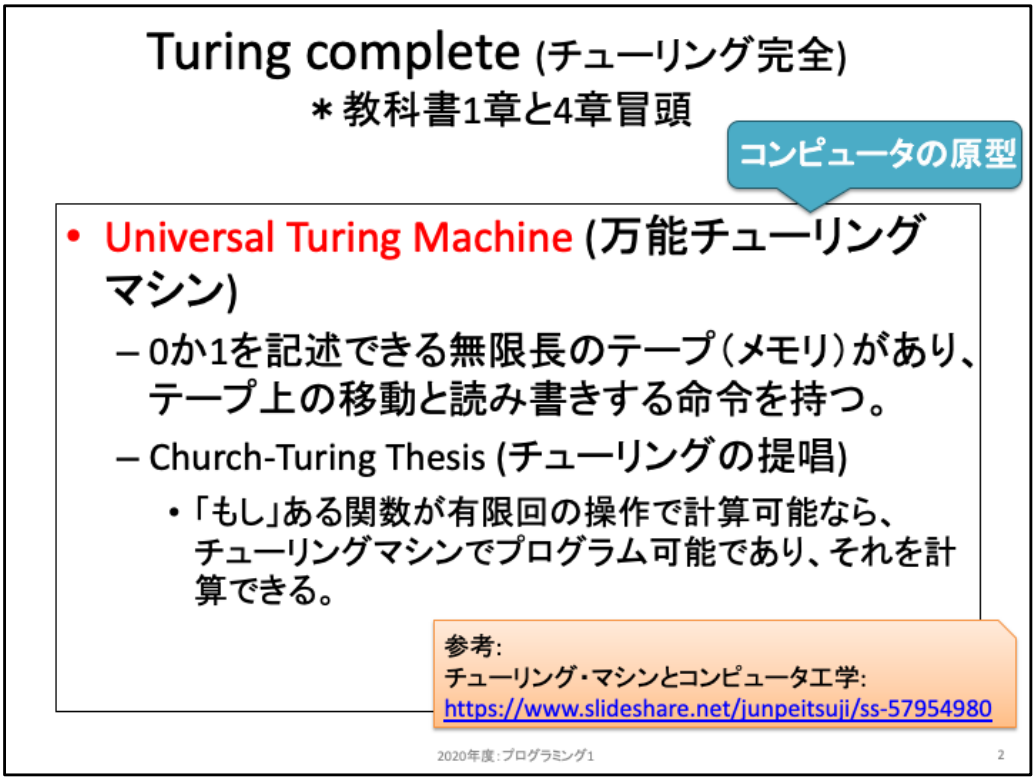
プログラミング1

(第6回) シーケンス集合・引数・スコープの補足

1. 万能チューリングマシン（イントロだけ）
2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. Chapter 4.1.2 Keyword Arguments and Default Values
5. Chapter 4.1.3 Scoping
6. 演習宿題

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



これまで述べてきたように、逐次処理・条件分岐・ループ処理（や関数定義）の組み合わせてあらゆるプログラムを実装できる。理論的には「万能チューリングマシン」と呼ばれるコンピュータ原型があり、「無限長のテープ」と、そのテープを移動する命令と、テープへ読み書きする機能があれば、【有限回の操作で計算できる処理】は全て記述することができる。これに対してプログラミング言語として用意されている逐次処理・条件分岐・ループ処理・関数定義、またstr型やlist型といった型は、これらがあると便利という意味で追加されている機能に過ぎない。例えばstr型なしに文字列を扱うプログラミング言語もある。

