

●「栢木先生の基本情報処理技術者教室」栢木厚 著　技術評論社(2019)
●「IT 専科」http://www.itsenka.com

学習活動と展開

学習活動の目的

- プログラムをモジュールに分割する必要性を理解させる。
- プログラムを適切にモジュールに分割する力を身に付ける。

学習活動とそれを促す問い

	問　い	学習活動
展開 1	「図書館の本の貸出システム」においてどのようなプログラムに分割できるか考えよう。	「図書館の本の貸出システム」を例にどのようなプログラムに分割できるか，要件定義を踏まえて考えさせる。
展開 2	「貸し出す本があるか検索するプログラム」をモジュール分割しよう。	「貸し出す本があるか検索するプログラム」に対してモジュール分割を行い，他の人の例と比較する。
展開 3	分割したモジュールの中でカプセル化できるものがないか考えよう。	分割したモジュールの中でカプセル化できるものがないか考えることでモジュールの独立性を考える。

展開 1

問　い

- 「図書館の本の貸出システム」においてどのような機能が必要かを確認しよう。
- 「図書館の本の貸出システム」においてどのようなプログラムが必要かを考えよう。

▼

学習活動

- 要件定義を踏まえて必要な機能を確認する。
- 機能から逆算して必要となるプログラムを考える。

▼

指導上の留意点

要件定義からはみ出ないように必要な機能は厳選する。

展開 2

問　い

- 「貸し出す本があるか検索するプログラム」についてモジュール分割を考えよう。
- 他の人はどのように分割しているか比較しよう。

学習活動

- 「貸し出す本があるか検索するプログラム」のモジュール分割を検討する。
- 他の人はどのように分割しているか比較する。

指導上の留意点

- 入力，処理，出力の３つに分けて始めると取り組みやすいことを伝える。
- 他の人と比較することで必要ではないモジュールがないかを検証する。

展開 3

問　い

分割したモジュールの中からカプセル化できないか考えよう。

学習活動

- モジュールの中からカプセル化できるものがないかを考える。
- カプセル化する前と後でモジュールの独立性が高まったことを確認する。

指導上の留意点

データのアクセスをどのように行っているかでカプセル化ができるかを検証させる。

まとめ

まとめ

- モジュールの分割を通して，この後の作業が分業可能になることを伝える。
- モジュールの独立性はとても大事で，独立性が低いと，分業が難しくなることを伝える。
- 要求定義がしっかりなされていないと，モジュールの分割も困難であることを認識させる。

学習

23 分割したシステムの制作とテスト

研修内容

研修の目的

- 単体プログラムの制作に適したプログラミング言語を選ぶ方法について理解する。
- 単体プログラムを作成する方法について理解する。
- 単体プログラムをテストする方法について理解する。
- 単体プログラムをデバッグする方法について理解する。
- 単体プログラムを作成する一連の方法について授業することができる。

この学習項目で使用するプログラミング言語は Python です。

1 プログラミング言語の選択

プログラミング言語には，データの扱いに適したもの，機械の制御に適したもの，グラフィックスに強いものなどの特性があり，作ろうとするものに合ったプログラミング言語を選べば，効率よく目的を達成することができる。作成しようとするものに合わせて適切なプログラミング言語を選択することが大切である。図表 1 に代表的なプログラミング言語の例を示す。

作成したプログラムを実行する方式には，コンピュータが直接実行可能な形式を生成するコンパイル方式と，記述したプログラムを実行時に逐次コンピュータが直接実行可能な形式に翻訳するインタプリタ方式がある。コンパイル方式は一般に動作が速い利点があるが，開発時にはプログラムの動作を確認するたびにコンパイル作業を行う必要がある。C，C#，Java，Swiftはコンパイルを行うが，後述するIDE（統合開発環境）を使用することで対話的な開発が可能である。インタプリタ方式はコンパイル作業が不要だが，動作させるコンピュータにインタプリタのプログラムが入っている必要がある。 JavaScript，PHP，Python，R，Rubyはインタプリタ方式に該当する。作成しようとするシステムの性質により，また学習する生徒の特性に合ったプログラミング言語を選択する必要があるだろう。

また，プログラムを作成するだけではなく，正しく動作するかどうかのテストも行うために，命令補完機能を備えたソースコードエディタや，デバッガを備えているIDEで開発に取り組むことも効率的である。その際に，インストール不要となっているオンライン上のエディタや開発環境も活用したい。

言語	用途の例
C	OS や組込機器のプログラムなど
C#	Windows のアプリケーション開発など
Java	情報システム開発，Android のアプリケーション開発など
JavaScript	Web 画面で動作するプログラムなど
PHP	Web サーバで動作するプログラムなど
Python	機械学習のプログラムなど
R	統計処理のプログラムなど
Ruby	Web サーバで動作するプログラムなど
Swift	Mac や iOS のアプリケーション開発など

図表1　プログラミング言語の例

演習・1

EXERCISE

作成したいシステムに向いているプログラミング言語を選択してみましょう。

・iOS でも Android でも動作するアプリを開発するためにどのようなプログラミング言語を選択するとよいか。
・Windows OS を前提とした図書館の図書管理システムを開発するためにはどのようなプログラミング言語を選択するとよいか。

2 単体プログラムの作成

単体プログラムの作成のゴールは，機能ごとに分割されたシステムをプログラミング言語で表現することである。単体プログラムの多くは，入力に対して処理を行い，その結果を出力するものが多い。まず，入力と出力のデータの形式を定めることが必要である。

単体プログラムのデータの入力及び出力としては次のものが一般的である。

・関数への引数や戻り値として渡す
・ネットワーク通信のメッセージとして渡す

ここでは，インターネットで公開されている関数(Web API)で用意されている機能を利用して，データを取得する簡単なプログラムを作成して実行するまでの処理について学んでいく。

例えば，キーワードを送ることにより，関連した図書のデータを返すWeb APIを考える。作成するプログラムは，入力されたデータをWeb APIの仕様に定められた形式で送り，Web APIの仕様に定められた出力形式の図書データを受け取ることになる。図表2に，実習で使用できる図書検索用Web APIの仕様を示す。このAPIを利用して図書の検索を行う単体プログラムを作成する。

変数	説明
url	図書検索を行うサーバの URL
以下は，図表3 の params で指定するパラメータ	
token	Web API を利用するためのアクセスキー
table	使用するテーブルの名前
column	検索する列の名前
value	検索する文字列
format	出力形式に json を指定する

図表 2　使用する図書検索 Web API の仕様

ここで得られた書籍の情報を，単体プログラムの出力として表示する処理を考える。今回のサンプルでは，図書情報として書名，著者名，出版社名，ISBNコードが検索できるようになっている。図表3に「Python入門」というタイトルで検索するPython言語のプログラムの例を示す。

```
import urllib.request, urllib.parse, json
def get(url,params):
    p=urllib.parse.urlencode(params)
    url=url+"?"+p
    with urllib.request.urlopen(url) as res:
        return json.loads(res.read().decode("utf-8"))

# パラメータを設定して図書を検索する
params = {
        "token": "test",
        "table": "book",
        "column": "title",
        "value": "Python 入門 ",
        "format":"json"
}
url = "https://api.eplang.jp/mext2/search"
data = get(url,params)
print(data)
```

図表 3　Web API を利用した図書検索プログラム

プログラムを動かしながら試すためには，公式サイトからダウンロードして手元のPCにPythonをインストールすることも可能だが，Googleのアカウントがあれば，Google Chromeなどのブラウザから Google Colaboratoryで実行することができる。アカウントがなくても，次のURLに今回のサンプルプログラムを実行できるページを用意した。

(1)次のURLにアクセスする。(※)

http://bit.ly/39dK9qc

(2)「Open in playground」(PLAYGROUNDで開く)をクリックする。

(3)「Copy to Drive」(ドライブにコピー)をクリックする。

(4)サンプルプログラムの左側にある「[ ]」にマウスカーソルを置くと▶に変わる。クリックするとプログラムが実行されて結果が表示される。プログラムを編集することも可能である。

