プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

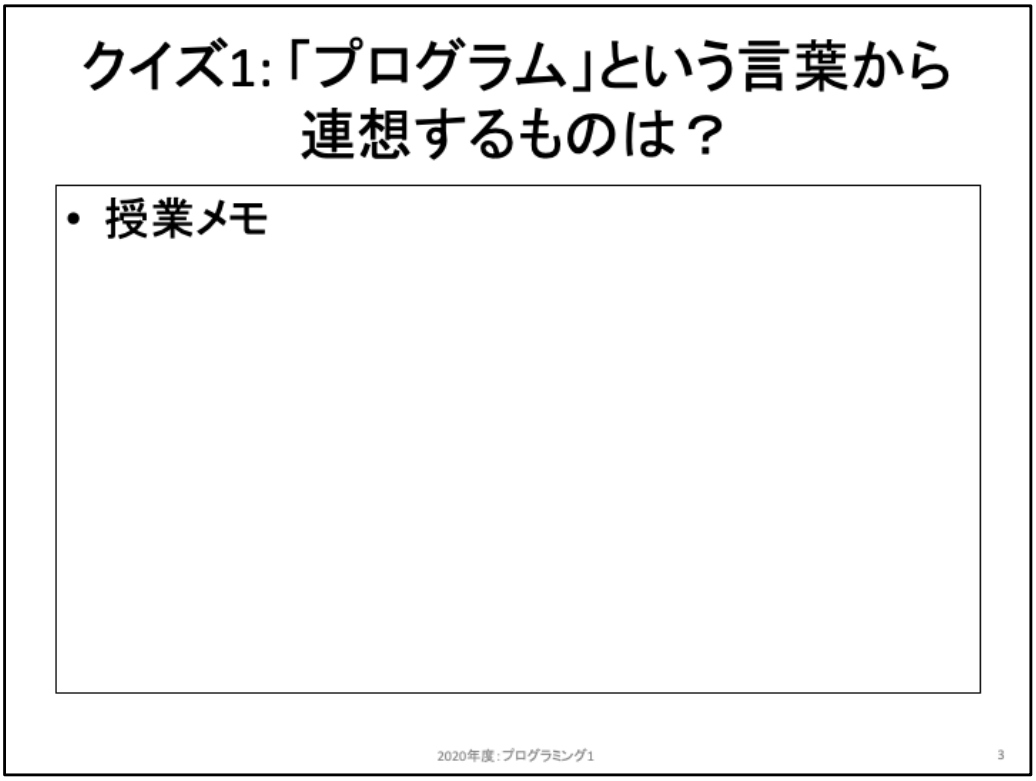
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2020/prog1/

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

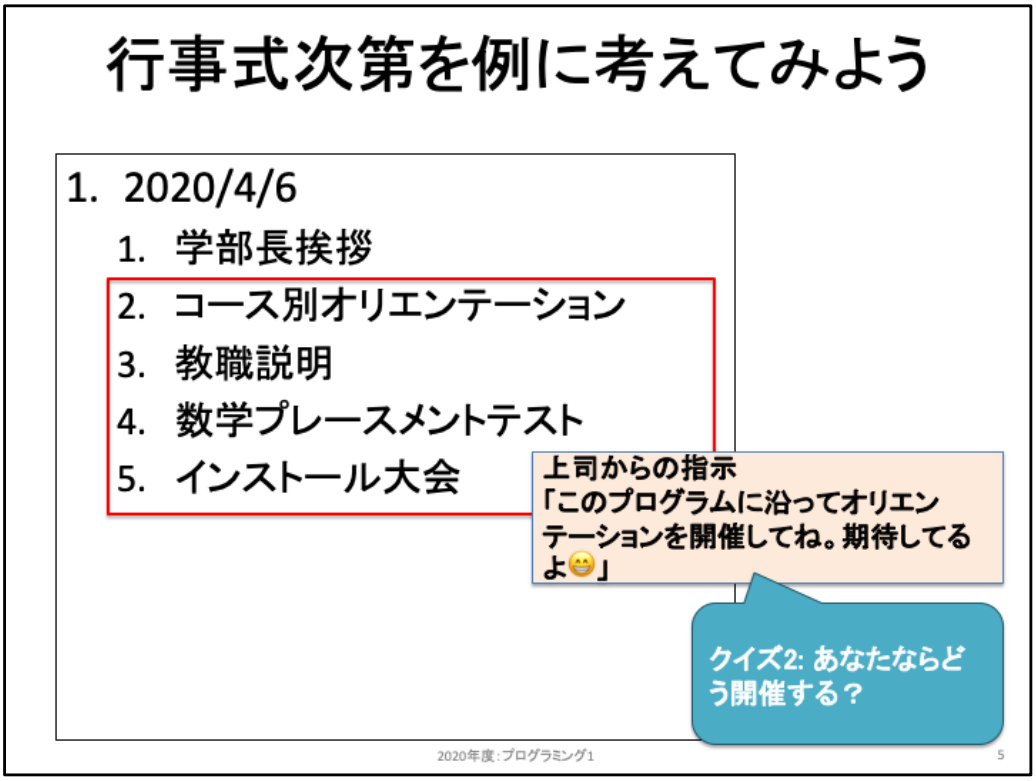
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



専門的な話を抜きにして、一般的な言葉としての「プログラム」は何を指すだろうか？

「プログラム」という言葉の意味

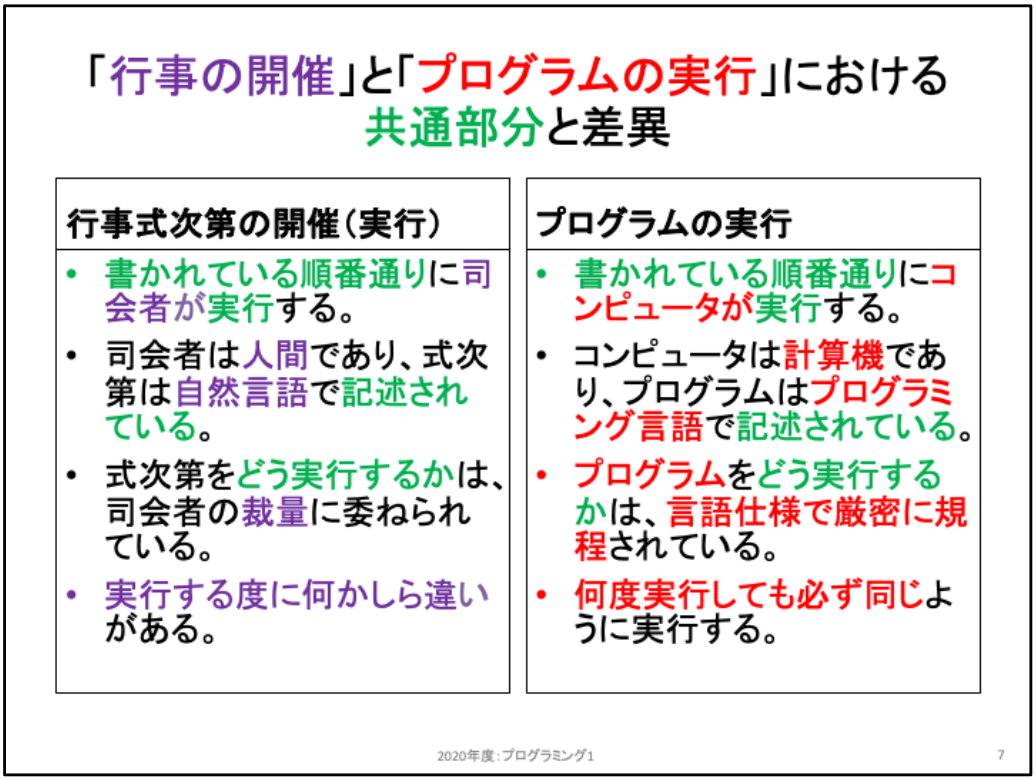
- 出典: スーパー大辞林
  - (1) 物事の予定。行事の進行についての計画。
  - (2) 映画演劇コンサートなどの演目や曲目，あらすじや解説などを書いた表や小冊子。
  - (3) コンピューターに，情報処理を行うための動作手順を指定するもの。また，それを作成すること。
- 授業「プログラミング」でやるのは(3)。でも、(1)～(3)は何かしら共通してるところがあるから「プログラム」と呼ばれているのでは？



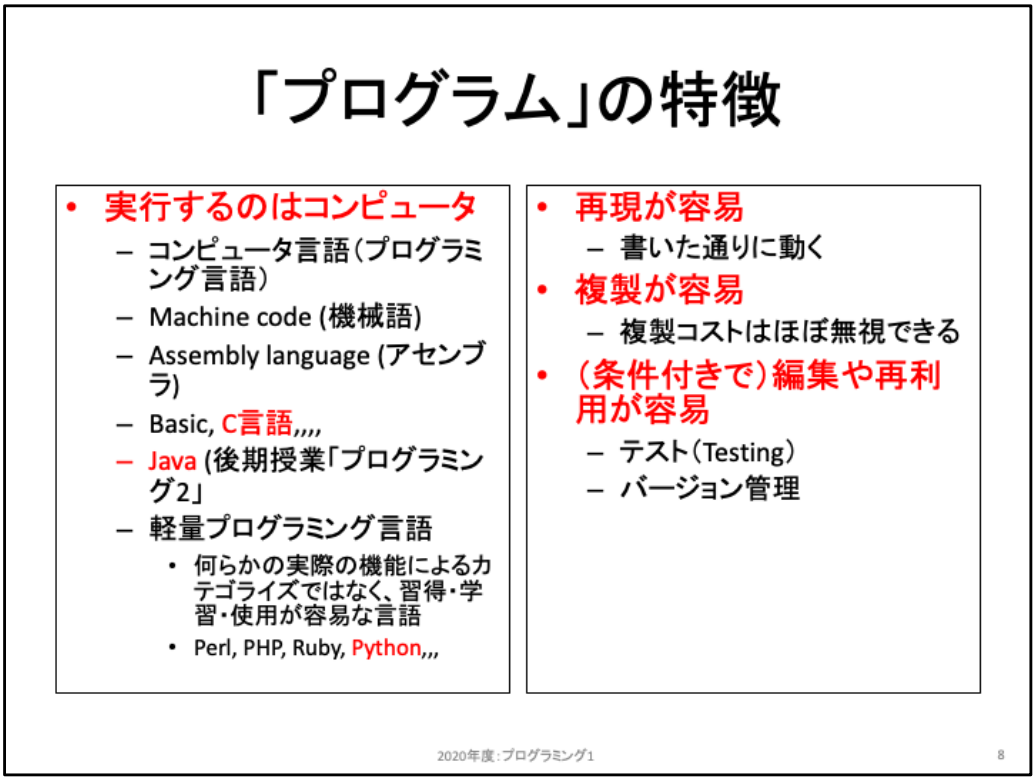
一般的な「行事（イベント）」であれば、主催者か司会者といった人が中心となり、特定のプログラムに従って進めていく。しかし具体的にどのように実施するのか事細かなことは「式次第（プログラム）」には書かれておらず、詳細は別資料が用意されていたり、もしくは明確に用意されていない部分についてはその場の状況に応じて柔軟に対応することで、進められていくことが多い。

