~~鏖(37564)勉強会~~

Python勉強会@HACHINONE

第3章

Simple

簡単な算術プログラム

お知らせ

Python勉強会@HACHINOHEでは、ジョン・V・グッターグ『Python言語による
プログラミングイントロダクション』近代科学社、2014年をみんなで勉強しています。

この本は自分で読んで考えて調べると力が付くように書かれています。

自分で読んで考えて調べる前に、このスライドを見るのは、いわばネタバレを
聞かされるようなものでもったいないです。

是非、本を読んでからご覧ください。

第2版の翻訳

Python勉強会@HACHINOHE

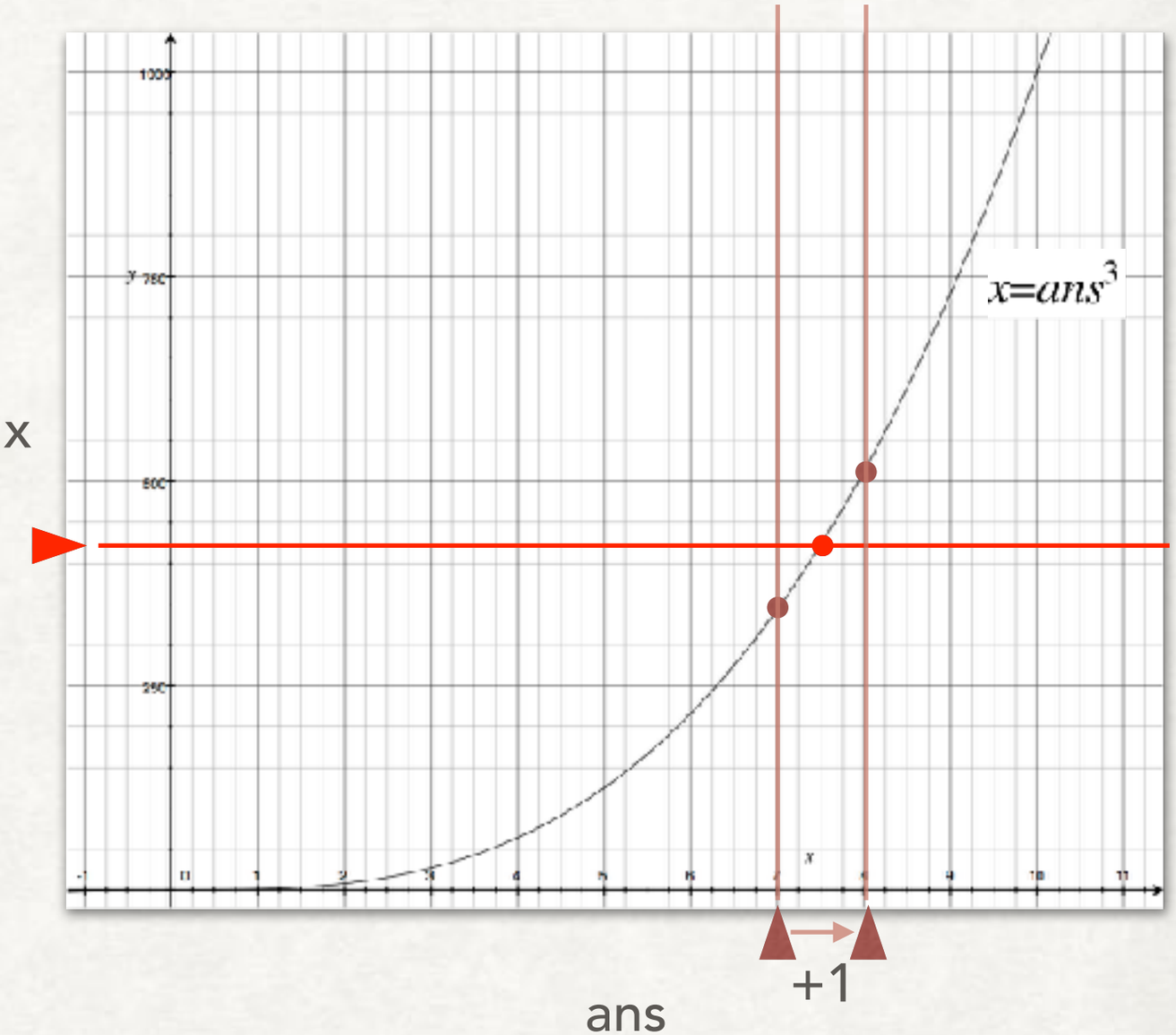
- ジョン・V・グッターグ『Python言語によるプログラミングイントロダクション第2版』近代科学社、2017年
- 9月1日発売
- 後半が大幅に増えた
- Python 3.5