(※)短縮していないURL

https://colab.research.google.com/drive/1r26wcPi3jcVfe0BhzbyTj0-oGhi8AgBC

演習 2

EXERCISE

著者名で検索して，該当する書籍を一覧で表示するプログラムを作成しましょう。

3 単体テスト

大規模なプログラムを作成する場合や，業務や研究としてプログラムを作成する場合には，他者が決めた仕様通りに動くプログラムを作成することになる。作成したプログラムが内部設計の仕様を満たしているかどうか，また正常に機能するかどうかを確かめるテストを単体テストという。単体テストを繰り返してプログラムのバグを発見し，単体プログラムを修正していく。

ここでは代表的なテストの考え方として，ホワイトボックステストとブラックボックステストについて紹介する。ホワイトボックステストでは，「ソースコードのこの条件はテストしているか」「ソースコードのこの条件分岐の実行は両方ともテストされているか」のように，プログラムのソースコードを見ながらテストを行う。ブラックボックステストでは，ソースコードを見ずに仕様書の機能を満たしているかを考えながらテストを行う。

(1)ホワイトボックステスト

単体テストでは，ホワイトボックステストの考え方を用いることが多い。ホワイトボックステストは，ソフトウェアの内部構造や関数の内容を参照して，プログラムの論理的な構造が正しいかを確認するテ

ストである。

ホワイトボックステストの考え方として,「全ての命令を少なくとも1回は実行する」命令網羅や,「全ての条件分岐を少なくとも1回は通る」分岐網羅／条件網羅などの考え方がある。

次のような処理を行うプログラムについて考えてみよう。

例えば，100点満点の成績を与えたときに，図表4のような3段階の評価を返す機能を持つ単体プログラムhyouka（ ）を作成する。

評価 grade	A	B	C
成績 score	100 ～ 80	79 ～ 60	59 ～ 0

図表4　成績判定プログラムの判定基準

この機能が正しく実装されているかどうかを調べるために,「全ての命令が必ず1回以上テストで実行されているか」を確認する命令網羅の考え方からテストを行う場合は，図表5のようなテストが全て成功することを確認すればよい。

```
1   # 成績判定のプログラム
2   def hyouka(score: int):
3       if score < 60 and score >= 0:
4           grade = 'C'
5       elif score < 80  and score >= 60:
6           grade = 'B'
7       elif score <= 100  and score >= 80:
8           grade = 'A'
9       return  grade
10
11  # ここからは単体テストのためのコード
12  if hyouka(59) == 'C':
13      print("59 → C:success!")
14  else:
15      print("59 == 'C':failed!")
16  if hyouka(79) == 'B':
17      print("79 → B:success!")
18  else:
19      print("79 → B:failed!")
20  if hyouka(100) == 'A':
21      print("100 → A:success!")
22  else:
23      print("100 → A:failed!")
```

図表5　成績判定プログラムのテストコード

テストはキーボードから条件を入力して結果を確認することも可能だが，プログラムの開発ではソースコードが変更されるたびに確認する必要があることから，図表5のようなテストプログラムを書いて残しておくことが重要である。

図表5では,「全ての命令が必ず1回以上実行される」テストコードを作成した。テストの成績であるため，入力値は0から100までの数値が期待されるが，負の値や100より大きい値が入力された場合は，どの条件判定にも当てはまらず，関数の最後まで到達してしまうため，テストが行われないことになる。命令網羅に加えて分岐網羅の考え方を取り入れることで,「全ての分岐を通らない場合」の戻り値を確認するテストも設計に入ることになり，仕様の曖昧な点やプログラム作成時に検討していない条件についても確認することができる。

(2) ブラックボックステスト

ブラックボックステストでは，ソースコードを見ずに仕様書の機能を満たしているかを考えながらテストを行う。ここでは，同値分割と境界値分割の考え方を紹介する。

❶同値分割

同値分割とは，入力データをいくつかの領域に分割し，分割された領域にある値は，どれをテストしても同じという考え方である。分割された領域のことを同値クラスという。例えば貸出期限の日付に関する処理を考える。

同値クラス	テストする値の候補
貸出の当日	6月1日
貸出から2週間以内である	6月7日　6月10日
貸出から2週間を越えている	6月15日　6月30日

図表6　図書館システムの貸出期限の例

図書館の貸出システムにおいて，貸し出した本は2週間以内に返却するものとする。図表6に，6月1日に図書館システムを実行したときの，貸出期限ごとの処理をテストするための値の例を示す。

この例では，貸出から2週間以内の処理をテストする日付として6月7日と6月10日の2つの候補があるが，これらは2週間以内と同値クラスにあたるため，どちらか1つだけをテストすればよいことになる。

❷境界値分割

境界値とは，同値クラスの上限や下限，またそのすぐ隣の値のことである。条件の値の範囲がある場合，その範囲の両端やそれぞれを越えた値が有効であるか無効であるかをテストする。

先ほどの貸出期限の例を取ると，境界値分割でテストするケースは，貸出日からちょうど2週間後(14日後)の日，とその翌日(15日後)のとき，それぞれの境界値の値で正しい処理が行われるかどうかをテストすることになる。うるう年の日付の処理や，夏休みなどの特例の長期貸出などについての例外処理も入る可能性がある。

演習3

EXERCISE

図書館の貸出システムでは貸出希望の図書を1冊ずつ貸出判定していきます。貸出冊数の上限5冊に達した場合，または既に借りている書籍の延滞がある場合には，新しく借りることができないものとします。この判定を行うプログラムを作成する場合，分岐を網羅するテストとしてどのような入力値の組み合わせを設定すればよいでしょうか。

4 デバッグ

テストを行っていくうちに，「仕様通りにプログラムが動作しないが，原因がすぐには分からない」というようなバグと呼ばれる不具合に遭遇することが多い。このバグを取り除くことをデバッグという。

デバッグの工程は大きく次の2つに分けられる。

・エラーの性質と場所を特定すること
・エラーを修正すること

エラーの場所を特定するためには，原始的な手法として，次のようなものがある。

・プログラム中に表示命令を散りばめて，デバッグ用メッセージを出力する
・デバッガでの情報表示

前者の例は，分岐の処理でどの分岐をたどっているかを確認するために，ログやコンソール画面に分岐のどのルートを通った動作となっているかを表示させるように命令を追加するなどである。しかし，エラーを見つける前にソースコードに手を加えてしまうので，プログラム実行時のタイミングが原因で発生するエラーなどであれば挙動が変化してしまうケースもある。

後者では，デバッガでプログラム中の変数をトレースしたり，プログラムの任意の箇所で一時停止させるブレークポイントと呼ばれる処理を追加したりできるが，変数の値の組み合わせなどによって正しく動作しないケースなどは，突き止めるために何度もデバッガを操作しなければならない。

ソースコードの構造とテストデータ，テスト結果によってエラーの性質とエラーが起きている場所を突き止めていくための手法として次のような手法が挙げられる。

(1)帰納法

帰納的にエラーを見つけるためには，次のように調査を行う。

・プログラムの正常終了，異常終了の双方のデータ

を集める。
・エラーのパターン，プログラムの矛盾を見つけるためにデータを構造化する。
・エラーが発生する仮説を立てる。

(2) 逆戻りで考える

正しくない結果を返しているソースコードの箇所から，逆順にプログラムをたどっていく。それぞれの箇所で，プログラムの変数が持たなければならなかった値は何で，実際にはどうなっているのかを推定しながらソースコードを読んでいく。

(3) テストケースの利用

これまでのテストは仕様通りに動作するかどうかを調べるためのテストであったが，バグがある疑いが強くなった場合は，エラーがどこで発生しているかを突き止めるためのテストケースを設計するという手法も考えられる。

