学科	ITエンジニア科3年制		**シラバス**
コード			科目趣旨・目的（背景となる社会情勢・業界動向・資格試験との関係、および、カリキュラム全体における位置づけ）
年度	2024		近年、AIが脚光を浴びており、AIを支える機械学習技術およびデータ分析技術が急速に発展・普及している。機械学習やデータ分析を活用する多くのプログラムはPython言語で記述されているため、産業界においてPython言語のスキルを有した技術者の確保が課題となっている。また、Pythonは比較的シンプルな言語仕様から構成されており、元々、海外では初学者向けのプログラミング言語として人気があった。以上の背景を踏まえて、本科目ではPython言語を通してプログラミングの基礎を習得し、さらにPython言語の言語仕様ならびに代表的な機能と標準ライブラリを理解することを目標とする。なお、Python言語は動的型付け言語であり、また、オブジェクト指向プログラミングに対応した言語である。したがって、本科目では静的型付け言語と比較すると、型に関する学習時間は短くなっており、その分、Python特有の言語機能やオブジェクト指向プログラミング、アルゴリズム、ソフトウェア品質など、Pythonエンジニアに求められる幅広い知識を扱う。型に関する知識を補うためには、別の科目において、Javaなどの静的型付け言語にも触れることが望ましい。また、リストやスタック・キュー、ハッシュテーブル、木構造などのデータ構造を学ぶことで効率的なプログラムを開発する力を身に付けることを目的とする。他にも、ビット演算、文字列処理、デザインパターンなど、Pythonエンジニアに求められる幅広い知識を扱う。
学年	2		
期	後期		
分野名	基礎		
科目名	Python基礎		
単位			
授業形態			科目概要
実務連携型授業	○		本科目の前半では、コンピュータに関する基礎知識から始まり、変数・代入・制御構文など、プログラミングの基礎的な知識を学ぶ。その後、Pythonがサポートする様々なコレクションを通して、関数・クラスなど、プログラムのモジュール化に関する基礎的な知識を学ぶ。さらに、ソフトウェア品質、Pythonにおけるコーディング規約など、実務において必要とされる応用的な知識を学ぶ。後半では、計算量の概念を学び、代表的なアルゴリズムの学習やゲーム開発を通して、複雑な処理を有するプログラムを記述するスキルを鍛錬する。その後、リストや木構造などの基本的なデータ構造やビット演算、文字列処理、デザインパターンについて学ぶことで、効率的かつ高度なプログラムの開発に必要な知識およびスキルを習得する。
必修・選択	必修		
前提とする科目			
展開科目			キーワード
関連資格			制御フロー　関数　クラス　コレクション　モジュール　入出力　例外処理　コーディング規約　ソートアルゴリズム　探索アルゴリズム　データ構造　リスト　スタック　キュー　ハッシュテーブル　木構造　ビット演算　文字列処理　デザインパターン
教員		常勤	到達目標
			コンピュータやプログラミングの基礎的な知識を理解して、また、Python特有の機能（コレクション・モジュール・標準ライブラリ）を理解することで、Pythonの公式ドキュメントを独力で読んで内容を理解できるようにすることを目標とする。さらに、データ構造やデザインパターン、ビット演算、文字列処理を活用して、効率的かつ高度なプログラムを開発できるようにすることを目標とする。

コマシラバス（90分授業コマ単位のシラバス）

90分/コマ	コマ主題	コマシラバス項目		内容
1	プログラミング・Pythonの基本知識	1_1	コマ主題細目	①コンピュータとソフトウェア　②プログラム　③プログラミング言語の種類　④Python言語
		1_2	細目レベル	①コンピュータについて、主要装置（CPU、メモリ、ハードディスク、キーボード、モニター、プリンター等）を図示して、各装置の機能および役割を解説する。ソフトウェアについて、オペレーティングシステム（OS）・インタプリタ・アプリケーション・ライブラリなど、ソフトウェアを実行する際の階層構造を図示して、各階層の詳細を説明する。さらに、OSやアプリケーション・Webアプリケーション（Webアプリ）・ミドルウェアなど、いくつかのソフトウェアの種類ついて図示しながら解説する。 ②ソースコード・機械言語・アセンブリ言語などプログラムの設計情報とコンパイラやインタプリタなどの処理系について解説した上で、コンピュータがプログラムを読み取って実行するまでの流れを図示しながら解説する。 ③構成部品、および、各構成部品の関係性を示した上で、各部品の開発に利用可能なプログラミング言語の特徴を解説する。例えば、Webアプリの場合は、クライアントプログラムを記述する言語とて、HTML・CSS・JavaScriptについて解説した上で、JavaScriptにトランスパイル可能な言語およびJavaScriptフレームワーク（React・Vue.js・AngularJS）について解説する。また、サーバプログラムを記述する言語として、Java・PHP・Python・JavaScriptについて解説する。同様にパソコンやスマートフォン向けのアプリケーションやスクリプトを記述するための言語についても合わせて解説する。 ④一般的なアプリケーションの開発においてPythonを使用する際と、機械学習・データ分析においてPythonを使用する際の相違点について解説した上で、各場面で利用するライブラリおよびフレームワークについて解説する。また、Jupyter NotebookおよびGoogle Colabなどの実行環境について解説する。