皆さんの考える開催方法

- 授業メモ

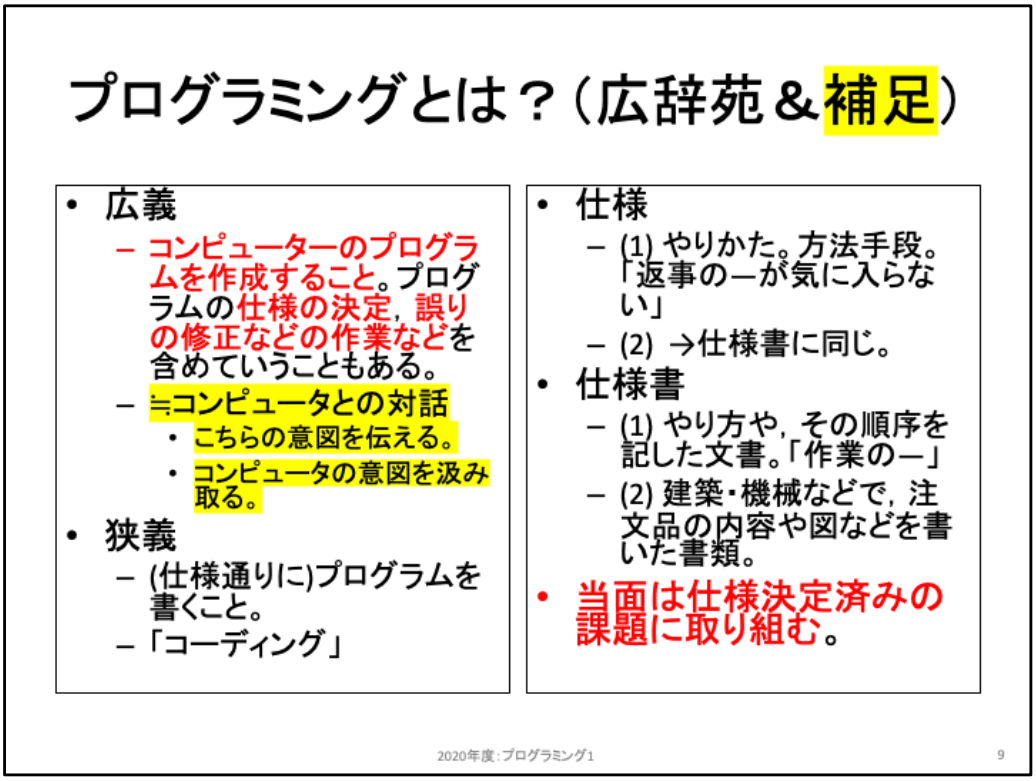


行事の開催と、コンピュータにおけるプログラムの実行には異なる部分も多いが、緑色で示したように共通する部分もある。この実行される内容を「プログラム」と呼んでいる。

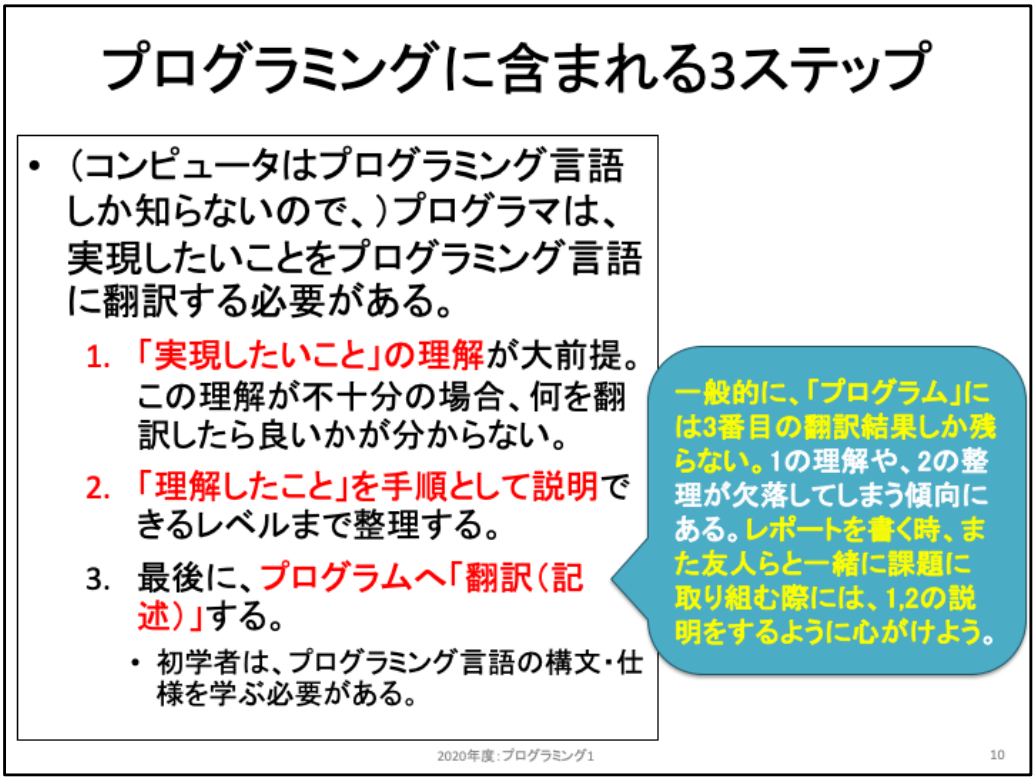


1年前期ではPythonのみを扱い、多くの言語に共通する「プログラミングの考え方」とともに具体的な言語仕様について学ぶ。1年後期では第2言語、第3言語としてJavaとC言語を学ぶ。

一般的な行事の場合、実行するためには場所や人員を確保し、プログラムの内容や実行方法を関係者全員に周知した上で実行する必要がある。このため様々な点でコスト（金銭・時間等）がかかり、また人間が実行するために、再現が容易ではない。これに対しコンピュータにおけるプログラムは、明確な仕様に基づいた専用のプログラミング言語で記述されており、実行方法が同一であり、コンピュータ（実行環境）さえあればどこでも再現可能である。この特徴を活かすことで、プログラムの更新履歴を管理（バージョン管理）したり、想定通りの動作をしているかの確認（テスト）を自動化するなど、コンピュータにおけるプログラムならではの特徴を踏まえた周辺技術についても学ぶ。



プログラミングとは、プログラムを作成することを指す言葉ではあるが、より講義には「どの順序で目的を達成したらよいか？」を検討すること（＝仕様の決定）や、自身や第三者が書いたプログラムにおける誤り（バグ）を修正する等、幅広い作業を含むことも多い。コンピュータに何を実行させたいかを記述したものがプログラムであることは既に述べたとおりだが、コンピュータはプログラミング言語しか解釈することができないため、まずはプログラミング言語の仕様について学ぼう。そのため、当面はほぼ仕様決定済みの課題について取り組むことになる。

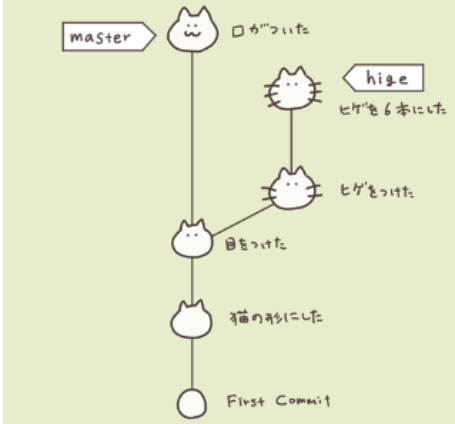


本来ならば「そもそも分からない点をどう伝えればよいのか」といったことも含めて「相談や教え合い」を大切にしてほしいが、2020年4月時点では新型コロナウイルスの影響のため、対面でのやり取りを制限せざるを得ない。そのため、文章でのやり取りを中心に、必要に応じてZoom（動画やり取り）等を使いつつ、伝える力を養おう。

プログラミングを円滑に進めるために

授業計画:
第13回

- バージョン管理



http://kray.jp/blog/git-pull-rebase/

授業計画:
第6回～

- より高度な機能
  - テスト(Testing)
  - デバッグ実行
  - ベクトル・行列演算
  - グラフ描画
  - 後期
  - (文字コード)
  - (正規表現)
  - (クラス)
  - (オブジェクト指向)

バージョン管理やより高度な機能については授業中間以降で取り扱う予定。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