演習 4

EXERCISE

10 名の 10 点満点の成績について，平均と中央値，最大値を表示するロジックを作成しました。しかし，統計モジュールで確認したところ値が異なります。元のプログラムのエラー箇所を特定するためには，どのように絞り込めばよいでしょうか。

```
1  import statistics
2  #  成績分析のプログラム
3  def def_mean(ary):
4      sum=0
5      n=len(ary)
6      for num in ary:
7          sum=num             # sum+=num の間違い
8      return sum / n
9  def def_median(ary):
10     n=len(ary)
11     index1=int(n/2)        # index1=int(n/2)-1 の間違い
12     if n%2==0:
13         index2=index1+1
14     else:
15         index2=index1
16     return (ary[index1]+ary[index2])/2
17 def def_max(ary):
18     x=ary[0]
19     for num in ary:
20         if x > num:     # if x < num: の間違い
21             x=num
22     return x
23
24 # ここからは単体テストのためのコード
25 def test_mean(ary):
26     return def_mean(ary) == statistics.mean(ary)
27 def test_median(ary):
28     return def_median(ary) == statistics.median(ary)
29 def test_max(ary):
30     return def_max(ary) == max(ary)
31 data = [2,3,3,5,6,6,7,8,8,9]
32 print(test_mean(data))
33 print(test_median(data))
34 print(test_max(data))
```

【参考文献】
●「はじめよう！ システム設計　要件定義のその後に」羽生章洋 著　技術評論社(2018)
●「知識ゼロから学ぶ ソフトウェアテスト［改訂版］ アジャイル・クラウド時代のソフトウェアテスト」高橋寿一 著　翔泳社(2013)
●「ソフトウェア・テストの技法 第 2 版」Glenford J.Myers 著　長尾真 監訳　松尾正信 訳　近代科学社(2006)

学習活動と展開

学習活動の目的

分割したシステムの一部である単体プログラムを作成し，作成したプログラムについて単体テストとデバッグを行えるようになる。

学習活動とそれを促す問い

	問　い	学習活動
展開 1	作成しようとするシステムについて，単体プログラムの作成に必要な情報は何か。	単体プログラムに与える入力データは，どのようなデータで，どのような形でプログラムに入力されるのかを設計から読み解く。 単体プログラムの結果として，どのようなデータをどのような形で出力すればよいかについて考える。
展開 2	図書館の図書管理システムの 1 つの機能について，単体プログラムを作成してみよう。	単体テストの実施計画を考える。 内部設計書から，テストで確認すべき項目とテストに使用するテストデータを検討し，テスト計画として文書にまとめる。 テストプログラムを書いて，テストを実行し，実行結果についてまとめる。
展開 3	作成した単体プログラムに対して，入力に対する出力結果が正しいか単体テストを行ってみよう。	作成した単体プログラムが設計通りに作成されているかを調べる単体テストを実施していく。

展開 1

問　い

作成しようとするシステムについて，単体プログラムの作成に必要な情報は何か。

学習活動

- 単体プログラムに与える入力データは，どのようなデータで，どのような形でプログラムに入力されるのかを設計から読み解く。
- 単体プログラムの結果として，どのようなデータをどのような形で出力すればよいかについて考える。

指導上の留意点

全員が同じ機能を持つ基本的なプログラムを作成するのか，チームに分かれて生徒一人一人が異なるプログラムを作成するのかは，生徒の技術レベルに応じて検討すること。

展開 2

問　い

図書館の図書管理システムの 1 つの機能について，単体プログラムを作成してみよう。

学習活動

- 図書管理システムの機能を 1 つ選び，その処理を実装するプログラムを作成していく。
- 入力データから適切な出力結果を得られるようにプログラムの処理を書いていく。
- 入力における漏れがないか，正しい論理構造となっているかを意識しながらプログラムを作成する。

指導上の留意点

- 全員が同じ機能を持つ基本的なプログラムを作成するのか，チームに分かれて生徒一人一人が異なるプログラムを作成するのかは，生徒の技術レベルに応じて検討する。
- データを扱うプログラムを作成する際は，生徒が入力に時間をかけないようにあらかじめ教員側で準備しておくことも考えられる。

展開 3

問　い

作成した単体プログラムに対して，入力に対する出力結果が正しいか単体テストを行ってみよう。

学習活動

- 単体テストの実施計画を考える。
- 内部設計書から，テストで確認すべき項目とテストに使用するテストデータを検討し，テスト計画として文書にまとめる。
- テストプログラムを書いて，テストを実行し，実行結果についてまとめる。

指導上の留意点

テスト項目について完全に網羅するテストを一度に設計することは難しいため，思い付きやすいテストから記述するように指導する。

まとめ

まとめ

- 情報システムのような大きなシステムを制作する段階の 1 つである，単体プログラムの作成について，理解させる。
- 設計の仕様通りに単体プログラムを作成できたことを検証するための単体テストについて理解させる。
- 単体テストで発見された不具合やバグは，デバッグの考え方を利用して取り除いていく。

学習

24 分割したシステムの結合とテスト

研修内容

研修の目的

- 分割して制作したプログラムを結合して確認する工程について理解する。
- モジュールの結合とテストについて理解する。
- 結合テストの概念について理解する。
- 総合テストで利用するシナリオ作成の技法について理解する。
- 分割したシステムの結合とテストについて生徒に考えさせることができる。

この学習項目で使用するプログラミング言語は Python です。

1 結合テスト

学習23で学んだように，「点数から成績を判定する」「特定の書籍を検索する」のような特定の処理を行う関数レベルのテストについては，単体テストを行うことが有用であった。これらを組み合わせて「画面から書名を入力し，その書籍を検索し，表の形で表示する」といった複数の関数から構成される，まとまった処理を行うモジュールのテストについては結合テストを行う必要がある。

結合テストを行うためには，モジュールを構成する個々の関数のようなサブモジュールについて単体テストを行った後に，それらの関数を組み合わせて動作するモジュールの形に結合する。結合テストでは，プログラムの構造によるテストではなく，仕様書にある通りの動作をするかどうか仕様書に沿ったテストを設計して実行していく。

図表 1 に，図書を検索するモジュールの例を示す。このモジュールには，あらかじめ本教材用に用意された関数get（ ）の他に，3個の関数がユーザーにより定義されている。検索語を入力する関数「title_input（ ）」は，画面から検索語を入力し，その文字列を返す。図書を検索する関数「title_search（ ）」は，Web APIを使用して検索語に当てはまる図書を検索し，結果を返す。結果を表示する関数「disp（ ）」は，JSON形式のデータを受け取り，見やすい形で画面に表示する。末尾の3行は全体を実行するメインの処理である。これらの関数は個別に単体テストを行っており，その後，図表 1 のように一つのモジュールとして利用できるように結合した。

なお，一般的に大規模なシステム開発では，モジュールは単体テストの対象となり，複数のモジュールを組み合わせて結合テストが行われる。以下の一般的な説明ではこの位置付けでモジュールを扱っている。高等学校の授業では小規模なシステムを扱うため，関数レベルのサブモジュールに単体テストを行い，それらを結合してモジュールを作り結合テストを行うと考えると分かりやすい。

(1)結合テスト

結合テストはモジュールを結合して動かし，全体として正しい機能を備えていることを検証するテストである。ここで結合するモジュールは単体テストが終わっていることが前提である。結合テストで明らかになる不具合は，主にモジュール間のインタフェースの不備である。呼び出すときに引数の対応が間違っている，渡す引数に正しい値がセットされていない，などのエラーが想定される。

```
import urllib.request, urllib.parse, json
def get(url,params):
    p=urllib.parse.urlencode(params)
    url=url+"?"+p
    with urllib.request.urlopen(url) as res:
        return json.loads(res.read().decode("utf-8"))

# 検索語入力関数
def title_input():
    print(" 検索するタイトルを入力してください ")
    title=input()
    return title

# 図書検索関数
def title_search(title):
    params = {
            "token": "test",
            "table": "book",
            "format":"json",
            "column": "title",
            "value": title
    }
    url = "https://api.eplang.jp/mext2/search"
    data = get(url,params)
    return data

# 結果表示関数
def disp(data):
    for record in data:
        print(record["title"]+"=>"+record["price"]+"円")

# 全体の実行
title=title_input()
result=title_search(title)
disp(result)
```