		1_3	5キーワード	①プログラム　②プログラミング言語　③Python言語　④データ分析　⑤実行環境
		1_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし

2	変数・データ型・代入・数値計算	2_1	コマ主題細目	①変数　②代入　③データ型　④演算子と数値計算
		2_2	細目レベル	①変数とは、プログラムで使用するデータに名前をつけて保持しておくための概念であることを解説する。また、変数の使い方について、変数を定義して値を読み書きするサンプルプログラムを提示しながら解説する。 ②代入とは、変数に新たなデータを格納する処理であることについて、サンプルプログラムを提示しながら解説する。また、代入文は、「=」の記号を用いて「(変数) = (値)」と記述することを解説する。 ③基本的なデータ型であるint型、float型、str型、bool型について、サンプルプログラムを提示しながら解説した上で、コレクションのデータ型(list型、tuple型、set型、dict型)について簡単に解説する。 ④演算子の一覧、計算時の優先順位、int型とfloat型の数値計算について、サンプルプログラムを提示しながら解説する。
		2_3	5キーワード	①変数　②演算子　③代入　④データ型　⑤数値計算
		2_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
3	制御フロー	3_1	コマ主題細目	①リストの基礎　②繰り返し (1)　③条件分岐　④繰り返し (2)　⑤繰り返しの制御　⑥パターンマッチ
		3_2	細目レベル	①リストの基礎について解説する。リストリテラルの表記方法、リスト内の各要素へのアクセス方法について、サンプルプログラムを提示しながら解説する。 ②for文の構文と意味を説明した上で、for文とrangeクラスを組み合わせた繰り返し方法について解説する。また、for文はリストなどの指定した要素集合の各要素を列挙しながら、処理を繰り返すための構文であることについて、サンプルプログラムを提示しながら解説する。 ③if文およびif式の構文および意味を解説する。また、if文およびif式を用いることで、指定した条件を満たしたときにだけ、特定の処理を実行できることについて、サンプルプログラムを提示しながら解説する。 ④while文を用いた繰り返しについて解説する。while文は指定した条件を満たす間、処理を繰り返すための構文であることについてサンプルプログラムを提示しながら解説する。 ⑤無限ループについて解説した後に、break文およびcontinue文を扱った繰り返し処理の制御方法について、サンプルプログラムを提示しながら解説する。 ⑥match 文について解説する。match 文はPython3.10 でサポートされた新機能であり、パターンごとに分岐を記述でき、マッチしたパターンの処理だけを実行できることについて、サンプルプログラムを提示しながら解説する。また、パターンマッチ特有の機能であるワイルドカードについても、サンプルプログラムを提示しながら解説する。
		3_3	5キーワード	①リスト　②for文　③if文　④while文　⑤match 文
		3_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
4	関数とメソッド	4_1	コマ主題細目	①関数　②オブジェクトとメソッド
		4_2	細目レベル	①関数とその使用法を解説する。まず、電車を表示するサンプルプログラムを用いながら、関数の定義・機能と関数を用いるための要件である「定義」および「呼び出し」を説明する。次に、仮引数及び実引数の定義を紹介する。電車を表示するサンプルプログラムを参照しつつ、仮引数を関数定義に使用することによってより複雑なプログラムを作ることが可能となることを説明する。関数定義において仮引数を使用する場合の注意点についても解説する。 ②文字列・リスト・関数といったオブジェクトは、メソッドを有する。本講義では、まず、オブジェクトのID(本講義では、オブジェクトの同一性をこのように呼称する)をis演算子または「= =」を用いて比較することができることについて、文字列を例にとって確認し、upper()、lower()といった文字列のメソッドをいくつか紹介していく。これまでの講義と同様、それらの説明はサンプルプログラムを用いて行う。次に、リストのメソッドについて解説する。具体的には、append()、extend()、reverse()について、サンプルプログラムを使用して説明をする。リストのメソッドと文字列のメソッドとの間には、いくつかの相違が見られるため、学習者はその点についても把握しなければならない。最後に、関数をオブジェクトとして使用した場合の挙動について、サンプルプログラムを用いて解説する。
		4_3	5キーワード	①関数　②引数　③戻り値　④オブジェクト　⑤メソッド
		4_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし

5	コレクション	5_1	コマ主題細目	①コレクションの種類 ②リスト ③タプル ④集合 ⑤辞書