実現したいことを理解し、手順として整理し、プログラミング言語で記述（翻訳）すること。ペアプロで互いに教え合おう。コンピュータとの対話を大切に。

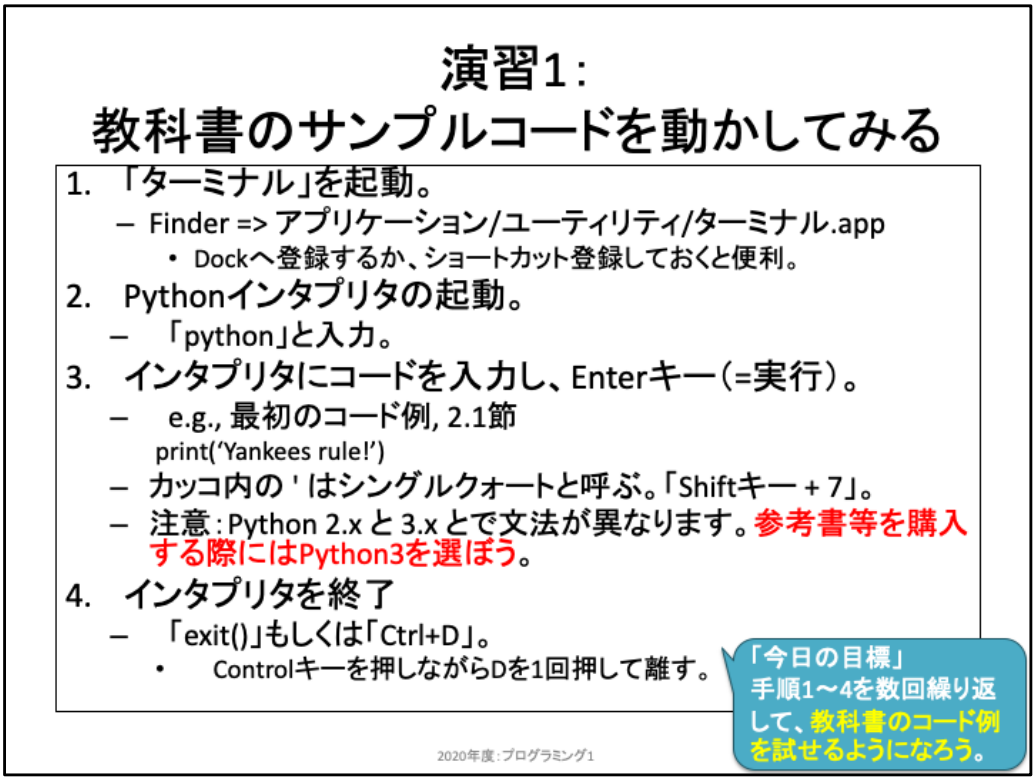
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



サンプルコードを動かすためには、ターミナルを起動し、その上でPythonインタプリタを起動する必要がある。現在ターミナル上で何が動作しているかはプロンプトで判断できる。プロンプトとは「命令の入力を待っている状態である」ということを明示するための目印である。

ターミナルを起動した直後は自動的にシェルが動作しており、Enterキーを押すことで同じプロンプトが返ってくる。この状態（シェルが入力を待っている状態）で、pythonと入力し、Enterキーを押すとPythonインタプリタが起動される。Pythonインタプリタが動作しているなら、「>>> 」というプロンプトが出力されているはずだ。この状態で何も命令を記述せずEnterキーを押すと、何度でもプロンプトが返り、「命令待ち」であることを示される。

この一連のやり取りも「コンピュータとの対話」である。コンピュータに対し何を伝えているのか、その結果コンピュータは何を返してきているのかを意識しながら取り組もう。

print('Yankees rule!') とは、そのカッコ内で指定されたリテラルや変数等を標準出力（ターミナル上に出力）しろという命令である。仮に print(1+2) のように括弧内に演算子が含まれている場合には、その演算結果が出力される。

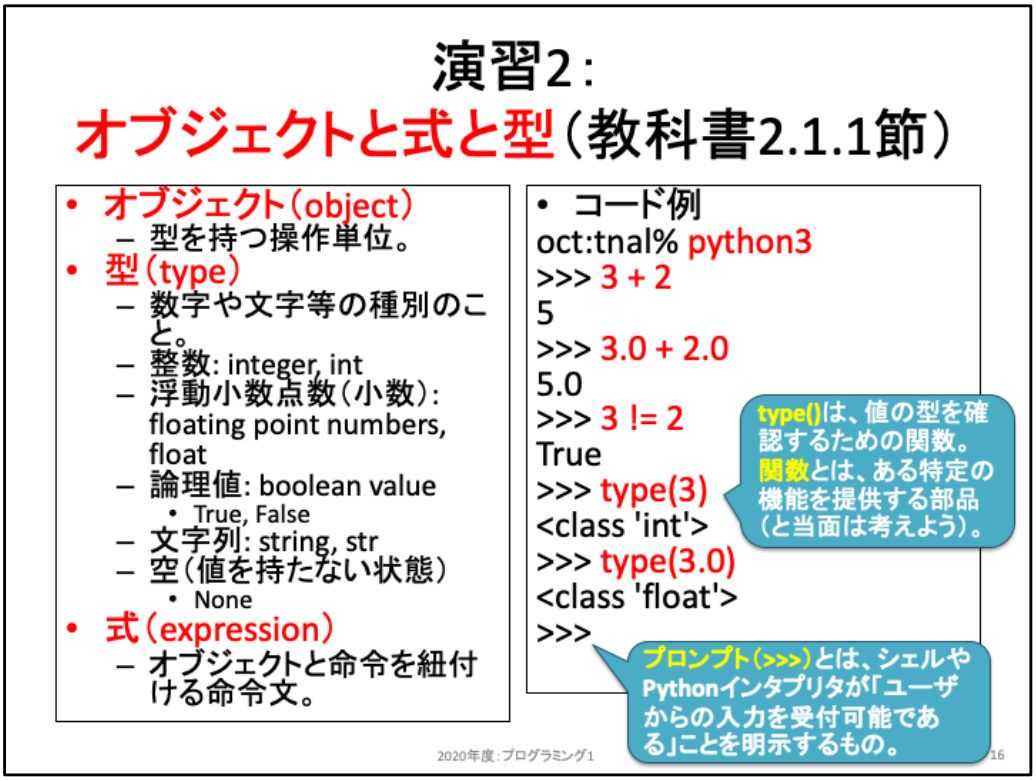
プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2: オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

ターミナル＋Pythonインタプリタを使った作業工程に慣れよう。Python2と3の違いに注意。

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



コンピュータで扱う具体的な値はリテラルと呼ばれている。数字の1や文字列の"hello"などのことをリテラルと呼ぶ。リテラルは必ず何かしらの「型」を持っており、あらゆる演算・式・評価といった処理は型に応じてその動作が決定される。

例えば、
>>> 1 + 1
上記は数字の1と数字の1を足し合わせている。
これに対し、
>>> 1 + '1'
上記は「数字の1」と「文字列の1」を足し合わせようとしている。が、そのような動作は定義されておらず、コンピュータから妙な結果が返ってくることになる。

型を把握することはとても大切であり、ここでは基本的な型として int, float, str をひとまず覚えておき、必要に応じて教科書を参照して別の型についても調べられるようにしておこう。また、型の調べ方として type() を覚えておこう。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2: オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

代表的な型（int, float, str）、型の確認方法と演算子を覚えよう。

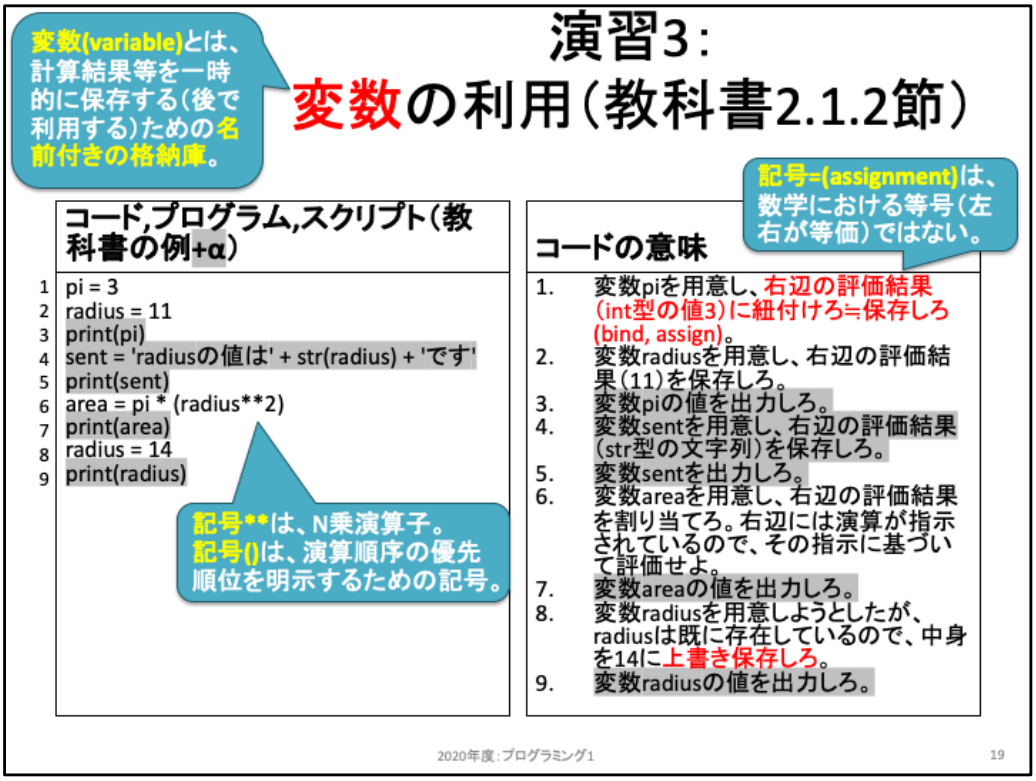
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