図表 1　図書を検索するモジュールの例

モジュールを分割するときに，各モジュールが持つ機能とモジュール間のインタフェースを設計する。結合テストでは，その設計の不備や，単体プログラムのバグを明らかにできる。

ここで，他のモジュールを呼び出すものを上位モジュール，他のモジュールから呼び出されるものを下位モジュールという。

上位モジュールから順に結合していく方法をトップダウンテストといい，呼び出される下位モジュールが完成していない場合は，スタブと呼ばれるダミーの下位モジュールを用意してテストを行う。

下位モジュールから順に結合していく方法をボトムアップテストといい，上位モジュールが完成していない場合は，テストドライバと呼ばれるダミーの上位モジュールを用意してテストを行う。スタブを利用するときには，呼び出しや受け取りのパラメータのチェックや正常値・異常値の判定など，呼び出し機能に関わる必要項目だけのテストとなることが多い。

(2) スタブとテストドライバ

スタブは，何も処理せずに特定の結果を呼び出したテスト対象の上位モジュールに返すことが多い。上位モジュールから渡された引数などの値を表示・印刷する機能があると便利である。

図表2に，図書検索関数「title_search()」のスタブの例を示す。これは，どのようなタイトルを引数で受け取った場合でも，常に同じ図書のデータを返す。実際の図書検索関数が完成していなくても，代わりにこのスタブを上位モジュールから呼び出すことで，モジュール全体の動作を確認することができる。

テストドライバは，適切な引数を設定してテスト対象の下位モジュールを呼び出すことが多く，一般的にテスト対象の下位モジュールが返したリターンコードや引数の値を，表示・印刷する機能がある。これを検証することで対象の下位モジュールのテストができる。

例えば，図表1では末尾の3行がテストドライバの役割を果たしている。この3行は，検索語の入力，検索語による図書の検索，検索結果の表示という3つのモジュールを結合するテストを行うことができる。

スタブやテストドライバは個別に作成する場合もあるが，汎用的なスタブ，テストドライバを生成できるパッケージも存在する。

```
1  def title_search(title):
2      data = json.loads('[{"title":"Pythonの教科書","author":"クジラ飛行机
   ","publisher":"マイナビ出版","year":"2016","ISBN":"--","price":"2580"}]')
3      return data
```

図表2　図書検索関数「title_search（）」のスタブの例（2行目は長い1行）

演習1

EXERCISE

蔵書検索で該当する書籍のうち，1つを選んで貸し出し予約に追加する機能を実現します。関数（API）を利用した蔵書検索機能と予約管理機能を結合させてみましょう。

結合テストの方式	特徴	欠点
トップダウンテスト	・インタフェースエラーなど重大な欠陥を早期に検出できる。 ・重要度の高い上位モジュールを繰り返しテストすることになり，全体の信頼性があがる。 ・テストドライバを作成する必要がない。	・作業の分散が難しい。 ・スタブを作成する必要がある。
ボトムアップテスト	・分担しての並行作業が容易であり，早期に多くのモジュールをテストできる。 ・欠陥の影響が大きい共通モジュールの品質を先行して高めることができる。 ・スタブを作成する必要がない。	・重要な欠陥が後になって検出される。 ・テストドライバを作成する必要がある。

図表3　結合テストの方式

なお，実際のシステム開発においては，図表3のトップダウンテストとボトムアップテストを同時に進めていくことが一般的でありサンドイッチテストと呼ばれる。また，比較的小規模なプログラムでは全てのモジュールを一度に連結してテストすることも行われビッグバンテストと呼ばれる。

(3) ユースケーステストによる機能テスト

システムを，開発側からの視点の他に，利用する側の視点でテストすることも有用である。利用者の使用

事例であるユースケースを列挙し，図書検索機能であれば「どの項目を(例えば書名)」「どのキーワードで(例えばPython入門)」「検索件数は(例えば10件表示されたら多すぎるかどうか)」などを利用する視点から想定しテストを行う。図表4にユースケースの検討例を示す。

予約機能
誰が
どの書籍を
いつ
何人待ちか

図書検索機能
どの項目を
どのキーワードで
検索件数は

図表4　ユースケーステストの検討例

2 総合テスト

システムの制作では，プログラムの作成と単体テスト，結合テストを経てから，システム全体として必要とされていた機能や仕様を満たすかどうかの総合テストを行うことになる。このためには，実際にシステムを利用する場面を想定した試験手順書を作成し，一連の流れに沿ってシステムを操作したときのシステム全体の動作を確認しなければならない。また，利用者の立場になって操作する場合でも，通常の利用ではなかなか出てこない場面の動作確認や異常時の動作確認なども総合テストの中で行うことになる。

システム開発では，単体テストと結合テストが終了した後は，総合テストと呼ばれる段階を経てシステム完成となる。総合テストは，機能テストと非機能テストに大きく分けられる。

(1)機能テスト

システム開発の機能テストでは，要求仕様に規定されている機能がシステムに盛り込まれているかどうかを確認するテストである。しかし，総合テストでは，実際にユーザーが利用するのと同じように，手作業で入力したり操作したりして試験するため，全ての機能の動作を網羅するには作業工数が多くなってしまう。各機能の詳細については，単体テストでしっかりと確認し，機能テストの段階では利用シナリオに沿ったテストを実施していく。

図表5に，図書を登録するモジュールの例を示す。総合テストの機能テストでは，複数のモジュールを組み合わせたときのシステム全体の動作を確認する。この例では，既に登録されている図書データに加えて，図表5のモジュールで登録された図書データが，図書検索モジュールから正しく検索できることを確認するテストを行う必要がある。

```
import urllib.request, urllib.parse, json
def get(url,params):
    p=urllib.parse.urlencode(params)
    url=url+"?"+p
    urllib.request.urlopen(url)

# パラメータを設定して図書を登録する
params = {
        "token": "test",
        "table": "book",
        "title": "Pythonの教科書",
        "author": "クジラ飛行机",
        "publisher": "マイナビ出版",
        "year": 2016
}
url = "https://api.eplang.jp/mext2/write"
get(url,params)
```

図表5　図書の登録モジュールの例

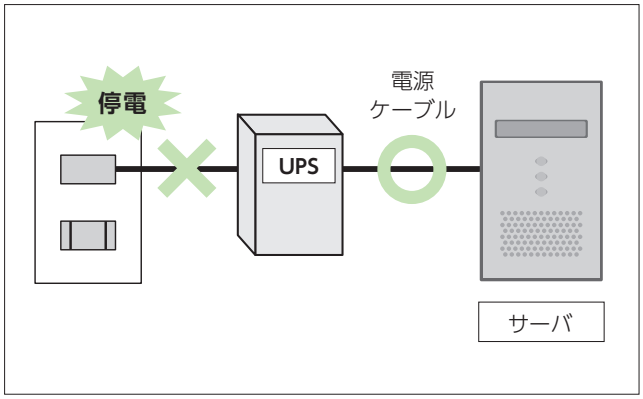
(2) 非機能テスト

システム開発の非機能テストでは，要求仕様で決められている場合には，性能や，ユーザーが操作をしてから結果が表示されるまでの処理時間や，負荷が高まったときでも正常に動作するかなど非機能要件と呼ばれる観点でのテストも行う。

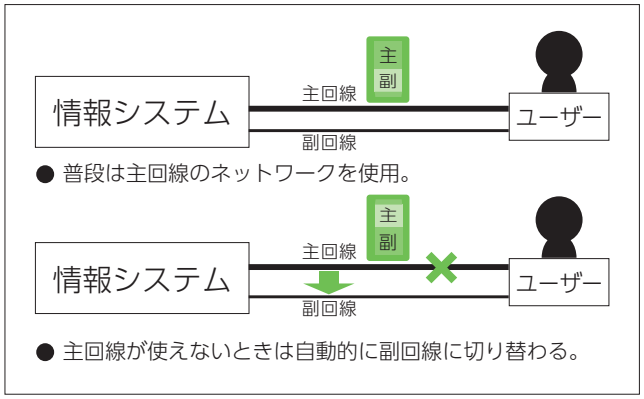
例えば，そのシステムが複数ユーザーで同時利用する性質のシステムであれば，多数のユーザーからアクセスされて負荷が高くなったときでも正常に動作するかどうかの性能テストは非常に重要になる。