		5_2	細目レベル	①リスト、辞書、タプルの概要について解説する。いずれも複数のデータを保持するためのデータ構造であることは共通しているが、データの保持の仕方に違いがあり、置換、削除、挿入、検索、走査などの処理の性能に違いがあることについて、サンプルプログラムを提示しながら解説する。 ②リストとは、複数のデータに番号を付与して特定の順番で並べたデータ構造であることについて、サンプルプログラムを提示しながら解説する。 ③タプルとは、ひとかたまりのデータを組みとして保持するデータ構造であることを解説する。 タプルはリストとよく似ているが、リストは複数のデータの繰り返しに用いることが通常であり、タプルは全体が一つのデータであって、各要素をデータの属性を表すために用いることについて、サンプルプログラムを提示しながら解説する。 ④集合とは、重複を許さず、また、順序を保持せずに複数の値の集合を記録するデータ構造であることについて、サンプルプログラムを提示しながら解説する。 ⑤辞書とは、複数のデータをキー（データ検索のID）とバリュー（実際のデータ）の対応関係で保持するデータ構造であることについて、サンプルプログラムを提示しながら解説する。
		5_3	5キーワード	①リスト ②タプル ③集合 ④辞書 ⑤キー/バリュー
		5_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
6	クラス	6_1	コマ主題細目	①名前空間とスコープ ②クラス ③オブジェクト指向プログラミング ④継承
		6_2	細目レベル	①Pythonにおいて、クラスは名前空間の仕組みを活用して実現される。そこで、まず、名前空間およびそれと関連するスコープについて解説する。 ②クラスの定義（コンストラクタ・インスタンス変数・インスタンスメソッド）と定義したクラスのインスタンスの生成方法を解説する。また、Pythonの組み込み型であるリスト・タプル・集合・辞書にもクラスがあり、各組み込み型のリテラルはインスタンス生成の簡略表記に相当することを解説する。 ③オブジェクト指向プログラミングとは、プログラムをオブジェクト（データと処理の組み合わせ）という概念に基づいて整理して部品化していく考え方をいう。本講義では、人間のBMI値を計算する場面を想定し、オブジェクト指向プログラミングの考え方に基づく場合と従来の手続型プログラミングの考え方に基づく場合とを比較しつつ、オブジェクト指向プログラミングの有用性を理解していく。 ④クラスの継承の方法および意味論とともに、継承によるモジュール化のメリットを解説する。さらに、オブジェクト間の継承関係を確認する方法について、isinstance関数とissubclass関数を扱ったサンプルコードを提示しながら解説する。また、多重継承について、基本的な概念のみを解説する。
		6_3	5キーワード	①オブジェクト指向プログラミング ②クラス ③インスタンス ④継承によるモジュール化 ⑤多重承継
		6_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
7	モジュール	7_1	コマ主題細目	①モジュール ②パッケージ ③標準ライブラリと外部ライブラリ
		7_2	細目レベル	①モジュール・パッケージ・ライブラリの特徴および相違点について解説する。 モジュールとは、1つのPythonファイルのことであることを解説する。パッケージとはPythonファイルを集めてディレクトリとし、外部に公開したものであることを解説する。ライブラリはパッケージをWeb上に公開し、誰でも利用可能なものとして保守・運用されているものであることを解説する。 モジュールの作り方、呼び出し方について、サンプルプログラムを提示しながら解説する ②パッケージの作り方、呼び出し方について解説する。 通常のディレクトリをPythonパッケージとして認識させるための _init_.py について、サンプルプログラムを提示しながら解説する。さらに、import 文の構文と意味を解説してから、絶対インポートと相対インポートを解説する。*を用いた全インポートについて、サンプルプログラムを提示しながら解説する。 ③標準ライブラリと外部ライブラリの違いと、具体例について解説する。 標準ライブラリとは、Python が公式に定めた、処理系が必ずサポートしなければならないライブラリのことであると解説する。一方、外部ライブラリとは、Pythonコミュニティのプログラマが独自に作成しているライブラリのことであると解説する。 Pythonは外部ライブラリが非常に充実しており、複雑な処理を初心者でも簡単に試用できることが大きな特徴であると述べる。
		7_3	5キーワード	①モジュール ②パッケージ ③標準ライブラリ ④外部ライブラリ ⑤import 文
		7_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし

8	入出力	8_1	コマ主題細目	①標準入出力 ②ファイル入出力 ③クリーンアップ処理
		8_2	細目レベル	①標準入出力および標準エラー出力の概念について解説した上で、Pythonにおける標準入出力および標準エラー出力の方法について、そこで使用される関数(input()関数、print()関数、strip()関数等)に関する一般的な説明とともに、サンプルプログラムを提示しながら解説する。 ②ファイルの入出力について、ファイルの概念に関する説明及びその具体例を交えながら解説する。その上で、Pythonにおいてファイルを開き、文字列データおよびバイナリデータを読み書きする方法について、サンプルプログラムを提示しながら解説する。 ③ファイルなどのリソースは、プログラムが本来期待される動作と異なる挙動をすることを防止するため、その使用後に解放する必要があることを解説する。その上で、開いたファイルを閉じてリソースを解放しプログラムが意図した動作をするようにする方法として、close()関数を使う方法とwith 文を使う方法を紹介する。その後、それらの方法について、サンプルプログラムを提示しながら、close()関数及びwith文の使用に際しての注意点も指摘しながら解説する。