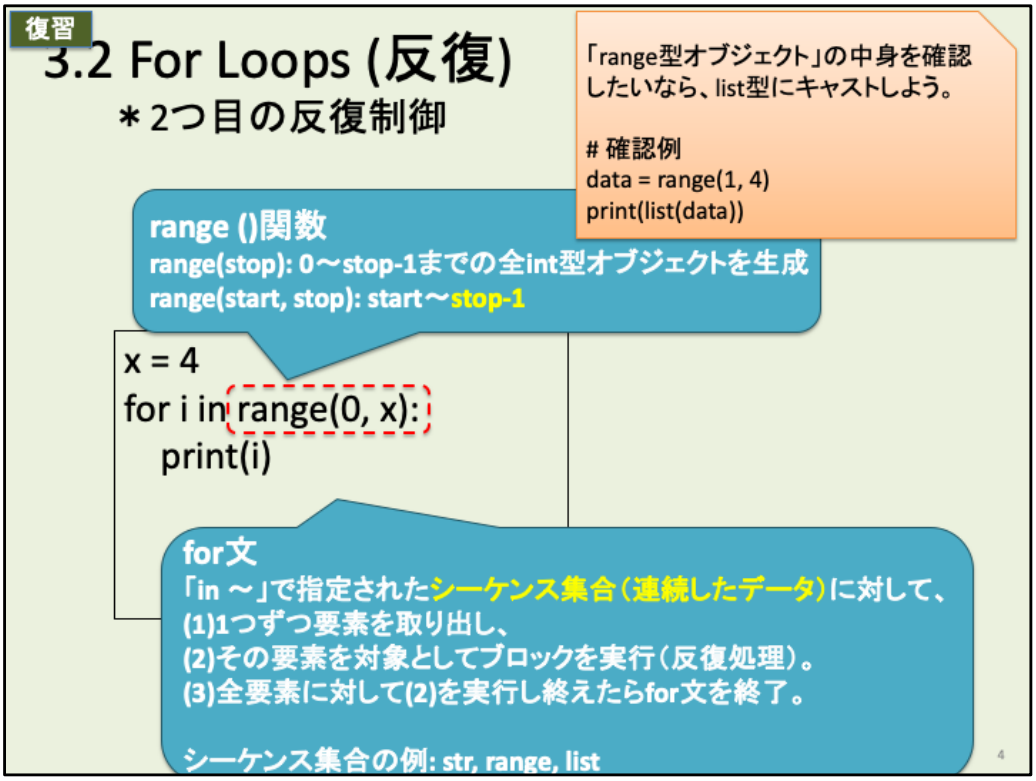
興味がある人はチューリングマシンについて調べてみよう。

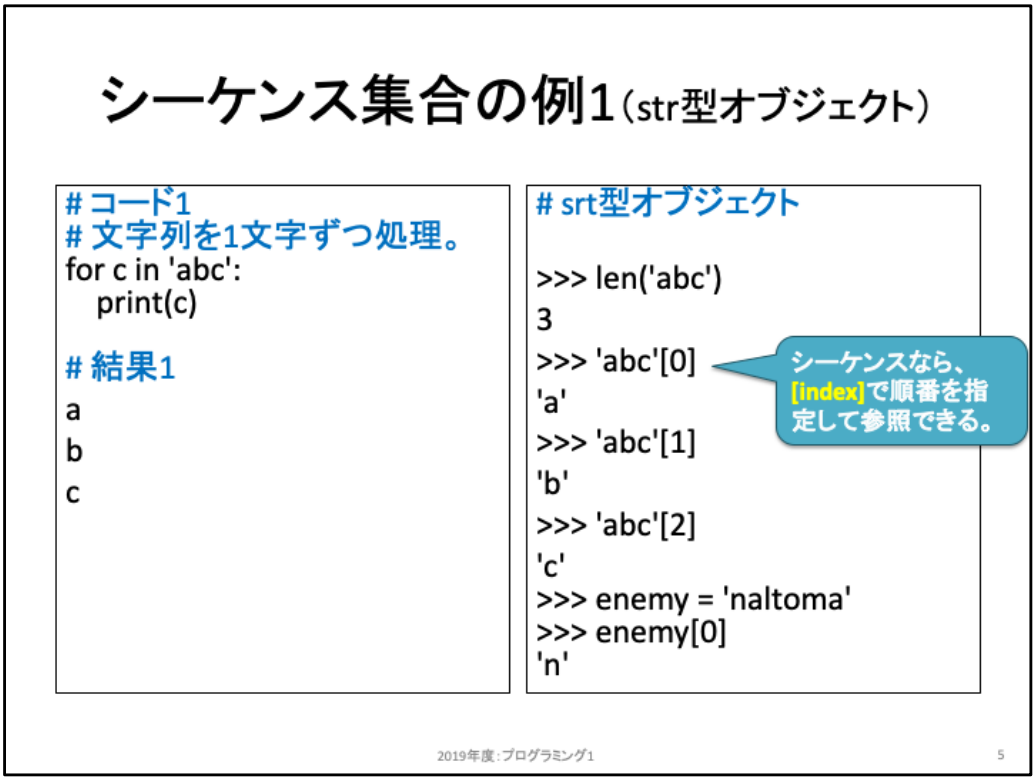
プログラミング1

(第6回) シーケンス集合・引数・スコープの補足、モジュール

コンピュータの原型である万能チューリングマシンについて調べてみよう。

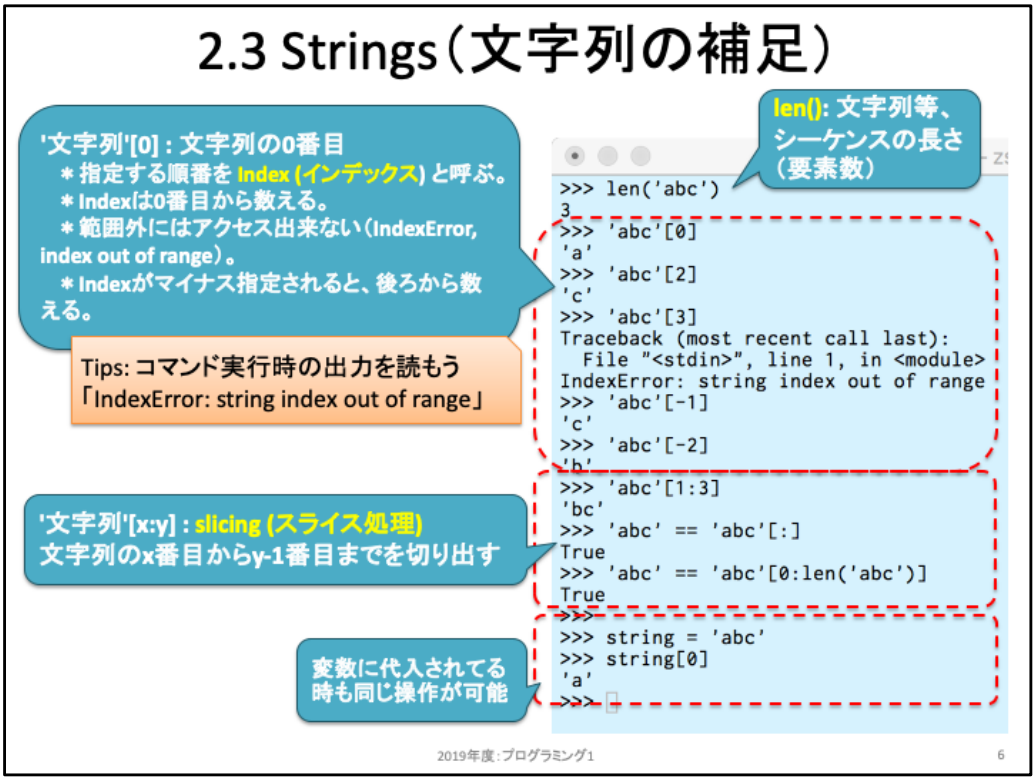
1. 万能チューリングマシン
2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. Chapter 4.1.2 Keyword Arguments and Default Values
5. Chapter 4.1.3 Scoping
6. 演習宿題