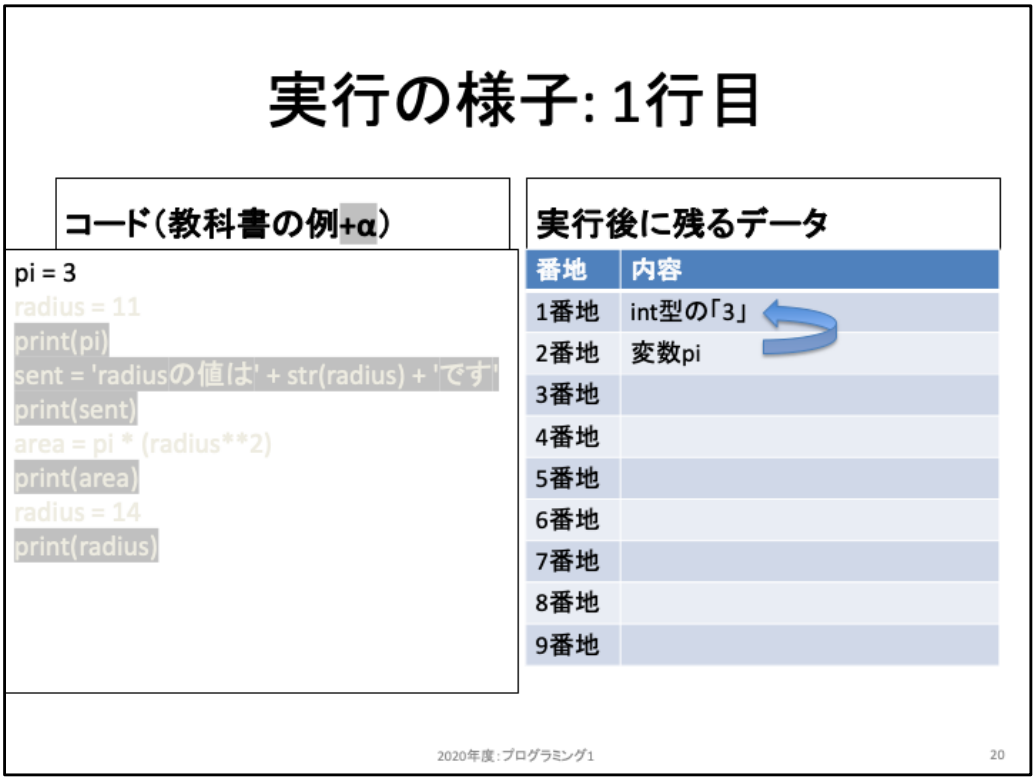
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



計算結果を一度しか使わないのであれば、変数はなくても構わない。そうでない場合、例えば「あるゲームの上位100人対してプレゼントを配布する」という処理をしたい場合には、その上位100人をメモしておき、一人ずつ処理する流れをとる、、、といった場合にはそのメモを参照しやすいように残しておきたい。このメモのことをプログラムでは変数と呼び、何かしらの名前をつけて計算結果を保存しておくことができる。

なお、変数名は「英数字やunderscore」を組み合わせて使うことが一般的だが、冒頭に数字を用いることはできない点に注意すること。(実際に試してみよう)

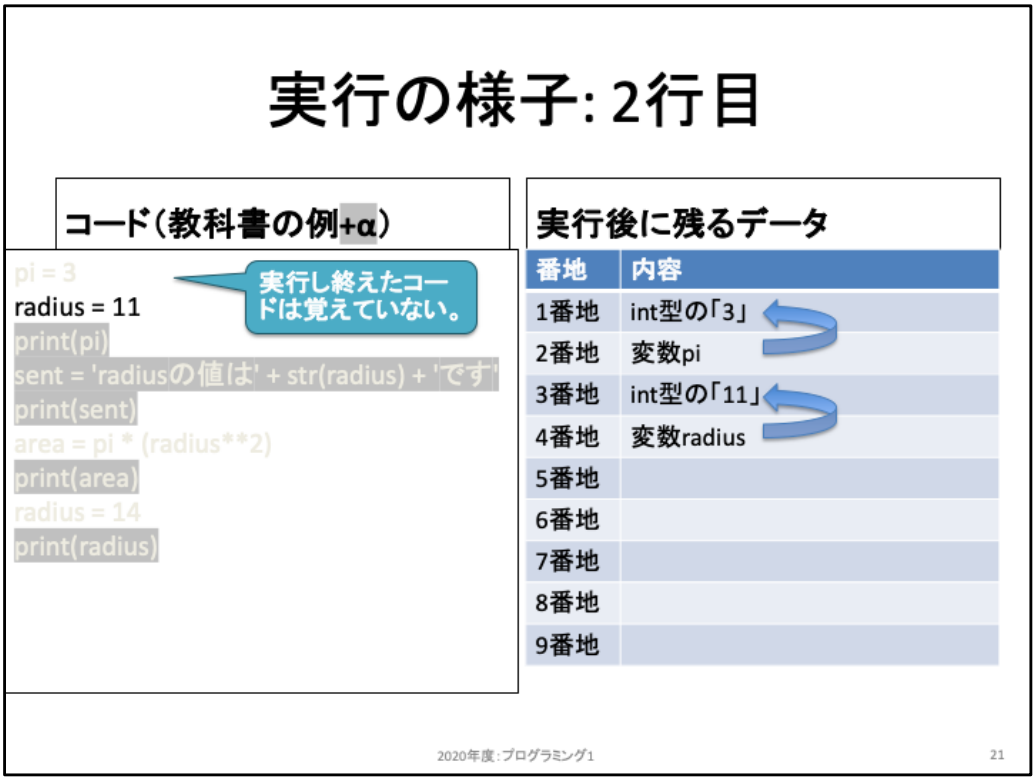
左側のコードに対応する形で、各コードの意味を右側に列挙している。これを理解できるようになろう。より具体的な動作は次のページ以降を参照しよう。(分からない点は質問しよう)



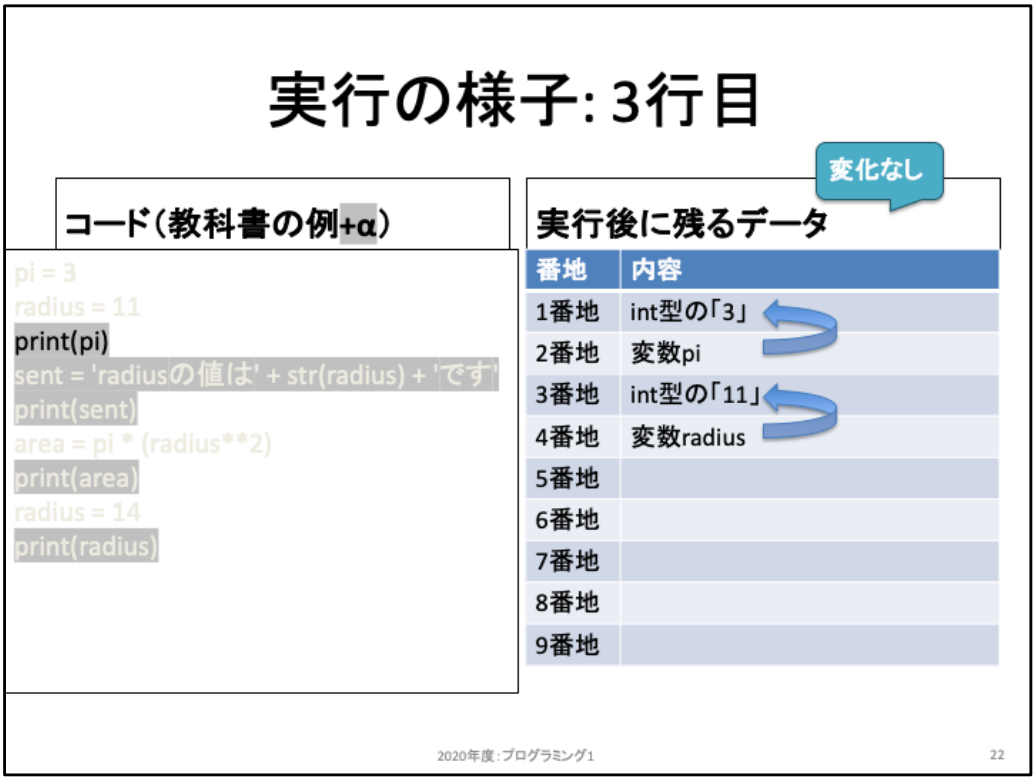
コンピュータ内部において、どのようにプログラムが処理されていくかについてイメージしよう。

コンピュータにおいて「何かを保存する」ためには、大別してストレージとメモリが使われる。ストレージは普段は使わないデータを格納しておく場所であり、メモリはプログラム実行中のデータを格納しておくために利用される。どちらも「どこに格納しているか」という住所を持っており、これを右スライドでは「1番地～9番地」として列挙している。本来ならば1箇所について保存できる情報のサイズが定められているが、ここでは簡易的に「何か一つの情報」が保持できるものとしよう。