		8_3	5キーワード	①標準入出力 ②標準エラー出力 ③ファイル入出力 ④with文 ⑤closeメソッド
		8_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
9	エラーと例外	9_1	コマ主題細目	①エラーの種類 ②例外の処理 ③例外の送出 ④独自例外の定義
		9_2	細目レベル	①Pythonには構文エラーと例外の2種類のエラーがあることを解説して、各種類のエラーの具体例やエラーから問題を突き止める方法について、サンプルプログラムとエラーの例を提示しながら解説する。 ②try文、try節、except節、catch節、finally節の構文および意味を解説して、例外が送出されたときの対応方法を解説する。また、いくつかの具体的な例外を取り上げ、各例外の意味について、例外を処理するサンプルプログラムを提示しながら解説する。 ③raise文の構文および意味を解説して、例外の送出方法を解説する。また、fromを用いて例外を連鎖させる方法を解説する。さらに、いくつかの具体的な例外を取り上げ、各例外の使い方について、例外を創出するサンプルプログラムを提示しながら解説する。 ④既存の例外クラスを継承したクラスを定義することで、独自の例外を定義する方法を解説する。親クラスとなるいくつかの具体的な例外を取り上げ、各例外を継承して独自の例外を定義する方法について、サンプルプログラムを提示しながら解説する。
		9_3	5キーワード	①構文エラー ②例外 ③例外処理 ④raise文 ⑤独自の例外
		9_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
10	ソフトウェア品質とコーディング規約	10_1	コマ主題細目	①ソフトウェア品質 ②コーディング規約
		10_2	細目レベル	①ソフトウェア品質は、利用者がソフトウェアを使用する際に操作するユーザインタフェースに関する品質から、開発者がソフトウェアを保守する品質まで、幅広い内容を包含する概念である。本講義では、ソフトウェア品質を定量的に評価するための枠組みとなるソフトウェア品質モデルおよびそこで示される各品質特性について概観する。JIS X 25010:2013およびISO/IEC 25010:2011（SQuaREシリーズ）において、ソフトウェア品質モデルは製品品質モデル、利用時の品質モデル、データ品質モデルの3種類に分類されており、本講義ではその中でもプログラムに関連する製品品質モデル及び利用時の品質モデルについて解説する。学習者には、ソフトウェア品質の概要を把握するとともに、プログラムの作成に密接に関わる品質特性である機能適合性、性能効率性、保守性に関し深く理解することが求められる。 ②コーディング規約とは、プログラムを記述する際に守るべきルールである。授業内では、コーディング規約の一例としてPythonにおいて最も著名なコーディング規約であるPEP 8の中からインデント、二項演算子の改行位置、式や文中の空白文字、コメント、命名規約を抜粋して解説する。また、コーディング規約に違反した場合の具体例を示し、その違反の影響を考えてもらい、コーディング規約に反した場合にプログラムの内容を理解することが難しくなることを体感する。コーディング規約の存在意義、遵守の必要性、コーディング規約とソフトウェア品質の関係性等を理解していく。
		10_3	5キーワード	①ソフトウェア品質 ②品質特性 ③品質副特性 ④PEP ⑤コーディング規約
		10_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし

11	探索アルゴリズム	11_1	コマ主題細目	①時間計算量・空間計算量　②線形探索　③二分探索
		11_2	細目レベル	①時間計算量と空間計算量の概要を説明した上で、プログラムの動的な品質に関して一般的な説明を行う。さらに、たとえ、同様の仕様を満たすプログラムであっても、ソートアルゴリズムのように、採用するアルゴリズムによって計算量が異なるケースがあることを解説し、その後、演習を実施することにより学習者の理解を促進する。 ②前提となる探索（検索）アルゴリズムの要件について確認する。 探索アルゴリズムの一つとして、リストや配列などに入った要素を先頭から1つずつ確認し探索を行うアルゴリズムである線形探索について解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。 ③探索アルゴリズムの一つとして、整列済みの配列などに対して探したい要素との大小関係をもとに探索を行うアルゴリズムである二分探索について解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。
		11_3	5キーワード	①時間計算量　②空間計算量　③探索アルゴリズム　④線形探索　⑤二分探索
		11_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
12	ソートアルゴリズム	12_1	コマ主題細目	①バブルソート　②選択ソート　③挿入ソート　④マージソート