他にも，ハードウェアやソフトウェア，そして災害などに起因する停電等の障害から復旧してシステムの稼働を再開できるかといったシステム回復テストなども本番稼働させる前にテストしておかなければならない。図表6に，無停電電源装置(UPS)を用意することで，停電などでシステムが停止することを防ぐ工夫の例を示す。

また，ネットワーク構成を二重化しているとき図表7なども，障害で通信に異常が発生した際，自動的にバックアップ回線に切り替わって通信を継続する仕組みが機能するかどうか，稼働前にテストしなければならない。



図表6　UPS を接続したシステムの図



図表7　ネットワークを二重化したシステムの図

演習 2

EXERCISE

これまで作成してきたシステムまたはプログラムで，動作中に強制終了などが起きてしまった場合，どのようなデータがバックアップされていれば，業務が再開できるかを考えましょう。
また，どのようなバックアップ先にどのようなやり方でバックアップを取ればよいか，考えましょう。

3 セキュリティテスト

情報システムを実際に運用する際は，開発者以外のユーザーが操作したり，他のコンピュータからアクセスできる環境に置かれたりすることがある。そのため，しっかりとしたセキュリティ対策が必要となる。システムの利用が始まってしまうと，脆弱性の改修が難しくなってしまうことがある。セキュリティの脆弱性を利用した攻撃手法には、次のようなものがある。

(1) ユーザ認証に対する攻撃

❶ブルートフォース攻撃

あるユーザIDに対して連続して認証を試す攻撃である。パスワードを変えていき何度も認証を試みる特徴があるため，一つのIDに対して一定時間内に認証失敗が連続するときには一時的に該当IDをログイン不可とするなどの対処がある。

ユーザ ID：AAA0001　パスワード：012345　ログイン失敗（1 回目）
ユーザ ID：AAA0001　パスワード：123456　ログイン失敗（2 回目）
ユーザ ID：AAA0001　パスワード：abcdef　ログイン失敗（3 回目）
ユーザ ID：AAA0001　パスワード：password　ログイン失敗（4 回目）
ユーザ ID：AAA0001　パスワード：qwerty　ログイン失敗（5 回目）　→　一時的に凍結

❷リバースブルートフォース攻撃

ブルートフォース攻撃に似ているが，典型的な弱いパスワードなどのパスワード一つに対して，次々とユーザIDを変えてログインを試す攻撃である。一つのIDに対しての失敗回数が少ないため，ブルートフォース攻撃への対策を回避する可能性が高くなる。

```
ユーザ ID：AAA0001　　パスワード：password　ログイン失敗（1 回目）
ユーザ ID：AAA0002　　パスワード：password　ログイン失敗（1 回目）
ユーザ ID：AAA0003　　パスワード：password　ログイン失敗（1 回目）
ユーザ ID：AAA0004　　パスワード：password　ログイン失敗（1 回目）
ユーザ ID：AAA0005　　パスワード：password　ログイン失敗（1 回目）
```

❸リスト型攻撃

他のサービスなどから入手したユーザIDやパスワードでログインできるかどうか試す攻撃である。ユーザIDにメールアドレスしか設定できない設計になっていると，リスト型攻撃の対象となりやすい。

(2) XSS（クロスサイトスクリプティング）

あるサイトから送られるHTMLページに，<Script>（スクリプトタグ）を追加して，そのサイトの範囲内で動的なページであるかのように振る舞う攻撃である。偽情報を表示してしまったり，Cookieとして保存された情報を取られてしまったりする被害が起こる。これはシステムに対する攻撃ではなくクライアントに対する攻撃である。

(3) CSRF（クロスサイトリクエストフォージェエリ）

Webブラウザからの情報を詐称して，Webサイトのプログラムに意図しない不正なリクエストを実行させ，サーバの情報を取得したり，コードを実行させたり，などの動作をさせる。他のサーバに書き込みなどをしたり，利用者が意図しない攻撃をしかけたりする挙動をする攻撃もある。

これら以外にも，通信を利用してメッセージのやり取りをしている場合には，通信方式が平文で暗号化されていない場合は盗聴や改ざんの可能性が考えられる。重要な通信をする場合は，SSL/TLSなどのプロトコルを利用して通信経路での暗号化を行う必要があるだろう。

演習 3

EXERCISE

ブルートフォース攻撃とリバースブルートフォース攻撃のどちらも防ぐことができるログイン認証システムには，ログイン失敗回数での制限以外に，どのようなセキュリティ対策をすればよいか考えてみましょう。

攻撃の種類	特徴	対策
ブルートフォースアタック	一つのアカウントに対して連続で複数のパスワードでログイン認証を突破しようとする	一定時間内に一つのアカウントで連続 5 回ログイン失敗した場合は，アカウントを一時凍結扱いにする
リバースブルートフォースアタック	パスワードを固定して，連続で複数のアカウントのログイン認証を突破しようとする	同一パスワードで 5 回以上の連続ログインが試みられたら脆弱なパスワードと判定して使用不可とする

演習 4

EXERCISE

システム構築に使用しているコンピュータと OS でどのようなポートを使用しているか，確認してみてください。その中に外部からのアクセスが不要なポートがあるかどうか確認しましょう。
Windows であればネットワークに接続している状況で，コマンドプロンプトから TCP の通信状態を調査する netstat コマンドを実行してみてください。

学習活動と展開

学習活動の目的

分割して作成されたプログラム同士を結合して，仕様として要求されている機能を実現できるようになる。

学習活動とそれを促す問い

	問　い	学習活動
展開 1	結合させたい単体プログラム同士の仕様から，どのような結合テストを計画すればよいか考えてみよう。	対象の単体プログラムをテストするためには，テストドライバとスタブのどちらを準備すべきかを考え，単体プログラム間の仕様を確認しながら，結合テストとして呼び出してみる。
展開 2	あるプログラムから渡されるデータを入力として，もう一つのプログラムの処理が完了するかテストしてみよう。	関係する 2 つの単体プログラムを結合して正常に動作することを確認する。
展開 3	総合テストで実施する項目を選び，実際に実行してみよう。	総合テストのうち，実施できるテストについて検討してみる。 作成したシステムのセキュリティについて，脆弱性があるかどうかを検討してみる。

展開 1

問　い

結合させたい単体プログラム同士の仕様から，どのような結合テストを計画すればよいか考えてみよう。

▼

学習活動

対象の単体プログラムをテストするためには，テストドライバとスタブのどちらを準備すべきかを考え，単体プログラム間の仕様を確認しながら，結合テストとして呼び出してみる。

▼

指導上の留意点

- 常にシステム全体を意識し，上位モジュールから結合する場合はスタブ，下位モジュールから結合する場合はテストドライバが必要なことを体験させる。
- 一度に複数のモジュールを結合させず，一つずつ結合して動作を確かめるようにする。

展開 2

問　い

あるプログラムから渡されるデータを入力として，もう一つのプログラムの処理が完了するかテストしてみよう。

学習活動

関係する 2 つのプログラムを結合して正常に動作することを確認する。

指導上の留意点

- データの受け渡しが正常に行われるよう，プログラム間のインタフェースを確認させる。
- プログラム間でどのようなデータが受け渡されたかを確認させる。
- 受け取ったデータが正常に処理された際に期待される状態を考えさせる。
- 実際に結合したプログラムの処理結果と上記を比較させる。

展開 3

問　い

総合テストで実施する項目を選び，実際に実行してみよう。

学習活動

- 総合テストのうち，実施できるテストについて検討してみる。
- 作成したシステムのセキュリティについて，脆弱性があるかどうかを検討してみる。

指導上の留意点

- システムを実際に利用するシナリオを考えさせ，それに沿った機能テストを実施させる。
- 機能以外にテストしなければいけないことを考えさせ，それをテストする方法を話し合わせ，テスト可能なものを実施させる。