復習
3.2 For Loops (反復)
＊2つ目の反復制御
「range型オブジェクト」の中身を確認したいなら、list型にキャストしよう。
# 確認例
data = range(1, 4)
print(list(data))
range ()関数
range(stop): 0～stop-1までの全int型オブジェクトを生成
range(start, stop): start～stop-1
x = 4
for i in range(0, x):
print(i)
for文
「in ～」で指定されたシーケンス集合（連続したデータ）に対して、
(1)1つずつ要素を取り出し、
(2)その要素を対象としてブロックを実行（反復処理）。
(3)全要素に対して(2)を実行し終えたらfor文を終了。
シーケンス集合の例: str, range, list
4



str型オブジェクトは、順序の付いたシーケンスの例である。'abc'は、最初に'a'、その次に'b'、最後に'c'と並んでいる文字列であり、この順番が入れ替わってしまうと異なる文字列になるため順序が重要だ。このように順の付いた要素集合をシーケンスと呼ぶ。

これまでに出てきたシーケンスはlist型とstr型の2種類であり、どちらも、それ以外の場合であってもインデックスを使って「指定した順番に保存している要素」へアクセスすることができる。スライド右のコードは、'abc'というstr型オブジェクトに対して、[0]で0番目の要素を参照したり、[1]、[2]も同様にインデックス（順番）を指定して参照している。直接str型オブジェクトに対して使うだけではなく、それを保存した変数に対しても同様にインデックス参照が可能だ。



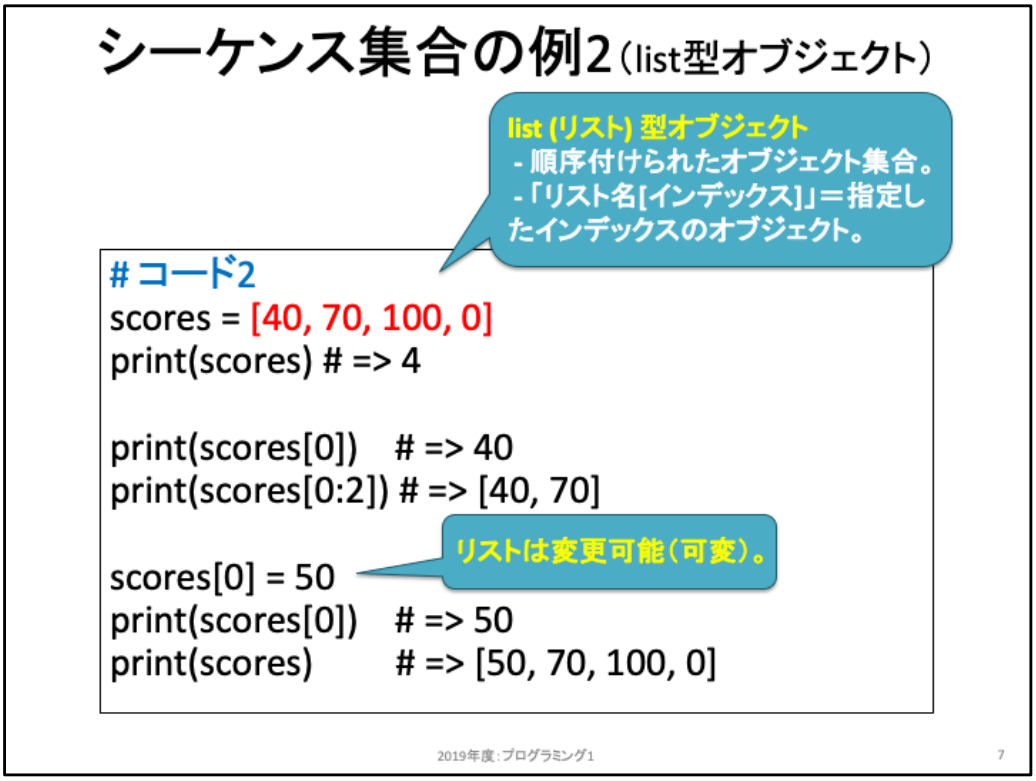
インデックスを指定する際、Python特有の記述が2つある。

1つ目は「後ろから数える」という指定方法であり、[-1]と指定すると「後ろから1番目」という意味になる。頭は0番目から数えるが、後ろから逆順に遡りたい場合には「-1番目、-2番目、-3番目、、、」のように指定できる。

これに対してlen関数を用いることで要素数を確認することができるため、この値を利用してインデックス指定することも可能だ。例えば len('abc') で 3 が得られる。このとき、一番最後の要素は0番目から数えると2番目にある。この2番目というインデックスは「要素数 – 1」であることから、次のようにして最後尾の要素を参照することができる。（他の言語ではこのような考え方で参照することが多い）

```
data = 'naltoma'
last = data[len(data) – 1]
print(last)
```