		12_2	細目レベル	①前提となるソート（並べ替え）アルゴリズムの要件について確認する。昇順・降順などの言葉の意味を確認する。 簡単なソートアルゴリズムの一つとして、隣り合う要素の大小を比較しながらソートを行うアルゴリズムであるバブルソートについて解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。 ②簡単なソートアルゴリズムの一つとして、配列内の最大値もしくは最小値を見つけてソートを行うアルゴリズムである選択ソートについて解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。 ③簡単なソートアルゴリズムの一つとして、追加要素を整列済みの配列のどこに入るか調べて挿入を行うことでソートを行うアルゴリズムである挿入ソートについて解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。 ④高速なソートアルゴリズムの一つとして、ソート対象のデータを分割していき、細分化を終えたら整列しながら要素を併合していくことでソートを行うアルゴリズムであるマージソートについて解説する。擬似言語で記述されたアルゴリズムを読んで理解させ、演習として、同アルゴリズムのPythonプログラムを実装させる。
		12_3	5キーワード	①ソートアルゴリズム　②バブルソート　③選択ソート　④挿入ソート　⑤マージソート
		12_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
13	アプリケーション開発（1）	13_1	コマ主題細目	①コンソールアプリケーションの説明　②コンソールアプリケーションの実装　③コンソールアプリケーションの追加実装
		13_2	細目レベル	①コンソール上で動くアプリケーション実装の例としてゲームの開発を行う。そこで、まず、コンソールアプリケーションと、開発を行う上で最低限必要になるプログラムの雛形について解説する。 雛形については、while文を用いて無限ループを発生させ、input関数、break文などを用いて、特定の文字列を入力したら、無限ループを終了するといった簡素なプログラムを用意する。 ②①で用意した雛形を使い、簡単なゲームの開発の演習を行う。 ゲームの内容は、フィールド上に置かれた敵オブジェクトを、同じフィールド上に駒を配置することで取るといったものである。 ゲームの仕様について解説し、雛形を元に制御構文や関数、コレクションなど、クラス以外でこれまで扱った構文を使って実装させる。 ③②で実装したゲームに、追加の仕様を実装し、より複雑なゲームの開発を行う。 追加のゲームの仕様は、敵オブジェクトや駒をランダムで配置し、駒を動かして敵オブジェクトを取っていくという内容である。駒は1つだけ配置するものとする。 ゲームの追加の仕様について解説し、②で開発したゲームを元に制御構文や関数、コレクションなど、クラス以外でこれまで扱った構文を使って実装させる。
		13_3	5キーワード	①コンソールアプリケーション　②while文　③input関数　④関数　⑤コレクション
		13_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし

14	アプリケーション開発(2)	14_1	コマ主題細目	①クラスを使ったコンソールアプリケーションの実装 ②クラスを使ったコンソールアプリケーションの追加実装
		14_2	細目レベル	①クラスを使ったコンソールアプリケーション開発の演習を行う。 前回講義で作成したプログラムに、追加でクラスを用いた実装をするゲーム開発を行う。 フィールドに配置する敵オブジェクトや駒、フィールドなど複数のクラスの定義し、それぞれのインスタンスを生成するような実装を行う。敵オブジェクトや駒の座標、フィールドの初期状態など、インスタンス生成時に各インスタンスのデータ属性を定義したいため、各クラスで_init_関数を用いて実装を行う。 クラスを用いたアプリケーション開発を通してオブジェクト指向プログラミングの理解を促す。 ②①で実装したゲームに、追加の仕様を実装し、より複雑なゲームの開発を行う。 追加のゲームの仕様は、配置する駒を複数フィールドに配置するという内容である。 追加の仕様を実装することで、インスタンスの複雑な操作を行い、オブジェクト指向プログラミングへのより深い理解を促す。
		14_3	5キーワード	①コンソールアプリケーション ②オブジェクト指向プログラミング ③クラス ④インスタンス ⑤_init_関数
		14_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
15	アプリケーション開発(3)	15_1	コマ主題細目	①継承を使ったコンソールアプリケーションの実装 ②リファクタリング
		15_2	細目レベル	①継承を使ったコンソールアプリケーション開発の演習を行う。 前回講義で作成したプログラムを元に、追加の仕様を実現したゲームを実装させる。実装する際には、継承を用いる。 追加の仕様として、異なる動きをする複数の種類の駒を実現する。この仕様を満たすために、各駒の共通するメソッドを持たせるような親クラスを定義し、異なる動きを実現するようなメソッドを持つ子クラスを定義するような実装を行う。 継承を用いたアプリケーションを通してオブジェクト指向プログラミングの理解を促す。 ②①で実装したゲームのリファクタリングを行う。 リファクタリングとは、外から見たプログラムの振る舞いを変えずに、容易な理解や修正を実現するようにプログラムの内部構造を改善することである。 具体的には、子クラスで親クラスのメソッドをオーバーライドする箇所について、オーバーライドするクラスのメソッドの処理を再度記述しなくても済むように、super関数を用いた実装に修正を行う。
		15_3	5キーワード	①継承 ②親クラス ③子クラス ④super関数 ⑤オブジェクト指向プログラミング
		15_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
16	リストと配列	16_1	コマ主題細目	①データ構造 ②単方向リスト ③双方向リスト ④循環リスト ⑤Pythonにおける配列