まとめ

まとめ

単体プログラムを結合させ，必要な結合テスト，機能テスト，非機能テストなどを考えて実施することにより当初の仕様に沿ったシステムになっているかを確認することができる。

学習

25 情報システムの評価・改善

研修内容

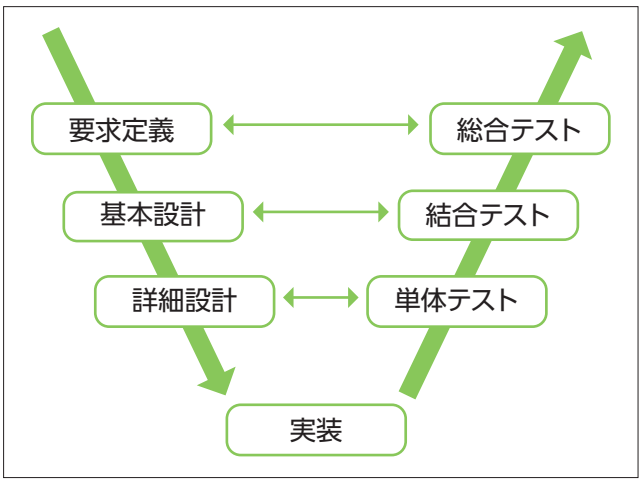
研修の目的

- 作成した情報システムについて，様々な視点から評価を行う方法について理解する。
- システムの作成において，プロジェクトマネジメントの視点から改善を行う方法について理解する。
- 適切な規模の関数に分ける等のソースコードの改良や，テストを改善する方法について理解する。
- 生徒が情報システムの評価・改善を理解し実施する授業ができるようになる。

この学習項目で使用するプログラミング言語は Python です。

1 情報システムの評価

情報システムを実際に稼働させる前に，要求定義をしっかりと満たしているかどうかの評価が必要となる。ウォーターフォール型のシステム開発では，開発工程とテスト工程を対応させたV字モデルという考え方がある図表１。ウォーターフォール型の開発は「時間をかけて仕様を設計し，その仕様の通りに実装してテストする」方式であり，仕様を決めたら変更が難しい銀行等のシステムなどの開発に従来から広く使われてきた。近年ではスマートフォンなどの普及により，「大まかに動く試作品を短期間で作り，その動作を確認しながら仕様を修正する」形のプロトタイプ型の開発や，機能を限定したシステムの作成と評価を行い，徐々に機能を追加しながら繰り返してシステムを作り上げるスパイラル型の開発も利用されている。



図表1　情報システム開発のV字モデル

要求定義を満たしているかどうかを総合テストやシステムテストと呼ばれるテスト工程で確認していく。総合テストには，機能テストや非機能テストといった分類があり，機能テストでは要求定義で必要とされている要求がプログラムの機能として盛り込まれているかどうかをテストしていく。実際の利用場面に沿ったテストケースを作成して，一連の機能が動作するかどうかを確認する。

非機能テストと呼ばれるテストでは，プログラムの機能ではなく処理速度や反応速度など，ユーザーが操作する際に不満がないレベルのものであるかをテストしていく。性能テストとは，情報システムを実際と同じように動かしてみて，要件を満たす性能が出るかどうかを確認するテストである。

総合テスト仕様書に記載すべき項目として，次のようなものがある。

・テスト概要　：どのようなテストなのか
・テストの実施環境　：ハードウェアやソフトウェアなどの環境
・テストケース　：機能テスト，性能テストなどのテストの種類
　どのような結果が得られればテスト合格なのか
・テスト手順　：どんなテストデータを用いるかも含めて具体的に記述
・テスト実施担当者　：テスト結果を確認する担当者が異なる場合は，それも記載
・テストスケジュール：テスト環境の構築やテストデータの用意なども含む

演習 1

EXERCISE

総合テストにおける機能テストとして，「新着図書の登録機能」をテストするためのシナリオを考えてみましょう。

2 プログラムの改善・改良

(1) プログラムの構造化

日常書く文章は，段落も見出しもなく何ページも続くと，論理的な構造として表現することが難しく，それを正しく読むことも困難になってしまうことがある。プログラムも，「ここからここまでを繰り返す」「条件が成り立つときはこの処理とこの処理を実行する」などのように，明確な構造を持たせることが重要である。更に，まとまった内容については，章や節の見出しのように関数を定義して，その処理の名前を付けておくことでプログラム全体を把握しやすくなる。

(2) テスト駆動開発

通常はプログラムを作成したあとにテストのためのプログラムを作成するが，テストを中心にプログラムを作成していくテスト駆動開発(TDD：Test Driven Development)と呼ばれる手法も使われている。TDDでは，従来の開発のように単体プログラミングが完了してからテストを行うのではなく，顧客と相談して仕様を決定したあとに，まず入力に対して期待する結果や出力が得られるかどうかのテストの設計を行う。この時点ではプログラム本体をまだ作成していないのでテストには合格しない。このテストを全て実施できるようになり，テストで不合格が得られたことを確認してから，プログラムの作成に入る。このあとは，プログラムの作成とテストを繰り返して，テストに全て合格するようになった時点で単体プログラムが完成したと見なす図表2。

●テストツールの利用

これまでの学習では，テストを設計してテストコードを記述して実行してきた。テストをより効率的・効果的に行うためにはテストを補助してくれるテストツールを利用することも有用である。テストツールには，テストデータの生成を助けるものや，テストの網羅性を確認するためのカバレッジツール，テストを一度にもれなく行うためのテストランナーなどがある。また単体テストを助け，テスト駆動開発を行いやすくするテストのためのフレームワークがある。

Pythonにおいては，標準ライブラリに unittest モジュールというテストユニットが用意されている。

従来の開発順序	テスト駆動開発での開発順序
・内部設計書を作成 ・単体プログラムを作成 ・テストを設計 ・テストの実施（成功）	・仕様の確定 ・テストを設計 ・テストの実施（失敗） ・単体プログラムを作成 ・テストの実施（成功）

図表 2　開発順序の比較

演習 2

EXERCISE

次のテストプログラムを実行して結果を確認しましょう。

```
import unittest
import statistics

# 成績分析のプログラム
def def_mean(ary):
    sum=0
    n=len(ary)
    for num in ary:    # sum+=num の間違い
        sum=num
    return sum / n
def def_median(ary):
    n=len(ary)
    index1=int(n/2)    # index1=int(n/2)-1 の間違い
    if n%2==0:
        index2=index1+1
    else:
        index2=index1
    return (ary[index1]+ary[index2])/2
def def_max(ary):
    x=ary[0]
    for num in ary:
        if x > num:    # if x < num: の間違い
            x=num
    return x

# ここからは単体テストのためのコード
data = [2,3,3,5,6,6,7,8,8,9]
class Test(unittest.TestCase):
    def test_mean(self):
        self.assertEqual(statistics.mean(data), def_mean(data))
    def test_madian(self):
        self.assertEqual(statistics.median(data), def_median(data))
    def test_max(self):
        self.assertEqual(max(data), def_max(data))
if __name__ == '__main__':
    unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

演習 3

EXERCISE

この unittest の仕組みを使って，作成したプログラムの関数について単体テストを行ってみましょう。

(3) リファクタリング

情報システムについて，更なる機能の追加や改善が必要とされる場合もある。また，機能は変えないが，保守性を高めるためにプログラムを改善していくこともある。このような場合の考え方に「リファクタリング」が利用できる。

リファクタリングは「外部から見たときの振る舞いを保ちつつ，理解や修正が簡単になるように，ソフトウェアの内部構造を変化させること」と定義される。整理されたコードにするために，正常に動く状態のプログラムに変更を加えて書き直していく。変更を加えるが，リファクタリング前の動作や出力をリファクタリング後も維持していなければならない。リファクタリングのパターンの例として次のようなものがある 図表 3 。

パターン	対象	考え方
関数の抽出	ソースコード中に何度も現れる処理などを関数としてまとめる	・新たな関数を作る ・抽出したいコードを元の関数から新しい関数へコピーする ・抽出したコードに含まれる変数を調べ，新しい関数ではアクセスできない変数を特定し，引数として新しい関数に渡すようにする ・元の関数に残っているコードの部分を抽出された関数への呼び出しに変更する
変数のインライン化	戻り値として渡すためだけの一時的な役割の変数などを使わなくする	・代入の右辺に副作用（出力や変数の変更など）がないことを確認する ・代入が１度しか行われていないことを確認する ・その変数への最初の参照を探し，代入の右辺と置き換える ・変数を参照している箇所の置き換えを繰り返す ・変数の宣言と代入を取り除く
変数名の変更	現在の役割に対して良い名前ではないと思われる変数の名前を付け直す	・変数の宣言や変数の参照を探す ・それらの全ての箇所で変数名を変更する

図表３　リファクタリングのパターンの例

例えば次のようなプログラムを見てみよう。これは図書館の本の情報について，貸し出し可能かどうかを判定する機能を持つプログラムである。