2つ目のPythonb特有記述はスライスと呼ばれており、「x番目からy番目まで」というように連続した複数要素を切り出したいときに使う記述方法が提供されている。前述のように変数dataにシーケンスが保存されているとすると、
data[始まるインデックス:終わりインデックス]
という形式で指定する。この際、始まるインデックスのデフォルト値（省略した際の値）は0であり、終わりインデックスのデフォルト値は len(data) である。このため、 data[:2] は「冒頭から2番目まで」を指定しており、'na'が得られる。data[1:]は「1番目から最後まで」であり、'altoma' が得られる。



先程はstr型オブジェクトに対するインデックス指定を見ていったが、同様の処理がlist型に対しても行える。

ただし1点だけ異なる点がある。それは、list型の要素は変更できるという点だ。上記コード例ではscoresに保存されているシーケンスの0番目を変更している様子を示している。同様のことをstr型オブジェクトに対して実行しようとすると、エラーになる。（確認してみよう）

このように、Pythonでは値を変更できるか否か、「mutable（変更可能）」「immutable（変更不可）」という属性が型に付与されている。これまでに出てきた int, float, str, list の中では str型だけが変更できない。それ以外は変更可能である。なお、ここでいう変更できないというのは「一部の要素を変更することができない」といっているだけだ。例えば、以下のコードは「全体を変更している」ため、エラーにはならない。

```
data = 'naltoma'
data = 'hoge'
```

プログラミング1

(第6回) シーケンス集合・引数・スコープの補足、モジュール

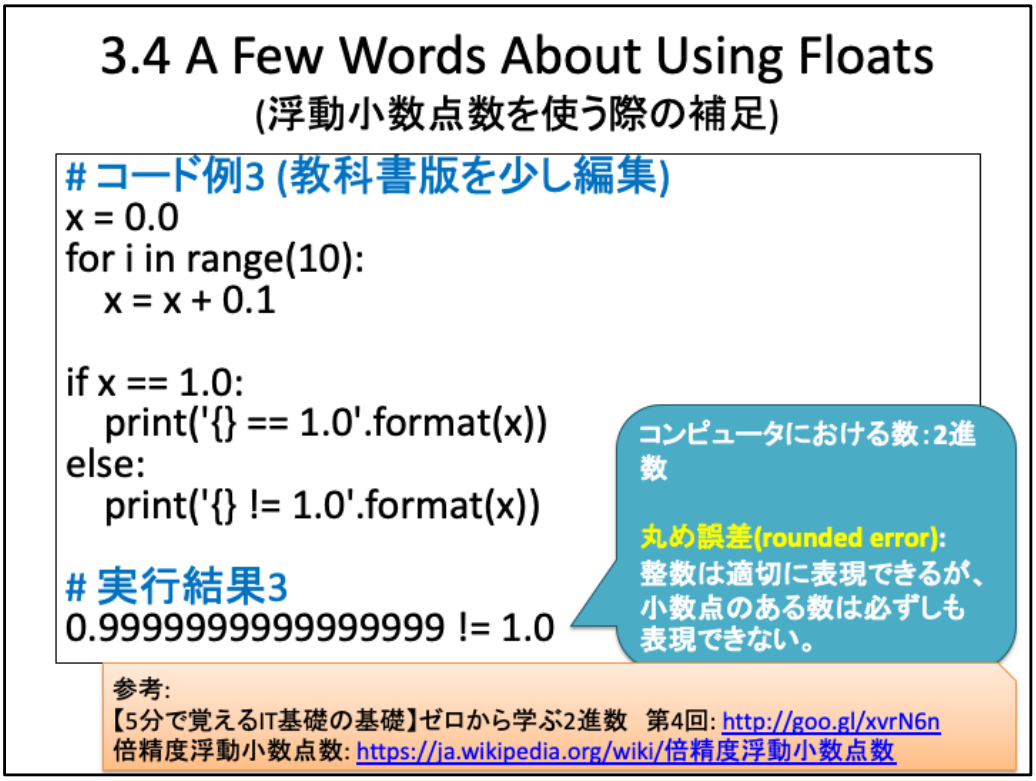
1. 万能チューリングマシン
2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. Chapter 4.1.2 Keyword Arguments and Default Values
5. Chapter 4.1.3 Scoping
6. 演習宿題