		16_2	細目レベル	①データ構造について解説する。 データ構造とは、データを効率的に扱うために組み立てられた可変的な情報のまとまりである。データ構造に関する基本的な概念と主要なデータ構造であるリスト・スタック・キュー・木・グラフの概要を説明し、データ構造が時間計算量および空間計算量に与える影響を解説する。 ②単方向リストについて解説する。 単方向リストは、データの順序付けのための一般的なデータ構造の1つで、各ノードがデータと次のノードへのリンクを持つ構造である。単方向リストの概要を説明して、単方向リストにおける値の探索・挿入・削除方法について、サンプルプログラムを提示しながら解説する。 ③双方向リストについて解説する。 双方向リストは、各ノードがデータと前後のノードへのリンクを持ち、データを前後どちらからもアクセス可能にするデータ構造である。双方向リストの概要と、単方向リストとの相違点を解説する。さらに、双方向リストにおける値の探索・挿入・削除方法について、サンプルプログラムを提示しながら解説する。 ④循環リストについて解説する。 循環リストは、末尾のポインタが最初の要素を指すようになっているリストで、単一方向リストや双方向リストを改変したデータ構造である。循環リストの概要と、単方向リストとの相違点を解説する。さらに、循環リストにおける値の探索・挿入・削除方法について、サンプルプログラムを提示しながら解説する。 ⑤Pythonにおける配列について解説する。 Pythonのarrayモジュールの使い方を説明して、リストとarrayの違いを理解する。Pythonのarrayモジュールの概要と、配列とリストの相違点を解説する。さらに、配列における値の探索・挿入・削除方法について、サンプルプログラムを提示しながら解説する。
		16_3	5キーワード	①データ構造 ②単方向リスト ③双方向リスト ④循環リスト ⑤配列
		16_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし

17	スタックとキュー	17_1	コマ主題細目	①スタック ②キュー ③優先度付きキュー
		17_2	細目レベル	①スタックについて解説する。 スタックは後入れ先出し(LIFO: Last In First Out)という性質を持つデータ構造である。要素が追加される際には一番上に積まれ、取り出される際には一番上の要素から取り出される。スタックの振る舞いやその使用例について、サンプルプログラムを提示しながら解説する。 ②キューについて解説する。 キューは先入れ先出し(FIFO: First In First Out)という性質を持つデータ構造である。新しく挿入される要素はキューの最後に追加され、取り出される要素は最初に入れられた要素から順に取り出される。キューの振る舞いやその使用例について、サンプルプログラムを提示しながら解説する。 ③優先度付きキューについて解説する。 優先度付きキューは、各要素が一定の優先度を持ち、優先度が高い要素が優先的に取り出されるようなデータ構造である。実際の問題解決方策など多面的な視点から理解を深めるように、優先度付きキューの振る舞いやその使用例について、サンプルプログラムを提示しながら解説する。
		17_3	5キーワード	①スタック ②後入れ先出し(LIFO) ③キュー ④先入れ先出し(FIFO) ⑤優先度付きキュー
		17_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
18	ハッシュテーブル	18_1	コマ主題細目	①ハッシュテーブル ②オープンアドレス法 ③チェイン法
		18_2	細目レベル	①ハッシュテーブルについて解説する。 ハッシュテーブルは、キーと値をマッピングするデータ構造の一つであり、高速なデータ検索を可能とする特性から、頻繁に利用されるデータ構造であることを説明する。さらに、キーを元にハッシュ関数を使いハッシュ値を計算し、その結果を使ってデータへのアクセスすることを説明する。ハッシュテーブルの動作原理と構造について、サンプルプログラムを提示しながら解説する。 ②オープンアドレス法について解説する。 オープンアドレス法は、ハッシュテーブルにおける衝突解決策の一つである。衝突が起こった際に、他の位置へデータを再ハッシュすることで衝突を解決できる。オープンアドレス法でのデータの追加、探索、削除といった具体的な動作について、サンプルプログラムを提示しながら解説する。 ③チェイン法について解説する。 チェイン法は、ハッシュテーブルにおけるもう一つの代表的な衝突解決策である。衝突が起こった際、同じ格納位置にチェイン(連結リスト)を形成し、その中にデータを格納することで、衝突が生じた場合でもデータの追加や取得が可能になることを説明する。チェイン法でのデータの追加、探索、削除といった具体的な動作について、サンプルプログラムを提示しながら解説する。
		18_3	5キーワード	①ハッシュテーブル ②ハッシュ関数 ③ハッシュ値 ④オープンアドレス法 ⑤チェイン法
		18_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
19	木構造	19_1	コマ主題細目	①木構造 ②二分木 ③完全二分木