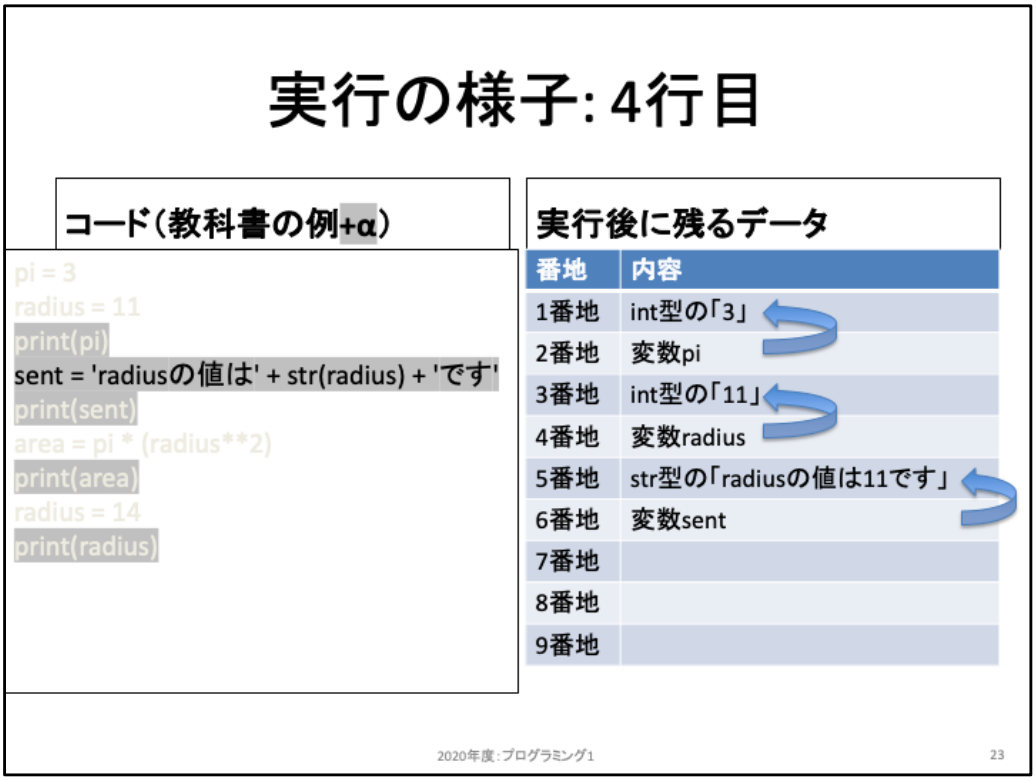
1行目のコード「pi = 3」では、まず「=演算子」が出てきているため、その右辺が評価される。ここで評価するとは「何かしら演算する内容があれば、それがなくなるまで演算を繰り返す」ことを指す。例えば「pi = 1 + 2 + 3」と書かれているならば、「1と2を足して、その結果に3を足した結果6になる」のが評価結果である。最初のコードでは、まず評価結果を1番地に保存し、次に変数名を2番地に保存し、最後にその変数が1番地を指すようためのリンク（参照と呼ぶことが多く、図では矢印で表している）を作成している。ここまでが1行のコードで実行される。


実行の様子: 2行目
コード（教科書の例+α）
pi = 3
radius = 11
print(pi)
sent = 'radiusの値は' + str(radius) + 'です'
print(sent)
area = pi * (radius**2)
print(area)
radius = 14
print(radius)
実行し終えたコードは覚えていない。
実行後に残るデータ
番地
内容
1番地
int型の「3」
2番地
変数pi
3番地
int型の「11」
4番地
変数radius
5番地
6番地
7番地
8番地
9番地
2020年度：プログラミング1
21

1行目同様。



3行目はprint()関数が命令として記述されているため、ターミナル上に何かを出力することになる。ここでは変数piが指定されているため、その中身が出力されるはずだ。



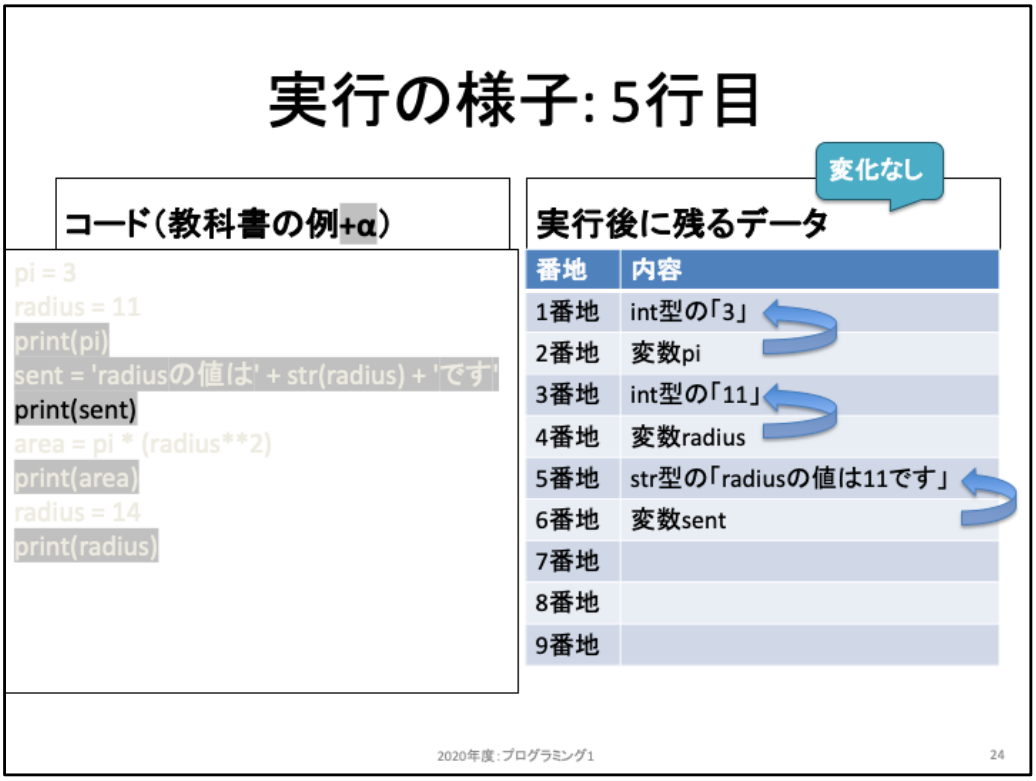
見た目はややこしいが、基本的な動作は1行目や2行目と同じ「変数名 = 演算を含む内容」であり、右辺を評価師を得た結果を変数sentに保存する。

手順ごとに分けて説明する前に、ここでは新しい2つの要素が出てきている。
(1) str()関数。これは指定された中身をstr型に変換するための関数である。
(2)「'文字列' + '文字列'」という文字列同士を結合するための+演算子。型がstrの場合には。結合になる。

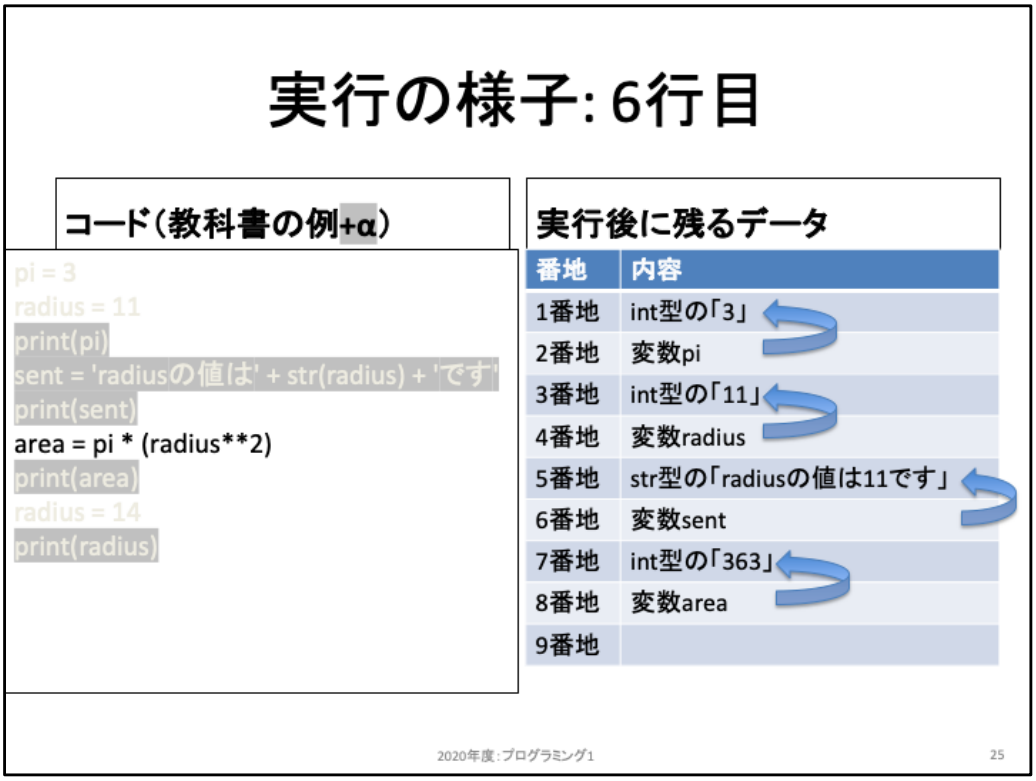
上記を踏まえると、まず最初の文字列'radiusの値は'はそのまま残り、次のstr関数の部分では変数radiusに保存されているint型の11をstr型に変換している。その結果、'radiusの値は'という文字列と'11'という文字列同士を結合する処理を実行することになり、'radiusの値は11'という文字列を生成する。最後にこの文字列に'です'を結合することで'radiusの値は11です'という文字列を生成する。この文字列が変数sentに保存される。

良く分からない場合には、「コードを分解」してみよう。具体的には、
>>> 'radiusの値は' + str(radius)
がどうなるだろうか？
>>> str(radius)
はどうなるだろうか？？