```
import urllib.request, urllib.parse, json
def get(url,params):
    p=urllib.parse.urlencode(params)
    url=url+"?"+p
    with urllib.request.urlopen(url) as res:
        return json.loads(res.read().decode("utf-8"))

# 検索語入力関数
def title_input():
    print(" 検索するタイトルを入力してください ")
    title=input()
    return title

# 図書検索関数
def title_search(title):
    params = {
             "token": "test",
             "table": "book",
             "format":"json",
             "column": "title",
             "value": title
    }
    url = "https://api.eplang.jp/mext2/search"
    data = get(url,params)
    return data

# 貸出中判定関数
def disp(data):
    for record in data:
        if record['lent']=="true":
            print(record["title"]+"=>"+"貸出中です ")
        else:
            print(record["title"]+"=>"+"貸出可能です ")

# 全体の実行
title=title_input()
result=title_search(title)
disp(result)
```

演習・4

EXERCISE

「関数名の変更」や「変数のインライン化」などのリファクタリングの考えに基づき，貸し出し可能かどうかを判定するプログラムにおいて，機能を変えないようにプログラムを改善しましょう。

3 チームでの開発の進め方

(1) コーディングスタイル

複数人のチームで開発する場合，コーディングスタイルを統一する必要がある。コーディングスタイルとは，変数名や関数名の規則，コメントの書き方など考えなければいけないルールについて統一した基準を作成し，それに従うことである。また，言語には決められた規則が存在する。例えばPythonの構文規則では，変数名について次の制約が存在する。

- 変数名の1文字目は数字にはできない
- 予約語になっている語句は変数名に使用できない

コーディングスタイルとしては，変数名が長くなったときに変数名に大文字を使うのか，「_（アンダースコア）」でつなぐのか，などいくつかの方法が考えられるが，これをプロジェクト内で統一しておかなければ個人の手を離れたときに他のメンバーの作業効率を下げてしまうことになる。仕様としては決められていないがPythonでは，次のようなルールが推奨されている。

- 関数名や変数名は小文字のみとする
- 読みやすくするために必要に応じて単語の間を区切る
- 単語の区切りは「_（アンダースコア）」で表す

またそれ以外にも，演算子の前後にスペースを入れるかどうか，タブでインデントする場合に1つのタブがスペース何個か，などをチームで統一しておきたい。Pythonで推奨されるコーディング基準を定めた規約にPEP8（Python Enhancement Proposal 8: Style Guide for Python Code）がある。

演習・5

EXERCISE

これまでに作成したPythonのプログラムが，Pythonの標準的なコーディング規約に適合しているかどうか，チェックツールで確認してみましょう。PythonのソースコードをチェックするオンラインツールとしてPEP8 Online Checker（http://pep8online.com/）などがあります。

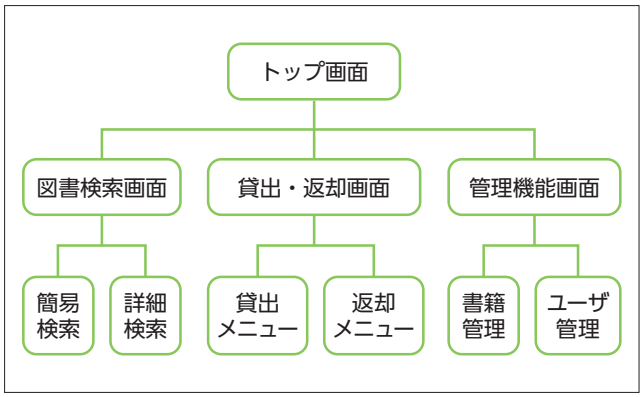
(2) プロジェクト・マネジメント

プロジェクトを成功させるために，プロジェクト・マネジメントと呼ばれる様々な管理手法が提案されている。

❶WBS

WBS（Work Breakdown Structure）とは，作業を分解して構造化する手法であり，システム開発プロジェクトにおける作業単位を決定するために有効である。プロジェクトに必要な全ての作業を細かくリストアップして，どのような作業にどれだけ時間がかかり，誰が担当するかを決定していく。

WBSを作成するにはプロジェクトのゴールを決定する。次に，ゴールに到達するために必要な作業を列挙していき，更には細分化していく。例えば，図書館システムの構築をWBSで表現すると次のようになる図表4，5。



図表4　図書館システムの機能分割の例

件名	担当者	状態	開始日	終了日	期限
簡易検索の画面設計	A	完了	2020/2/1	2020/2/10	2020/2/14
詳細検索の画面設計	A	処理中	2020/2/12		2020/2/28
検索機能モジュール作成	C	処理中	2020/2/1		2020/3/31

図表 5　図書管理システムの WBS の例

❷ガントチャート

プロジェクトの進捗を管理するためのツールの一つにガントチャート(Gantt chart)がある。ガントチャートとは，作業工程を分割して，作業に担当者を振り，時間軸で可視化したものである。先行作業との関係性などを確認しながら作業の進捗を確認できる図表6。

ガントチャートの機能を含むプロジェクト管理ソフトウェアには，市販ソフト以外でもフリーソフトウェア，Webブラウザで利用するタイプのものなどがあるが，まずは簡易なものを表計算ソフトウェアなどで作り，概念を理解するとよいだろう。

作業	開始日	完了予定日	先行作業	達成率	2020年4月 1	2	3	4	5	6	7	8
作業A	4/1	4/2	なし	100%								
作業B	4/1	4/3	なし	100%								
作業C	4/3	4/4	作業A	100%								
作業D	4/4	4/8	作業B	20%								
作業E	4/5	4/8	作業C	0%								

図表6　ガントチャートの例　本日

❸コミュニケーションツールの利用

プロジェクト管理をしていく上で，対面でのコミュニケーション以外にもビジネスチャットツールなどを利用したコミュニケーションでプロジェクトを進めていくことが考えられる。

ビジネスチャットツールなどでは，具体的な作成物やソースコードを共有しながらコミュニケーションを取ることが容易にできる。

ただし，開発を行う環境においてセキュリティ上の問題や規約上の問題がないかどうかは事前に検討しなければならない。

【参考文献・参考サイト】
●「Python プロフェッショナルプログラミング第 3 版」 株式会社ビープラウド 著　秀和システム(2018)
●「リファクタリング　既存のコードを安全に改善する　第 2 版」 Martin Fowler 著　児玉公信，友野晶夫，平澤章，梅澤真史 共訳　オーム社(2019)
●「Python unittest　ユニットテストフレームワーク」 https://docs.python.org/ja/3.7/library/unittest.html
●「PEP8 online check」 http://pep8online.com/