		19_2	細目レベル	①木構造について解説する。 木構造は情報科学における重要なデータ構造の一部であり、親子関係を持つデータを視覚化し、データ間の関係を簡単に理解することが可能であることを説明する。木構造の概念、木を構成する要素、性質、利点について解説し、簡単なサンプルプログラムを提示しながら解説する。 ②二分木について解説する。 二分木は木構造の一種であり、各ノードが最大で2つの子ノードを持つ性質を持つことを説明する。さらに、完全二分木や二分探索木など二分木のバリエーションについても簡単に説明する。二分木の特性や操作方法について、簡単なサンプルプログラムを提示しながら解説する。 ③完全二分木について解説する。 完全二分木は二分木の特殊なケースであり、すべての葉が同じ深さを持つか、あるいは最下層だけが右側からいくつかの葉が欠けている特性を持つことを説明する。この特性により、データのバランスが保たれ、効率的な処理が可能となることを説明する。完全二分木の特性や利点について、簡単なサンプルプログラムを提示しながら解説する。
		19_3	5キーワード	①木構造 ②ノード ③親子関係 ④二分木 ⑤完全二分木
		19_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし

20	二分探索木	20_1	コマ主題細目	①深さ優先探索　②幅優先探索　③二分探索木
		20_2	細目レベル	①深さ優先探索について解説する。 深さ優先探索は一つの経路を根から葉まで探索し、次に別の経路を同じように探索するアルゴリズムであることを説明する。また、3つの探索順序があり、先行順、中間順、後行順であることを説明する。深さ優先探索における各探索順序について、サンプルプログラムを提示しながら解説する。 ②幅優先探索について解説する。 幅優先探索は木構造やグラフを探索する際のアルゴリズムで、深さ優先探索とは異なり、まず近いノードから探索を行うことを説明する。また、最初に見つけた解が最短経路となる問題等に適していることを説明する。幅優先探索の動きやその適用例について、サンプルプログラムを用いて詳細に説明する。 ③二分探索木について解説する。 二分探索木は探索木の一種であり、各ノードが最大で2つの子ノードを持つ特性があること、また、特定の情報を効率的に探索することが可能であることを説明する。二分探索木の作成方法や探索、削除について、サンプルプログラムを用いて詳細に説明する。
		20_3	5キーワード	①深さ優先探索　②探索順序　③幅優先探索　④最短経路　⑤二分探索木
		20_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
21	複雑な木	21_1	コマ主題細目	①平衡木　②多分木　③ヒープ
		21_2	細目レベル	①平衡木について解説する。 平衡木は、データの追加や削除といった操作を行っても常に木の高さがバランスを保つように設計されたデータ構造であり、バランスを保つことにより、検索や挿入、削除の操作を高速に行うことが可能となることを説明する。平衡木の概念や性質、種類、操作方法について、平衡木の一種であるAVL木のサンプルプログラムを提示しながら解説する。 ②多分木について解説する。 多分木は、各ノードが2つ以上の子ノードを持つことが可能なデータ構造であり、階層的なデータの表現に利用され、木構造の中でも非常に柔軟性のあるデータ構造であることを説明する。多分木の概念や特性、その操作方法について、多分木の一種であるB木のサンプルプログラムを提示しながら解説する。 ③ヒープについて解説する。 ヒープは完全二分木の性質を持つ半順序木の一種であり、親子関係に一貫した順序付けを保つような木構造である。ヒープの概念や特性、種類、データの追加・削除といった操作方法について、サンプルプログラムを提示しながら解説する。
		21_3	5キーワード	①平衡木　②AVL木　③多分木　④B木　⑤ヒープ
		21_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし
22	ビット演算	22_1	コマ主題細目	①N進数　②シフト演算　③論理演算　④ビットマスク
		22_2	細目レベル	①N進数について解説する。 N進数とは、数を表現する際に用いられる基数のことであり、最も一般的なものが10進数で、コンピュータの内部で2進数が主に用いられていることを説明する。進数の概念と異なる進数間の変換方法を説明し、サンプルプログラムを提示しながら、10進数から2進数や16進数への変換方法やその逆の変換方法を解説する。 ②シフト演算について解説する。 シフトとは、ビット列を左右どちらかに動かす演算であり、ビットを左にシフトすると倍の値になり、右にシフトすると半分の値になることを説明する。サンプルプログラムを提示しながら、左シフトと右シフトの操作を具体的に解説する。 ③論理演算について解説する。 論理演算は、ビット単位でのAND、OR、XORなどの操作を行うものであることを説明する。論理演算の基本的な動作と、それを利用したプログラムの例について、サンプルプログラムを提示しながら解説する。 ④ビットマスクについて解説する。 ビットマスクとは、特定のビットだけを選択的に操作するための方法であり、ビット演算の中でも特に利用頻度が高いことを説明する。また、ビットマスクを活用することで、時間計算量・空間計算量を削減できることを説明する。サンプルプログラムを提示しながら、ビットマスクの作成方法や利用方法を解説する。
		22_3	5キーワード	①N進数　②左シフト　③右シフト　④論理演算　⑤ビットマスク
		22_4	コマ要素	□実務連携　□理解度確認テスト　■オリジナル教材　■ICT活用　■実習・実技・実験・演習　□該当なし

23	文字列処理	23_1	コマ主題細目	①Pythonの文字列処理 ②文字列アルゴリズム ③データ圧縮