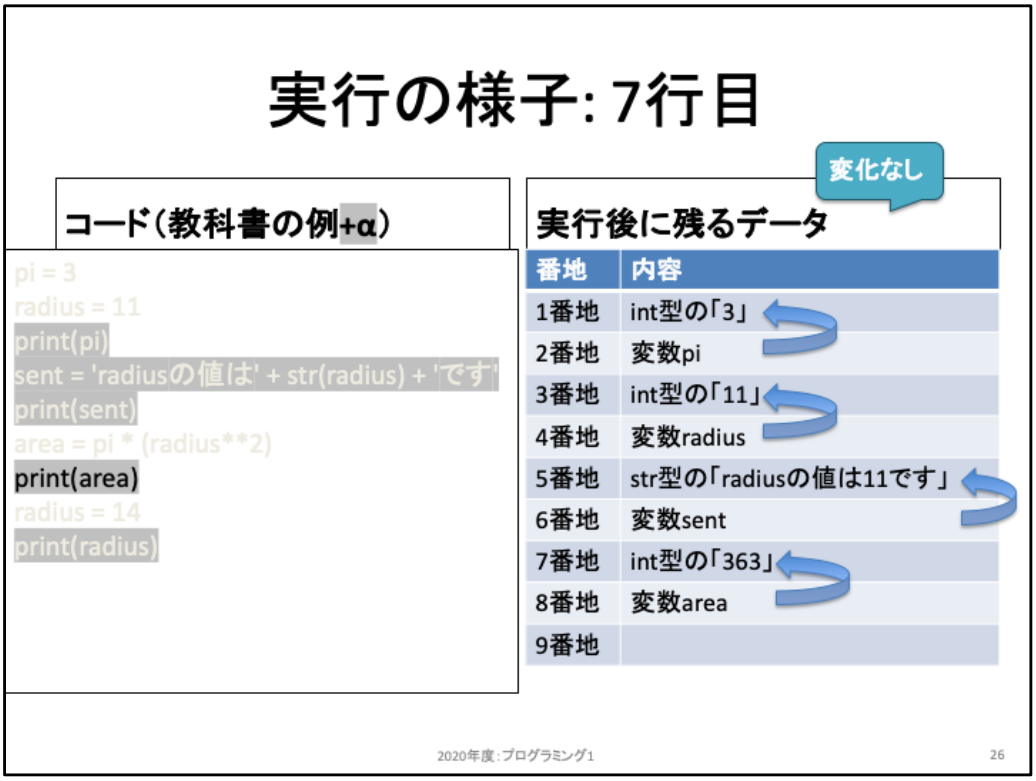
このように「分かる粒度になるまで分解する」のが基本的な考え方である。


実行の様子: 5行目
変化なし
コード（教科書の例+α）
pi = 3
radius = 11
print(pi)
sent = 'radiusの値は' + str(radius) + 'です'
print(sent)
area = pi * (radius**2)
print(area)
radius = 14
print(radius)
実行後に残るデータ
番地
内容
1番地
int型の「3」
2番地
変数pi
3番地
int型の「11」
4番地
変数radius
5番地
str型の「radiusの値は11です」
6番地
変数sent
7番地
8番地
9番地
2020年度：プログラミング1
24

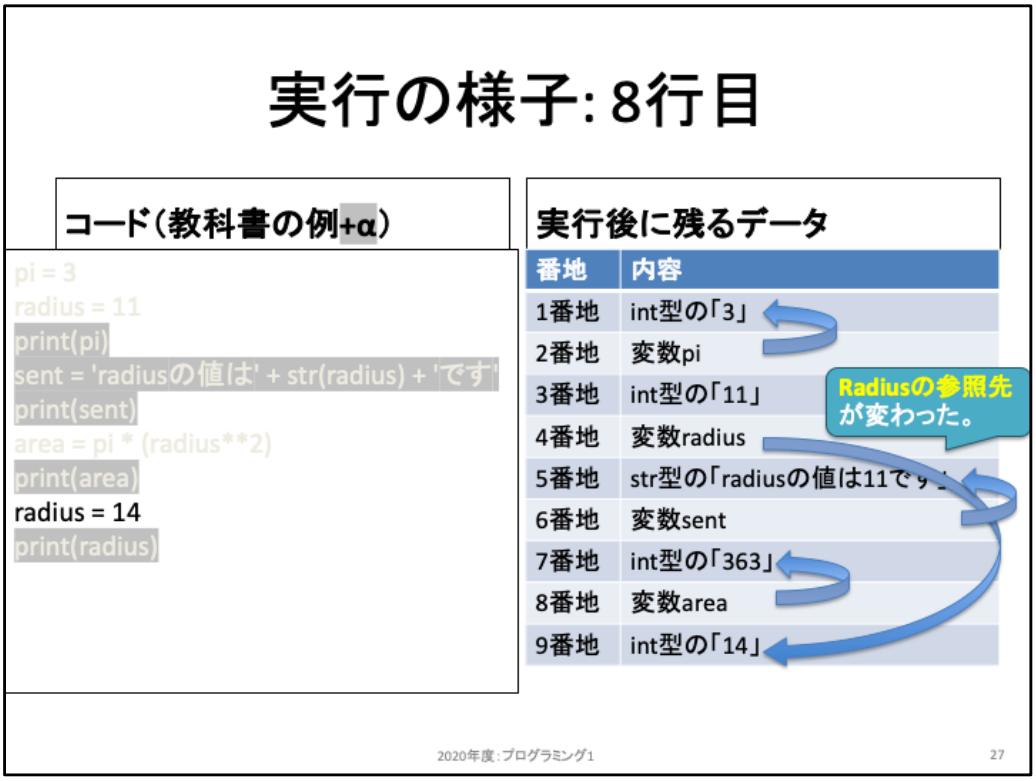
出力するのみ。



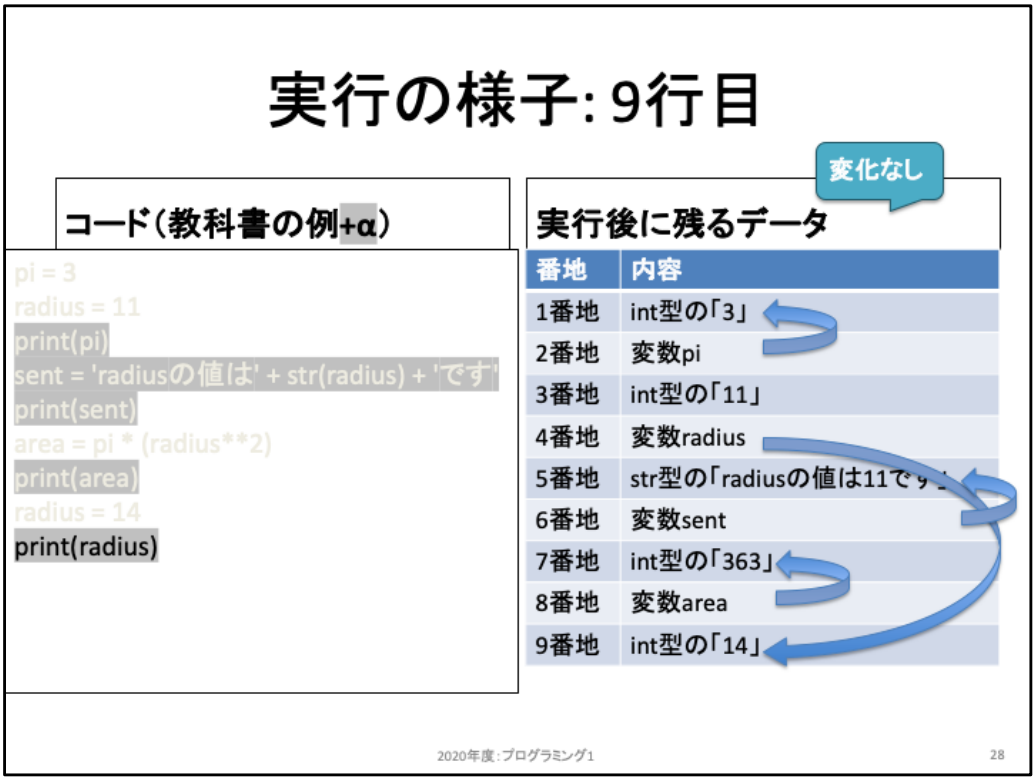
4行目のコードと同様に考えてみよう。注意点として、()がある場合にはそこの処理を優先すること。これは数学と同一の考え方である。


実行の様子: 7行目
コード（教科書の例+α）
pi = 3
radius = 11
print(pi)
sent = 'radiusの値は' + str(radius) + 'です'
print(sent)
area = pi * (radius**2)
print(area)
radius = 14
print(radius)
変化なし
実行後に残るデータ
番地
内容
1番地
int型の「3」
2番地
変数pi
3番地
int型の「11」
4番地
変数radius
5番地
str型の「radiusの値は11です」
6番地
変数sent
7番地
int型の「363」
8番地
変数area
9番地
2020年度：プログラミング1
26

出力するのみ。



この行だけを単独で解釈すると、単に「イコールの右側の評価結果であるint型の14を、変数radiusに保存する」となる。実際そのとおりに動作するのだが、これは高校までに習う数学の考え方とは大きく異なるため、違和感を感じる人もいるだろう。数学ならば、2行目の「radius = 11」により、変数radiusはどのようなことがあろうとも常に11である。それなのに8行目では「常に14である」と、矛盾した方程式を書いているようにみえる。しかしプログラムにおける「=」は方程式ではなく、「右辺の評価結果を変数に保存しろ」というだけである。このため、何度でも変数の中身を上書きして変更することができる。このことを右側では「radiusの参照先が変わった」と表現している。


実行の様子: 9行目
変化なし
コード（教科書の例+α）
pi = 3
radius = 11
print(pi)
sent = 'radiusの値は' + str(radius) + 'です'
print(sent)
area = pi * (radius**2)
print(area)
radius = 14
print(radius)
実行後に残るデータ
番地
内容
1番地
int型の「3」
2番地
変数pi
3番地
int型の「11」
4番地
変数radius
5番地
6番地
変数sent
7番地
int型の「363」
8番地
変数area
9番地
int型の「14」
2020年度：プログラミング1
28

出力するのみ。

覚えてほしいこと

- 等号(=)の意味
  - 数学では、等式。「左辺と右辺は等価である」という意味。
  - Python(プログラム)では、assignment(割り当て)やbinding(紐付け)のことを指し、一時的に名前の付いた保管庫へ保存する動作のこと。
  - 次のように動作する。
    - (1)右辺をevaluate(評価≒実行)し、
    - (2)その結果を左辺の変数に保存する。

- 実行後に残らない内容
  - 何をどう処理したかは記録に残らない。
- 実行後に残るデータ
  - コードを実行することで「得られた値」が一時的に残る。
  - 「得られた値」は、等号を使って変数に紐付ける(保存する)ことで、再利用することができる。
- 逐次処理
  - 順番通りに実行する。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