コンピュータの原型である万能チューリングマシンについて調べてみよう。

シーケンス集合に対してインデックスやスライスを使えるようになろう。

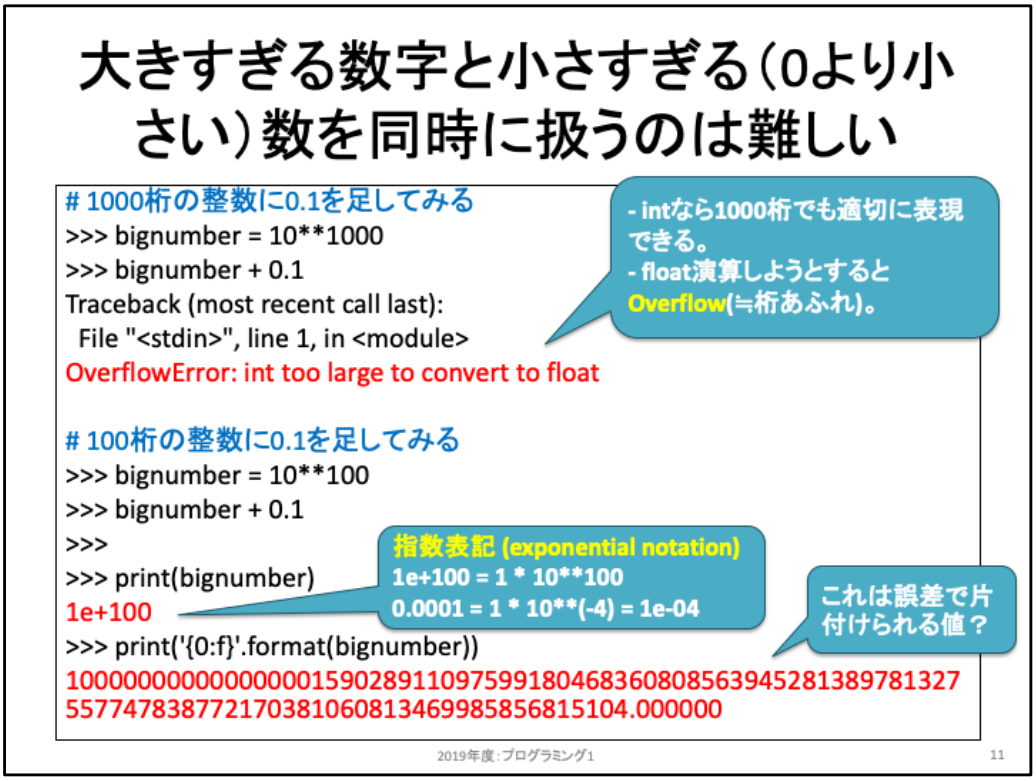
Chapter 3.4の補足

3.4 A Few Words About Using Floats



コンピュータは0,1の2進数ですべての情報を処理するため、少数点数を厳密に表現することは難しい。できなくはないが、プログラミング言語や書き方の工夫が必要であり、一般的には「このぐらいの精度で表現できていれば一般用途上問題ない」ように処理している。少数点数を表現する方法は「浮動小数点数」と呼ばれている。これは、例えば32bitで少数点数を表現しようとする際に、「1以上の数と1未満の数のバランス」を調整することで32bit内で表現しようとする。調整しようとしているのは、例えば小数点の位置を固定してしまうと「より大きな数」もしくは「より小さな数」を扱えなくなるためだ。このため、小数点の位置を固定するのではなく、可変にすることでバランスを取ろうとする。このように小数点の位置を固定していないため、「浮動小数点数」と呼ばれている。より詳細はスライド内リンク先を参照してみよう。

ここでは浮動小数点数を扱う際の問題点について紹介する。1つ目が丸め誤差だ。これは、小数点のある数は必ずしも厳密には表現できない(ことがある)ことを指している。スライド内のコード例では0.1を10回足しており、その結果は 1.0 になることを期待する。しかし 1.0 と等しいかどうかを確認すると False となり、print出力すると 0.99999,,, という数字が出力されてしまっている。



2つ目の問題点はオーバーフロー（桁あふれ）と呼ばれている。浮動小数点形式は固定小数点形式と比べると幅広い数値を扱うことが可能だが、それでも大きすぎる数字と小さすぎる数字を同時に扱うことは困難である。

スライドでは2つの例を示している。1つ目は「扱うことすらできないケース」であり、数値として保存することもできない。2つ目は、1つ目よりは小さな数値（100桁＋少数点数）を扱っているが、演算結果は想定と大きく異なっている。たかだか0.1を足しただけなのに、結果はそうなっていない。これは誤差と捉えても良いだろうか。

一般的には問題にならないことが多いが、このようなケースが起こりうることを認識しておこう。

プログラミング1

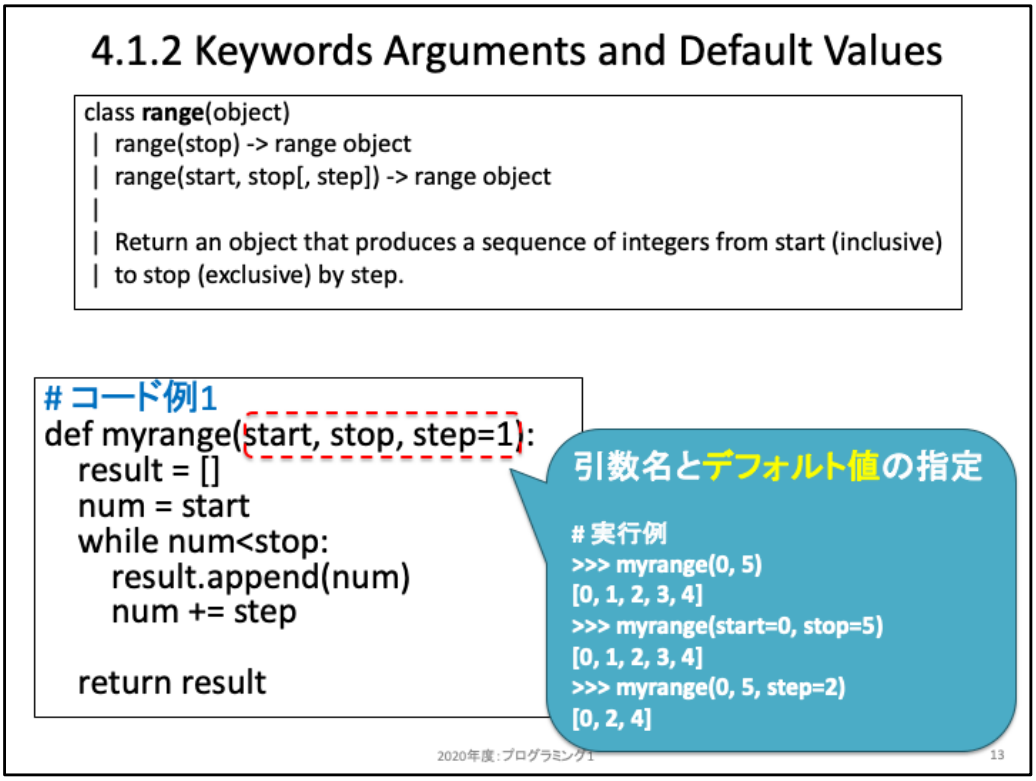
(第6回) シーケンス集合・引数・スコープの補足、モジュール

1. 万能チューリングマシン
2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. Chapter 4.1.2 Keyword Arguments and Default Values
5. Chapter 4.1.3 Scoping
6. 演習宿題