		23_2	細目レベル	①Pythonの文字列処理について解説する。 Pythonには、文字列を処理するための多くのメソッドが用意されており、文字列の置換、分割、結合等の操作だけでなく、大文字・小文字変換、前後の余白除去などの機能も含まれていることを説明する。これらのメソッドを使うと、様々な文字列操作が容易になる。提供されている文字列処理の各機能について、サンプルプログラムを提示しながら解説する。 ②文字列アルゴリズムについて解説する。 文字列の検索、比較等の操作を行うための代表的な文字列アルゴリズムの概念と実装方法を説明する。具体的には文字検索のアルゴリズムとして、線形探索法、KMP法、BM法を、文字列の比較としてレーベンシュタイン距離について紹介する。各種文字列アルゴリズムについて、サンプルプログラムを提示しながら解説する。 ③データ圧縮について解説する。 ハフマン符号・連長圧縮・辞書式圧縮など、代表的なデータ圧縮アルゴリズムを紹介する。さらに、連長圧縮などの実装が容易なアルゴリズムについて、サンプルプログラムを提示しながら解説する。
		23_3	5キーワード	①文字列処理 ②KMP法 ③BM法 ④データ圧縮 ⑤連長圧縮
		23_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
24	デザインパターン1	24_1	コマ主題細目	①デザインパターン ②Singletonパターン ③Decoratorパターン ④Strategyパターン
		24_2	細目レベル	①デザインパターンについて解説する。 デザインパターンとは、ソフトウェア設計において頻出する一般的な問題を予め解決済みのパターンとして抽象化したものであり、開発者があらゆる状況における最適な設計方法を毎回見つけ出す必要性を減少させることで、生産性を向上させることを説明する。ソフトウェアパターンおよびデザインパターンの基本概念を説明し、代表的なソフトウェアパターンおよびデザインパターンについて、具体的なパターンを提示しながら概要を説明する。 ②Singletonパターンについて解説する。 Singletonパターンは、特定のクラスのインスタンスが1つしか存在しないことを保証するデザインパターンであることを説明する。Singletonパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。 ③Decoratorパターンについて解説する。 Decoratorパターンは、既存のオブジェクトに新たな機能を動的に付加することができるデザインパターンであり、クラスを新たに作成することなく、オブジェクトの動作を変更する手段として利用されることを説明する。Decoratorパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。 ④Strategyパターンについて解説する。 Strategyパターンは、同一の問題を異なるアルゴリズムで解決することに焦点を当てたデザインパターンであり、アルゴリズムを切り替えることによってオブジェクトの振る舞いを動的に変更できることを説明する。Strategyパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。
		24_3	5キーワード	①デザインパターン ②ソフトウェアパターン ③Singletonパターン ④Decoratorパターン ⑤Strategyパターン
		24_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし
25	デザインパターン2	25_1	コマ主題細目	①Adapterパターン ②Stateパターン ③Factory Methodパターン ④Template Methodパターン
		25_2	細目レベル	①Adapterパターンについて解説する。 Adapterパターンは、既存のクラスやインターフェースが新たに求められる要件や規格に合わない場合にその中間に介在する形で既存のクラスやインターフェースと新規の規格とを結びつける役割を果たすことを説明する。Adapterパターンの概念や構造を説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。 ②Stateパターンについて解説する。 Stateパターンは、オブジェクトが内部状態に応じて振る舞いを変更することを可能にするデザインパターンであり、状態の変化に応じて振る舞いを変える場合に有効であることを説明する。Stateパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。 ③Factory Methodパターンについて解説する。 Factory Methodパターンは、オブジェクトの生成をサブクラスに委譲することで、クラスの再利用性を高めることを説明する。Factory Methodパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。 ④Template Methodパターンについて解説する。 Template Methodパターンは、アルゴリズムの骨組みを定義しながら、一部の処理をサブクラスに任せるデザインパターンであり、オブジェクト指向プログラミングにおける継承・多態性を活用したパターンであることを説明する。Template Methodパターンの概念や構造について説明し、使い所と実装する方法、メリットとデメリットについて、サンプルプログラムを提示しながら詳しく解説する。
		25_3	5キーワード	①デザインパターン ②Adapterパターン ③Stateパターン ④Factory Methodパターン ⑤Template Methodパターン
		25_4	コマ要素	□実務連携 □理解度確認テスト ■オリジナル教材 ■ICT活用 ■実習・実技・実験・演習 □該当なし