プログラミングにおける等号は、(1)右辺を評価(実行)して、(2)結果を変数に保存する。

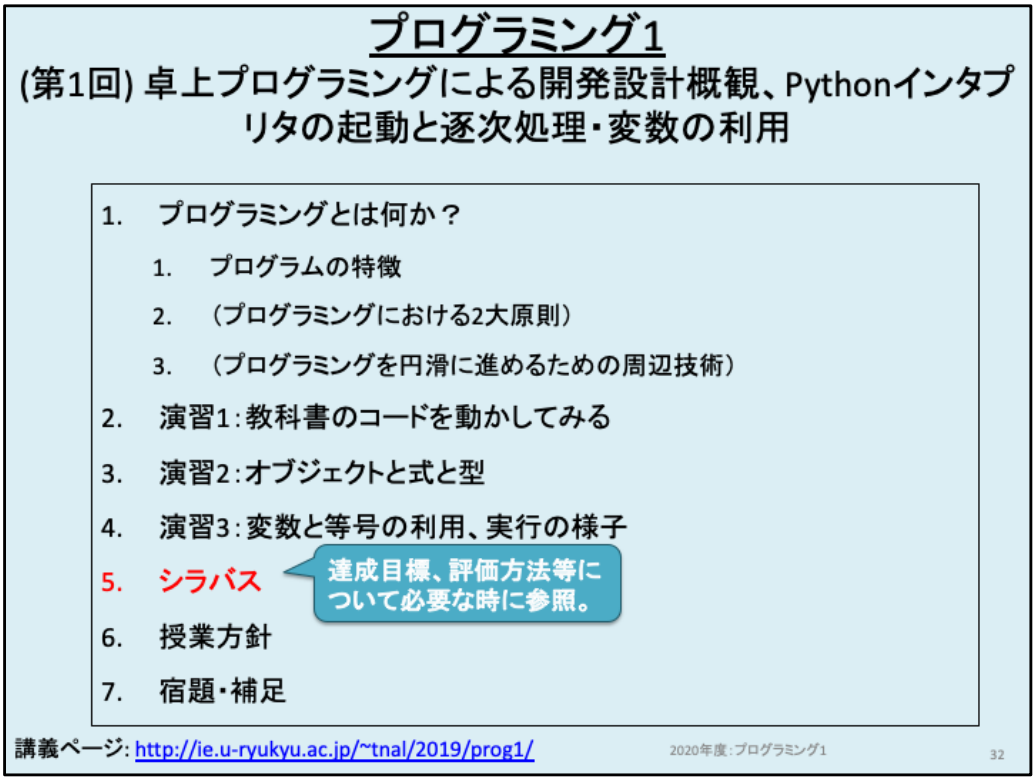
講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



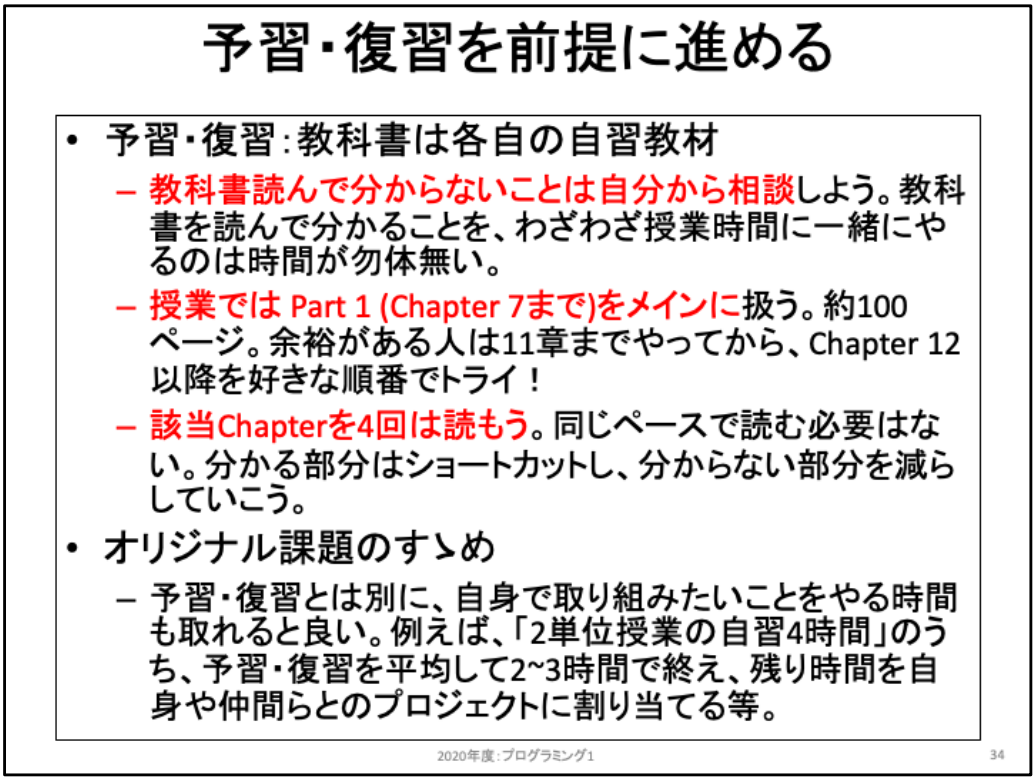
後日おいおい説明していきます。

プログラミング1

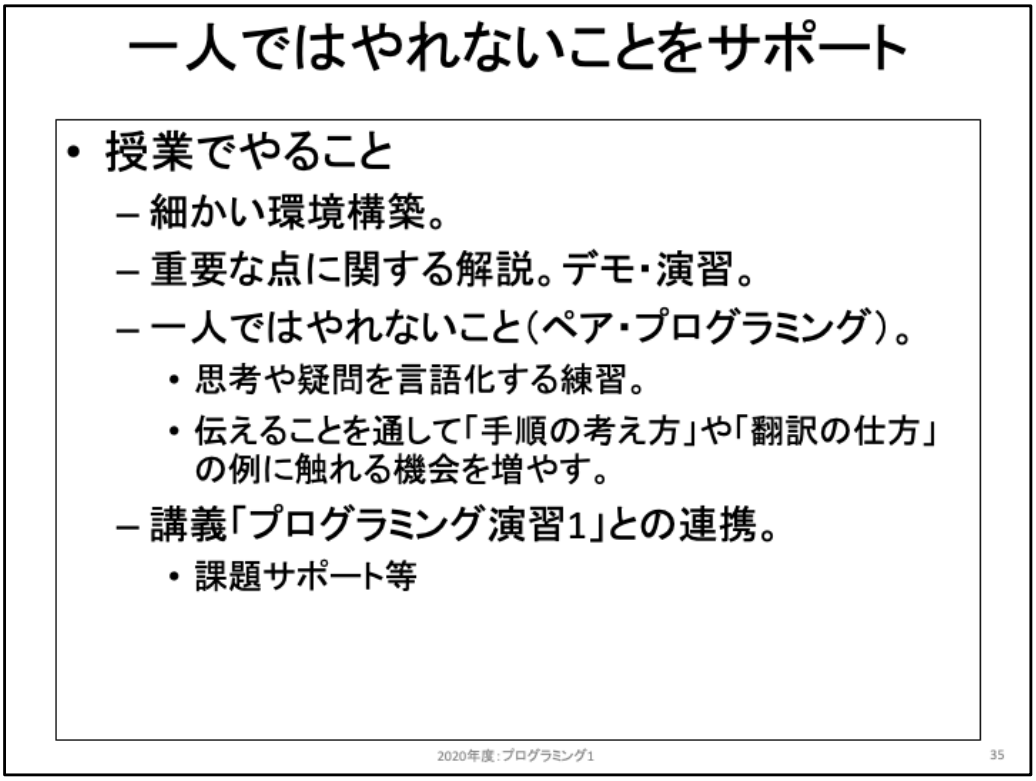
(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/



例年であれば友人らとのやり取りを強く推奨しているが、現時点では新型コロナウイルスの影響もあり、従来のやり方をそのまま踏襲することはまだ難しい。
Mattermost等何かしら工夫することを検討中。



ペアプログラミングに関しては、当面見送ります。

教育目標

- できるとは、「次の一歩」が分かること
- 授業方針
  - しばらくは「例示」。
  - 「次の一歩」が分かるまで手を貸す。
    - E.g., 2年次以降ではスマートフォンのアプリ開発したりしますが、そこで使う言語「Swift」等、新しい言語は独学になります。
    - 独学できる力、調べ方、トラブルシューティングの力、、、（＝「次の一歩」を検討する力）を学ぼう。正解があるとは限らない状況において、「次の一歩」を検討し、行動に移せるようになろう。

プログラミングを勉強する際のポイント

- 疑問を持とう
  - この単語はどういう意味？
  - このコードは何をやろうとしているのだろうか？
  - 何故こう書くのだろうか？
- やりたいことを文章や口頭で説明できるまで理解しよう
  - 説明できないことを「コードで書く」のは無理。
- 「こうなるだろう」と想像して実行しよう
  - 想像と異なる動作をした場合、それは学びを得るチャンス！　要因を調査しよう。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