コンピュータの原型である万能チューリングマシンについて調べてみよう。

シーケンス集合に対してインデックスやスライスを使えるようになろう。

小数を使う際には丸め誤差と桁あふれに注意。



関数定義における補足として、引数にデフォルト値を設定することができる。繰り返しになるが、デフォルトとは指定しなかった際に採用するものであり、デフォルト値の場合には指定しなかった際に採用する値のことを指す。

スライド上のコード例では、def文において def myrange(start, stop, steo=1) と記述している。このうち start, stop はこれまでと変わらない引数の書き方である。3つ目の step=1 は、第3引数が省略された場合には step=1 として関数ブロックを処理することを指定している。第3引数を省略せず、例えば myrange(0, 5, 2) として書くと、step=2 と保存した状態で関数ブロックを処理する。

このようなデフォルト値を利用することで、「多くの場合に用いる値をそのまま利用するケース」と、「カスタマイズして利用するケース」とを同じ関数で使い分けることができるようになる。

プログラミング1

(第6回) シーケンス集合・引数・スコープの補足、モジュール

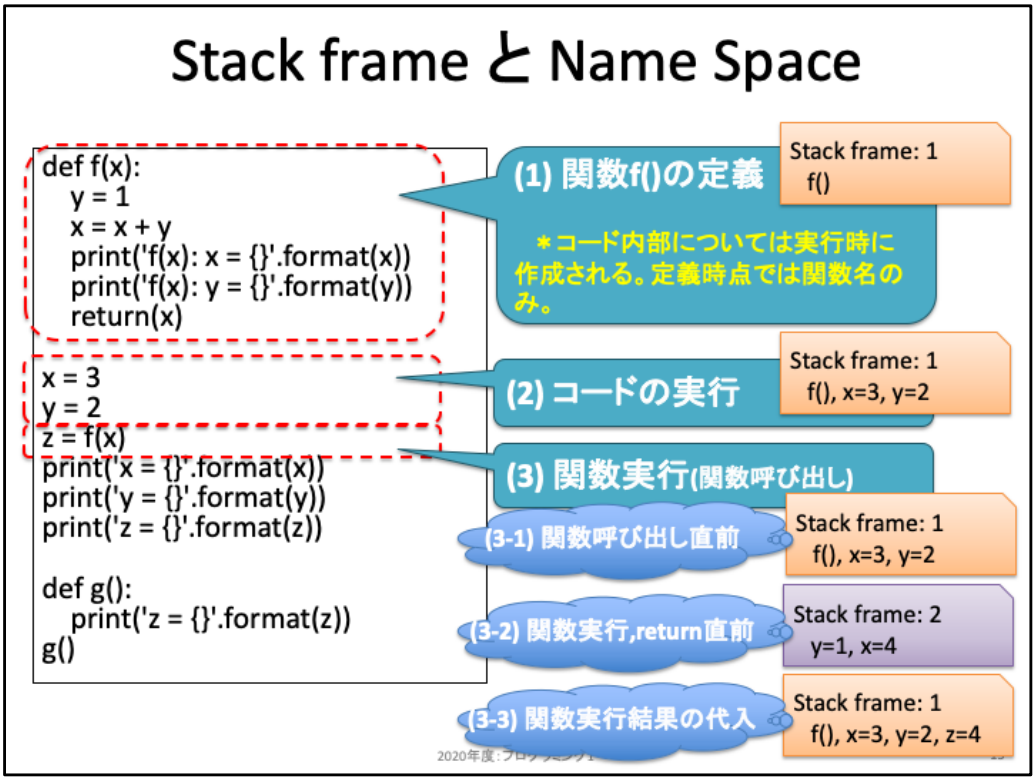
1. 万能チューリングマシン
2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. Chapter 4.1.2 Keyword Arguments and Default Values
5. Chapter 4.1.3 Scoping
6. 演習宿題