●「WBS とは？作り方の基本とガントチャートとの違いを解説」
https://backlog.com/ja/blog/wbs-introduce-how-to-make-excel-template-and-useful-tools/

学習活動と展開

学習活動の目的

- 制作した情報システムを評価できるようになる。
- 情報システムの制作過程を評価し，その過程を改善することができる。
- チームでの情報システム開発に利用される手法について理解する。

学習活動とそれを促す問い

	問　い	学習活動
展開1	制作した情報システムを評価するためには，要求定義に基づいてどのようなテストを行えばよいだろうか。	情報システムの総合テストについて，機能テスト，性能テストなどをどのように行うかを調べる。
展開2	情報システムの制作過程で，より効率的に開発を行うためにできる工夫について，考えてみよう。	グループで開発を効率的に進めるために必要なことと，それを実際に行う工夫について考える。
展開3	チームでシステム開発を行う場合に，守らなければいけない開発ルールには，どのようなものが考えられるだろうか。	グループで守らなければいけない開発ルールについて開発者の行動，プログラム作成，依頼者への対応など，様々な角度から開発ルールを考える。

展開1

問　い

制作した情報システムを評価するためには，要求定義に基づいてどのようなテストを行えばよいだろうか。

▼

学習活動

情報システムの総合テストについて，機能テスト，性能テストなどをどのように行うかを調べる。

▼

指導上の留意点

- 要求定義を確認させ，テストすべき項目をリストアップする。
- リストアップした項目について行う機能テストを考えさせる。
- 機能以外の部分でチェックすべき項目をリストアップする。
- リストアップした項目について性能テストを考えさせる。

展開 2

問　い

情報システムの制作過程で，より効率的に開発を行うためできる工夫について，考えてみよう。

学習活動

グループで開発を効率的に進めるために必要なことと，それを実際に行う工夫について考える。

指導上の留意点

- 効率的に開発を行う方法を Web ページや書籍で調べさせる。
- グループで話し合った工夫を全体で共有させる。
- 開発を効率的に行う方法としてプロジェクト・マネジメントなどの手法を学ぶ。
- その一つとして開発工程を細かく分けて，ガントチャートなどで表現させる。

展開 3

問　い

チームでシステム開発を行う場合に守らなければいけない開発ルールとしてどのようなものが考えられるだろうか。

学習活動

グループで守らなければいけない開発ルールについて開発者の行動，プログラム作成，依頼者への対応など，様々な角度から開発ルールを考える。

指導上の留意点

- 守るべきルールについて書籍や Web ページで調べさせる。
- グループで話し合ったルールを共有し，守るべきルールを整理させる。
- 複数の開発者が作成したソースコードを統合してシステムを作成する場合に必要なことを考えさせる。
- その一例としてソースコードのバージョン管理システムを調べ，実際に使用してみる。

まとめ

まとめ

- 情報システムが当初の仕様に合っているかを試す方法として，機能テスト，機能以外を試す非機能テストがあることを理解する。
- 情報システムを効率よく開発するためにガントチャートをはじめとしたプロジェクト・マネジメントの手法があることを理解する。
- 情報システムを様々な視点から評価し，制作過程の改善としてプロジェクト・マネジメントの手法を用いたり，グループで守るべきルールについて考えたりすることが有用であり，ソースコードのバージョン管理システムなどが必要であることを理解する。

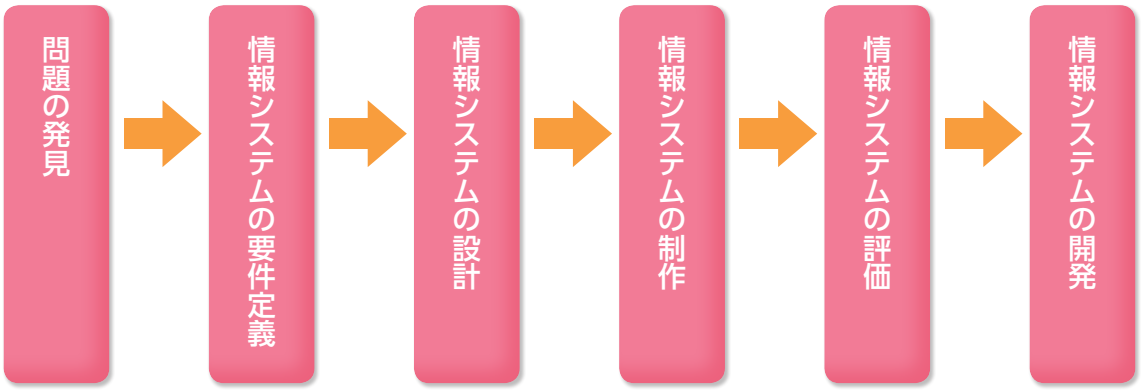
全体を通じた学習活動の進め方

全体を通じた学習活動の目的

●情報システムの果たす社会的な役割や影響を理解し，情報システムを設計・制作・評価した上で開発することができる。

全体を通じた学習活動の流れ

情報社会で実際に稼働している情報システムの仕組みやセキュリティ対策などについて調査する活動や，限られた教育現場の環境で実現できる小規模の情報システムを制作する一連の活動をグループで行う。



「情報システムとプログラミング」について授業をするにあたり，「情報Ⅰ」の「コンピュータとプログラミング」と「情報Ⅱ」の「情報とデータサイエンス」で学んだプログラミングの内容を参考に授業を計画する。その際留意しておくべきこととして

❶複数人による情報システムの開発
❷プロジェクト・マネジメント
❸情報システムの設計
❹情報システムの制作
❺情報システムの評価

などが挙げられる。

複数人による情報システムの開発 ❶

●実際の情報システムの開発は一人で行うものではないため，複数人で行うようにする。
●情報システムの開発の際には，担当ごとにそれぞれのモジュールのプログラムを開発し，これを結合してシステムとして稼働させるようにする。
●情報システムの設計・制作・評価を踏まえた上で，開発するように留意する。

プロジェクト・マネジメント ②	●情報システムの開発には，適切なプロジェクト・マネジメントが必要である。毎回ポートフォリオなどの活動記録を蓄積し，計画的に情報システムの開発過程を振り返り評価する場面を作る。 ●情報システムの開発を通して，課題解決をするためにプロジェクトの企画・進捗管理・コストの見積もり・グループメンバーへの作業の割り振りを体験させる。
情報システムの設計 ③	●情報システムが社会に与える効果やトラブルが起こった時の影響，更にはセキュリティ確保を意識した上で設計するようにする。 ●情報システムの設計においてその仕組みを表現する方法はDFDやUMLなど様々だが，目的に応じて適切で分かりやすい図を取り入れるようにする。
情報システムの制作 ④	●既存のモジュールを再利用し，外部のリソースを効果的に活用した制作を行う。 ●プログラムを制作しやすくするため，Web APIなどの組み込み関数やあらかじめ用意した関数を利用できるようにする。 ●開発は最低限の情報システムの制作にとどめ，実際に使用した上で，サイクルをかけながら機能を拡張していくことを想定して行う。
情報システムの評価 ⑤	●情報システムの設計及び開発の一連の流れにおいて，システムの自己評価や相互評価を取り入れ，それに基づいて改善させる。 ●可能であれば，実際に運用し活用する場面を設け，収集したデータをもとに評価を行う。 ●必要に応じて，間違った要件の評価の事例なども取り入れ，どのような要件の評価が適切か考えさせる活動を取り入れる。

全体を通じた学習活動を行う上での注意点

「情報システムとプログラミング」は,「情報Ⅰ」と「情報Ⅱ」のこれまでに学んできたプログラミングの内容を包括的に取り扱った単元である。教育現場の環境に合わせた情報システムとプログラミング言語を選定し，体験的に情報システムの開発が行えるようにする。

しかしながら，プログラミングで情報システムを実装する場合は，プログラミングの習熟度の差など生徒が一人でプログラミングをしていたときには想定していなかった様々な課題が発生する。その際には，プロジェクト・マネジメントを意識し，適宜進捗状況を把握させるようにする。

情報システムをただ開発するだけではなく，問題解決のための活用や，情報システムが果たす役割やセキュリティの確保についても考えられるように留意する。

全ての生徒が情報システムの開発者側になるわけではないが，情報システムの開発を通して，将来情報システムを発注する，あるいは利用する上で役に立つ視点を得られるように留意する。

MEMO