疑問に感じた点は
どんどん質問しよう。

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/

プログラミング1

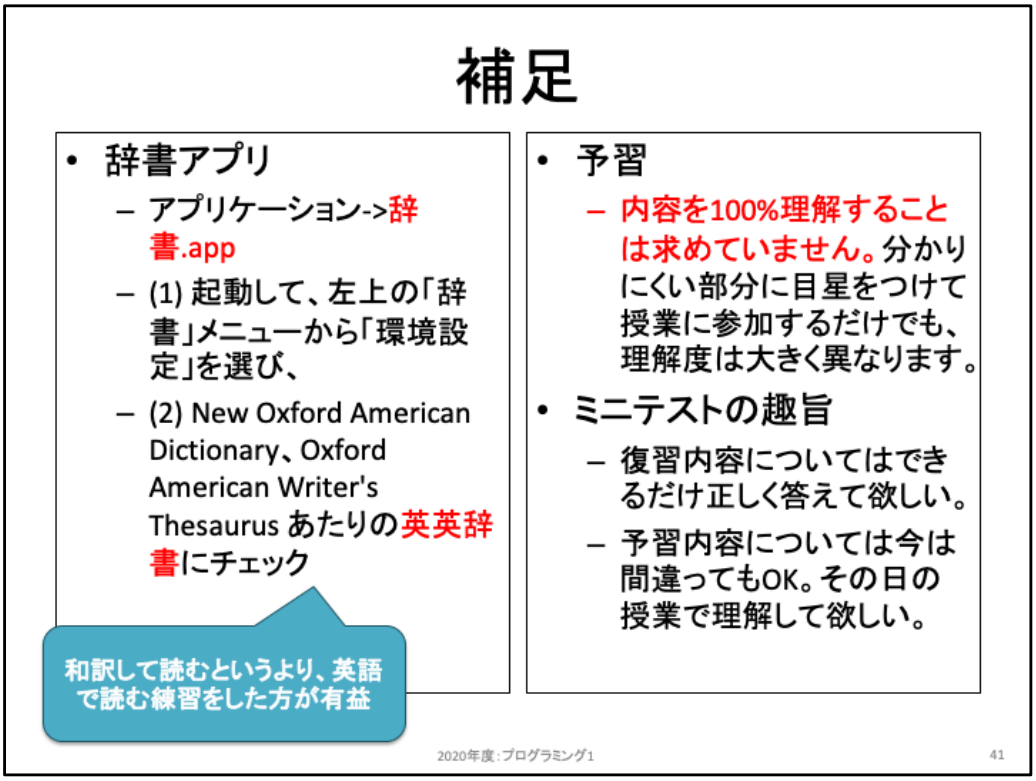
(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2019/prog1/

復習

- 復習: 教科書読み
  - 1章　＊概要掴むぐらい（30分程度）でok
  - 2章～2.1.2節まで　＊1～2時間程度想定
    - 2.1.3節はスキップ。
    - 余裕ある人は2.2節。
- 自習(オススメ)
  - progate
    - Python I, II
    - https://prog-8.com



ミニテストは3週目以降、授業開始後にやる予定です。

プログラミング1

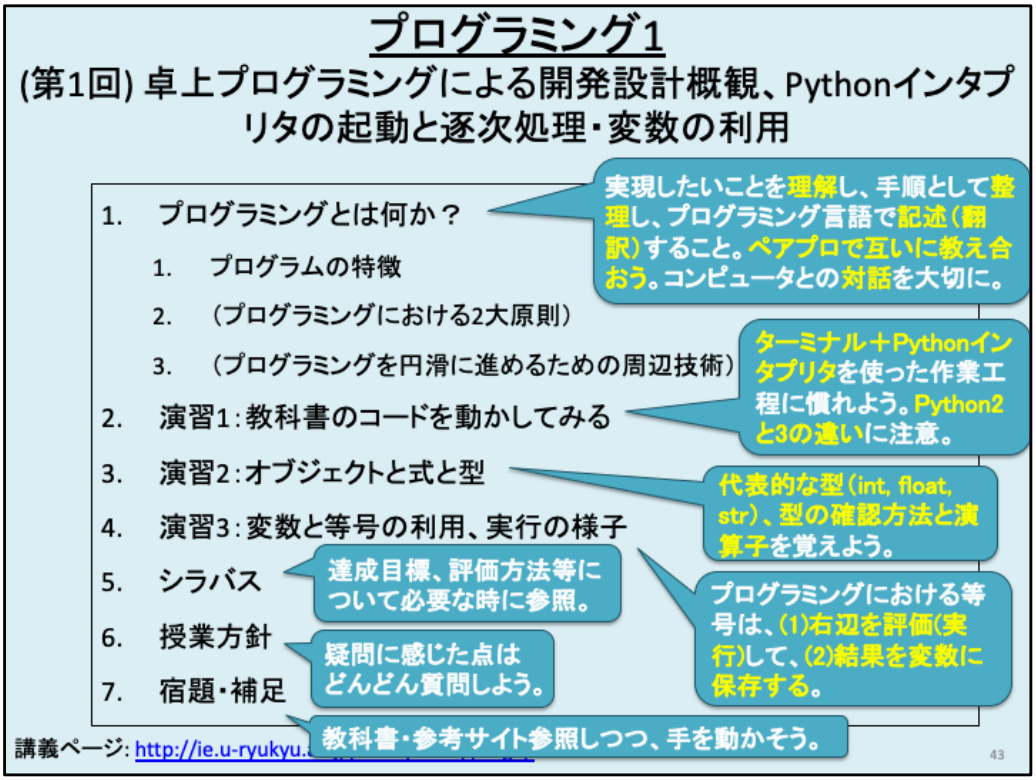
(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
    1. プログラムの特徴
    2. （プログラミングにおける2大原則）
    3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足

教科書・参考サイト参照しつつ、手を動かそう。

講義ページ: http://ie.u-ryukyu.

42


プログラミング1
(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用
1. プログラミングとは何か？
1. プログラムの特徴
2. （プログラミングにおける2大原則）
3. （プログラミングを円滑に進めるための周辺技術）
2. 演習1：教科書のコードを動かしてみる
3. 演習2：オブジェクトと式と型
4. 演習3：変数と等号の利用、実行の様子
5. シラバス
6. 授業方針
7. 宿題・補足
実現したいことを理解し、手順として整理し、プログラミング言語で記述（翻訳）すること。ペアプロで互いに教え合おう。コンピュータとの対話を大切に。
ターミナル＋Pythonインタプリタを使った作業工程に慣れよう。Python2と3の違いに注意。
代表的な型（int, float, str）、型の確認方法と演算子を覚えよう。
プログラミングにおける等号は、(1)右辺を評価(実行)して、(2)結果を変数に保存する。
達成目標、評価方法等について必要な時に参照。
疑問に感じた点はどんどん質問しよう。
教科書・参考サイト参照しつつ、手を動かそう。
講義ページ: http://ie.u-ryukyu.
43

参考文献

- 教科書: Introduction to Computation and Programming Using Python: With Application to Understanding Data
- 来たるべき、「みんな」のコードのために：平成4年生まれがつくるプログラマーの学校, http://wired.jp/2016/03/04/kusano-teacher/
- 20歳を過ぎてからプログラミングを学ぼうと決めた人たちへ, http://www.slideshare.net/ShuUesugi/20-9290892
- (progate) Python I, II: https://prog-8.com

時間があった場合のおまけ

翻訳に至る3ステップの例

プログラミングに含まれる3ステップ

- （コンピュータはプログラミング言語しか知らないので、）プログラマは、実現したいことをプログラミング言語に翻訳する必要がある。
  1. 「実現したいこと」の理解が大前提。この理解が不十分の場合、何を翻訳したら良いかが分からない。
  2. 「理解したこと」を手順として説明できるレベルまで整理する。
  3. 最後に、プログラムへ「翻訳（記述）」する。
     - 初学者は、プログラミング言語の構文・仕様を学ぶ必要がある。

一般的に、「プログラム」には3番目の翻訳結果しか残らない。1の理解や、2の整理が欠落してしまう傾向にある。レポートを書く時、また友人らと一緒に課題に取り組む際には、1,2の説明をするように心がけよう。

翻訳に至る3ステップの例 1/4

時間があれば最後に。

- Step 1: 実現したいこと
  - 受講生の中から一名をランダムに選ぶ。
- Step 2: 整理した手順
- Step 3: 翻訳

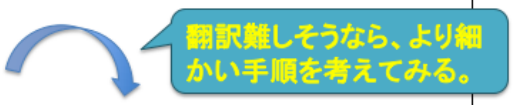
実現するためには、何をどの順番で実行したら良い？

翻訳に至る3ステップの例 2/4

- Step 1: 実現したいこと
  - 受講生の中から一名をランダムに選ぶ。
- Step 2: 整理した手順
  - 1. 受講生一覧を用意。
  - 2. 一覧をシャッフルする。
  - 3. シャッフルされた一覧から、頭の一人を抜き出す。
- Step 3: 翻訳

翻訳に至る3ステップの例 3/4

- Step 1: 実現したいこと
  - 受講生の中から一名をランダムに選ぶ。
- Step 2: 整理した手順
  - 1. 受講生一覧を用意。
    - 1-1. 受講生は学籍番号と一対一対応している。
    - 1-2. 学籍番号は205701～205767。
  - 2. 一覧をシャッフルする。
  - 3. シャッフルされた一覧から、頭の一人を抜き出す。
- Step 3: 翻訳



翻訳に至る3ステップの例 4/4

- Step 1: 実現したいこと
  - 受講生の中から一名をランダムに選ぶ。
- Step 2: 整理した手順
  - 1. 受講生一覧を用意。
    - 受講生は学籍番号と一対一対応している。
    - 学籍番号は205701～205766。
  - 2. 一覧をシャッフルする。
  - 3. シャッフルされた一覧から、頭の一人を抜き出す。
- Step 3: 翻訳

- Step 3: 翻訳

```
# 1. 受講生一覧を用意。
students = list(range(205701, 205767))

# 2. 一覧をシャッフル。
import random
random.shuffule(students)

# 3. 頭一人を抜き出す。
students.pop()
```

翻訳に至る3ステップの別例＆まとめ

- Step 1: 実現したいこと
  - E.g., 目玉焼きを作る。
- Step 2: 整理した手順
  - 1. フライパン、卵、油、皿を用意する。
  - 2. フライパンをコンロの上に起き、油を垂らして熱する。
  - 3. 卵を割ってフライパンの上に落とす。
  - 4. 1分でコンロを切り、予熱で仕上げる。
  - 5. 目玉焼きを皿にのせる。
- Step 3: 翻訳
  - Pythonの言語仕様に基づいて記述。

手順の考え方と翻訳技術を学ぶ

代表的な原則

- Done is better than perfect.
  - まずは動くものを。
- KISS原則, DRY原則
  - Keep it simple, stupid!
  - 小さく作る。(小さく分解して、組み合わせる)
  - Don't repeat yourself.
  - 繰り返しを避ける。

経験を積む

- 当面は決められた仕様を「どう実装するか」。
  - どう機能分解するか？
  - どんな手順になるか？
  - どう構造化するか？
- 少しずつ検討範囲拡大。
- ペア・プログラミング。
  - 互いに教え合う、言語化練習
  - 知識の共有化