コンピュータの原型である万能チューリングマシンについて調べてみよう。

シーケンス集合に対してインデックスやスライスを使えるようになろう。

小数を使う際には丸め誤差と桁あふれに注意。

関数の引数にはデフォルト値を設定できる！



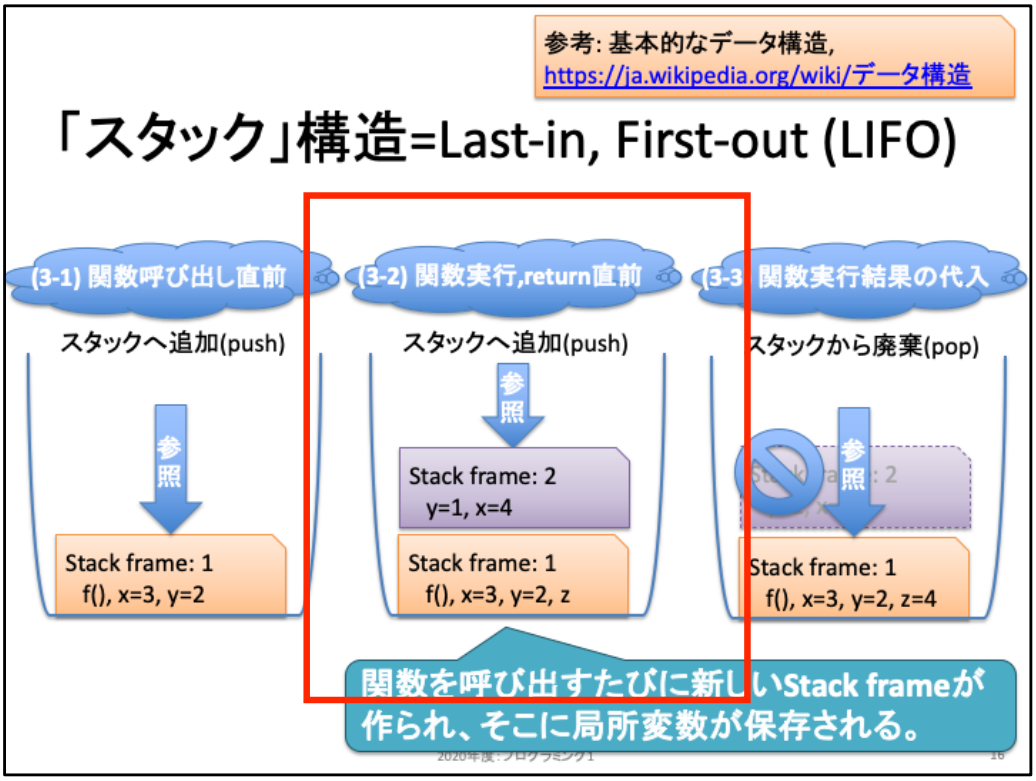
教科書4.1.3節のスコープについて詳細に眺めていこう。

このコード例を順序よく処理していくと、まずdef文により関数fが定義されている。ここでは定義しているだけのため、実行はせず、スタックフレーム上に関数fを記録しておくだけとなる。これまではメモリ上に関数やメモリを保存するといった説明をしてきたが、より厳密にはスタックフレームと呼ぶ。また、便宜上最初のスタックフレームを1と名付けておこう。

関数定義を読み終えると、スライド内(2)の2行の命令を実行する。ここでも同様にスタックフレーム1に変数x, y をそれぞれ保存することになる。

次のスライド内(3)の命令は関数呼び出しを指定しているため、まず関数の中に移動する。この「関数呼び出し」をする際には新しいスタックフレームを作成し、そこで呼び出された関数ブロックを処理しようとする。そのため新しく作成されたスタックフレーム内には、その呼び出し元となるスタックフレーム1とは独立して変数を利用することができる。スライド内の(3-1)～(3-2)がその様子を示しており、関数呼び出し直前のスタックフレーム1には関数f、変数x、yが登録されており、その状態で関数呼び出されるとスタックフレーム2が作成され、関数ブロックを処理してreturn文を実行する直前においては変数xとyだけが保存されている。また、スタックフレーム1のx,yと、スタックフレーム2のx,yは無関係である点に注意しよう。スタックフレーム2の中で（関数ブロックを処理している最中で）変数x,yをどのように処理したとしても、スタックフレーム1には影響を及ぼさない。

関数呼び出しを実行し終えると、その結果がreturn文で戻されているためその値を変数zに保存する。この際、関数実行が終わった時点でそのスタックフレームは破棄される。

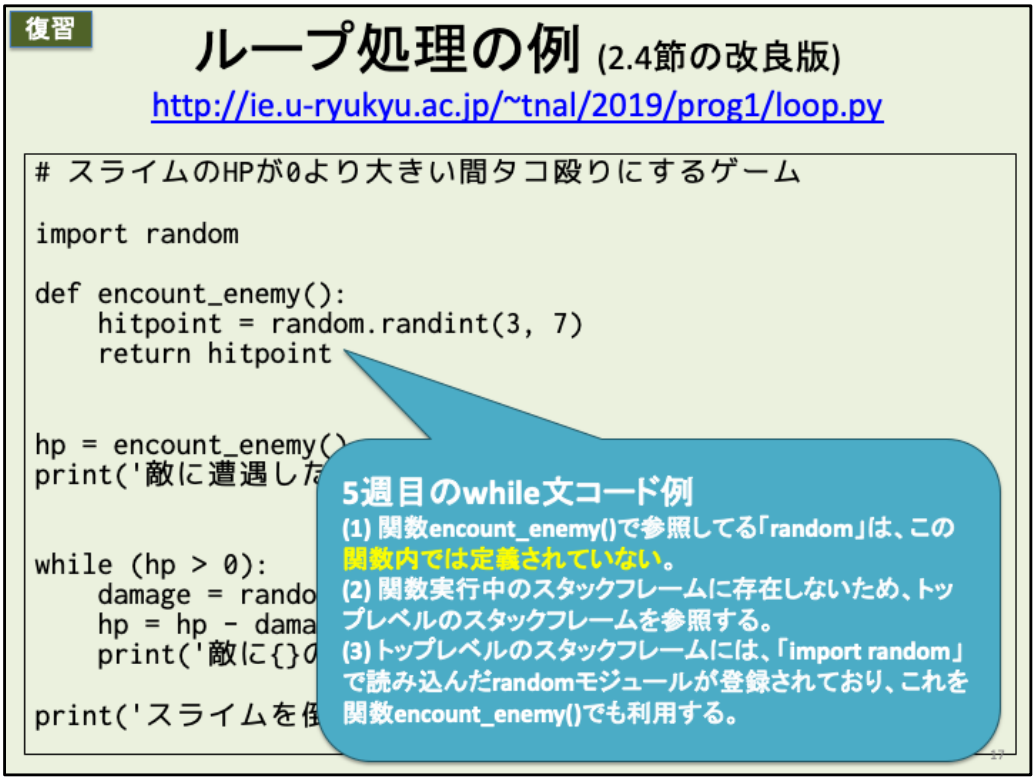


スタックフレームが構築され、破棄される流れを改めて確認してみよう。本スライドにおける(3-1)〜(3-3)は、前スライドに対応している。

(3-1)の状態では、スタックフレーム1に関数f、変数x,yが保存されている。このスタックフレーム1は、プログラム実行時に自動で生成されるスタックフレームである。例えば、関数呼び出しが一度もないプログラムを実行した場合には、全ての変数名はスタックフレーム1の中で処理されることになる。

(3-1)から関数呼び出しが行われると、スタックフレーム2が作成される。この時点で、スタックフレーム1の上に新しいスタックフレーム2が積み上げられている点に着目しよう。「フレーム（frame）」を英英辞書で調べると「a rigid structure that surrounds or encloses something such as a door or window.」と述べられており、何かしら囲まれた構造を意味する。「スタック（stack）」は専門用語であり、スライド内右上にあるスタック構造のことを指している。これはどのような構造かというと、本スライドで可視化しているように「箱」が積み重なっている構造のことを指す。下の箱の中身を確認するためには、上の箱をどかす必要がある。別の言い方をすると「最後に作られたスタックフレームから、参照される」特性を持つデータ構造のことを「スタック構造」と呼び、Last-in（最後に入れたもの）, First-out（最初に出される）という呼び方をする。より詳細は2年次の講義「アルゴリズムとデータ構造」で扱う。

変数の参照範囲を意味するスコープは、このスタックフレームと密接に結びついている。一般的には最上段にある最も新しいスタックフレームを参照する。ただし例外もあり、「もし現在参照しているスタックフレームに存在しない名称（変数・関数・モジュール等）があれば、スタックフレームを遡って参照する」ように動作する。この例を次のスライドで眺めてみよう。



このコードは前回のwhileループで利用した例である。

関数encount_enemyの中身を確認してみよう。ここでは random モジュールに登録されている関数 randint を利用している（呼び出している）。しかし、randomモジュールはこの関数の中にはどこにも定義していない。このため encount_enemy 関数が呼び出されて作成されたスタックフレーム内には random モジュールが記録されておらず、利用できない（NameError）になりそうだ。しかしながら、前スライドで述べた通り「もし現在参照しているスタックフレームに存在しない名称（変数・関数・モジュール等）があれば、スタックフレームを遡って参照する」ように動作するため、遡って関数呼び出し前のスタックフレームを参照し、そこには random モジュールが登録されているため、それを利用する。

このように、Pythonにおいてはスタックフレームを遡って参照することがあることを覚えておこう。

Stack Frame のポイント1

- トップレベル（インタプリタ起動時、もしくはスクリプトファイル実行時の最初のブロック）では、全ての関数名・変数名をトレースし、最初のスタックフレームで紐付けされる。
- 関数が呼び出されると、新しいスタックフレームが作られる。
  - ここでの処理は、スコープ外にあるスタックフレームには影響を及ぼさない。（例外あり）
  - 関数が終了すると、スタックフレームは廃棄(pop)される。
    - 廃棄しないとメモリに残り続るため、無駄。
  - 現スタックフレームに記録されていない名前が参照されると、「呼び出し元のスタックフレーム」を検索する。この遡り検索をトップレベルまで繰り返しても見つからない場合、NameErrorとなる。

スタックフレームをまとめると本スライドと次スライドのとおりとなる。

Stack Frame のポイント2

- スタックフレームとスコープに基づく名前空間の参照
  - あるコードで用いている名称のスコープはどこなのかを、スタックフレームとともに意識しよう。
  - 大原則
    - 関数を呼び出すと新しくスタックフレームが追加される。
    - 関数の処理を終えると、スタックフレームは破棄される。
    - 名前空間は現スタックフレームから遡って検索する。

プログラミング1

(第6回) シーケンス集合・引数・スコープの補足、モジュール

1. 万能チューリングマシン
   コンピュータの原型である万能チューリングマシンについて調べてみよう。
2. シーケンス集合とコード例
   シーケンス集合に対してインデックスやスライスを使えるようになろう。
3. Chapter 3.4 A Few Words About Using Floats
   1. 浮動小数点数の取り扱い
      小数を使う際には丸め誤差と桁あふれに注意。
4. Chapter 4.1.2 Keyword Arguments and Default Values
   関数の引数にはデフォルト値を設定できる！
5. Chapter 4.1.3 Scoping
   関数を呼び出しとスタックフレームの追加、名前空間の遡り参照を理解しよう。
6. 演習宿題

演習

演習1〜4: 初めてのペア・プログラミング

宿題

- 復習: 適宜(これまでの内容)
  - レポート課題2の良レポートの「良さ」を真似よう。
- 予習: 教科書読み
  - 4章
    - 4.3 Recursion
    - （4.4 Global Variables） ＊不要
    - 4.5 Modules
    - 4.6 Files

参考文献

- 教科書: Introduction to Computation and Programming Using Python: With Application to Understanding Data
- Python 3.5.1 documentation, https://docs.python.org/3.5/index.html
- チューリング・マシンとコンピュータ工学, http://www.slideshare.net/junpeitsuji/ss-57954980