1.さあ,始めよう！
2.Pythonの概要
3.簡単な算術プログラム
4.関数,スコープ,抽象化
5.構造型,可変性と高階関数
6.テストとデバッグ
7.例外とアサーション
8.クラスとオブジェクト指向プログラミング
9.計算複雑性入門
10.いくつかの単純なアルゴリズムとデータ構造
11.プロットとクラス
12.ナップサック問題とグラフ最適化問題
13.動的計画法
14.ランダムウォークと可視化
15.確率,統計とプログラム
16.モンテカルロ・シミュレーション
17.標本抽出と信頼区間
18.実験データの理解
19.無作為試験（無作為抽出試験）と仮説の照合
20.条件付き確率とベイズ統計
21.うそ,真っ赤なうそ,そして統計
22.機械学習はやわかり
23.クラスタリング
24.分類法
付録A Python 3.5 簡易マニュアル

立方根を見つける、ただし整数に限る

Python勉強会@HACHINOHE

- 試行錯誤法の一つ、総当たり
  - ある範囲を端から全部試す
  - 範囲と刻みがポイント
- xの立方根を求める
  - ansを0から1ずつ増やす
  - ans**3がx以上になるまで続ける
  - ans**3とxが等しければ整数の立方根が見つかった



総当たり法で完全立方数の立方根を見つけるコード

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
# 完全立方に対する立方根を求める
x = int(raw_input('整数を入力してください:'))
ans = 0
while ans**3 < abs(x):
    print '減少関数abs(x)-ans**3の値は' + str(abs(x) - ans**3)
    ans = ans + 1 # 1つずつ増やす

if ans**3 != abs(x):
    print x + 'は完全立方ではありません'
else:
    if x < 0:
        ans = -ans
    print str(x) + 'の立方根は' + str(ans) + 'です'
```

```
1957816251の立方根は1251です
```

```
740696101223634461 6の立方根は1949306です # 200万くらい
```

計算速度

Python勉強会@HACHINOHE

- 今日のコンピュータは、1秒間に1G(10**9。10億)命令以上、1ナノ秒以下で1命令実行できる
  - 1ナノ秒で光は30cm進む(光速は30万km/秒)
  - 音速340m/秒なので、音が34m進む間に1億(10**8)命令程度
- 参考: 今日のCPUは数十から数百G FLOPS(浮動小数点演算/秒)

```
# -*- coding: utf-8 -*-
max = int(raw_input('正の整数を入力してください:'))
i = 0
while i < max:
    i = i + 1
print i
```

```
$ time python code_3_1.py
正の整数を入力してください:1000000000
1000000000

real    1m8.703s
user    1m4.934s # Cだと2.7秒程度
sys 0m0.032s
```

指練習: 3.1節

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
num = int(raw_input('正の整数を入力してください:'))
find = False # 解が見つかったか否か(解が必ず見つかるなら不要)
pwr = 1
while pwr <= 5:
    root = 1
    while root <= num:
        if num == root ** pwr:
            print str(num) + '=' + str(root) + '**' + str(pwr)
            find = True
        root = root + 1
    pwr = pwr + 1
if find == False:
    print str(num) + '=root**pwrとなるrootとpwrの組はありませんでした'
```

```
正の整数を入力してください:6561
6561=6561**1
6561=81**2
6561=9**4
```

forループ

Python勉強会@HACHINOHE

- forループ

```
for 変数 in シーケンス※:
    コードブロック
```

※シーケンスはくりかえしの実行前に評価(実行)される

- range関数
  - リスト形式で等差数列を生成

```
range(初項，上限※，等差)
```

※厳密にはこの値未満の数列を生成

- break文
  - breakを含む最も内側のループを中断

forループで書いたコード

Python勉強会@HACHINOHE

- whileループよりもforループの方が、単純なくりかえしの場合は簡潔に書けることもある

```
# -*- coding: utf-8 -*-
# 完全立方に対する立方根を求める
x = int(raw_input('整数を入力してください:'))
for ans in range(0, abs(x) + 1): # 0からabs(x)の間に答えがあるはず
    if ans**3 >= abs(x):
        break

if ans**3 != abs(x):
    print x + 'は完全立方ではありません'
else:
    if x < 0:
        ans = -ans
    print str(x) + 'の立方根は' + str(ans) + 'です'
```

```
ans = 0
while ans**3 < abs(x):
    ans = ans + 1
```

指練習: 3.2節

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
s = '1.23,2.4,3.123' # 入力文字列
total = 0 # 合計
num = '' # 一時的に部分文字列を保存
for c in s:
    if c != ',': # ','でなければ
        # その文字を部分文字列へ追加する
        num = num + c
    else: # ','であれば
        if num != '': # 部分文字列が空でなければ(イレギュラーな入力をスルー)
            # 部分文字列を小数に変換して合計に加える
            total = total + float(num)
            # 部分文字列を空にする
            num = ''
# 最後の部分文字列を処理する
if num != '':
    total = total + float(num)
print total
```

```
# 試した文字列
s = '1.23,2.4,3.123'
s = ''
s = '1.23,2.4,,3.123'
s = '1,2.4,3.123'
s = '.23,2.4,3.123'
```

float関数を使わない誰得な指練習: 3.2節

Python勉強会@HACHINOHE

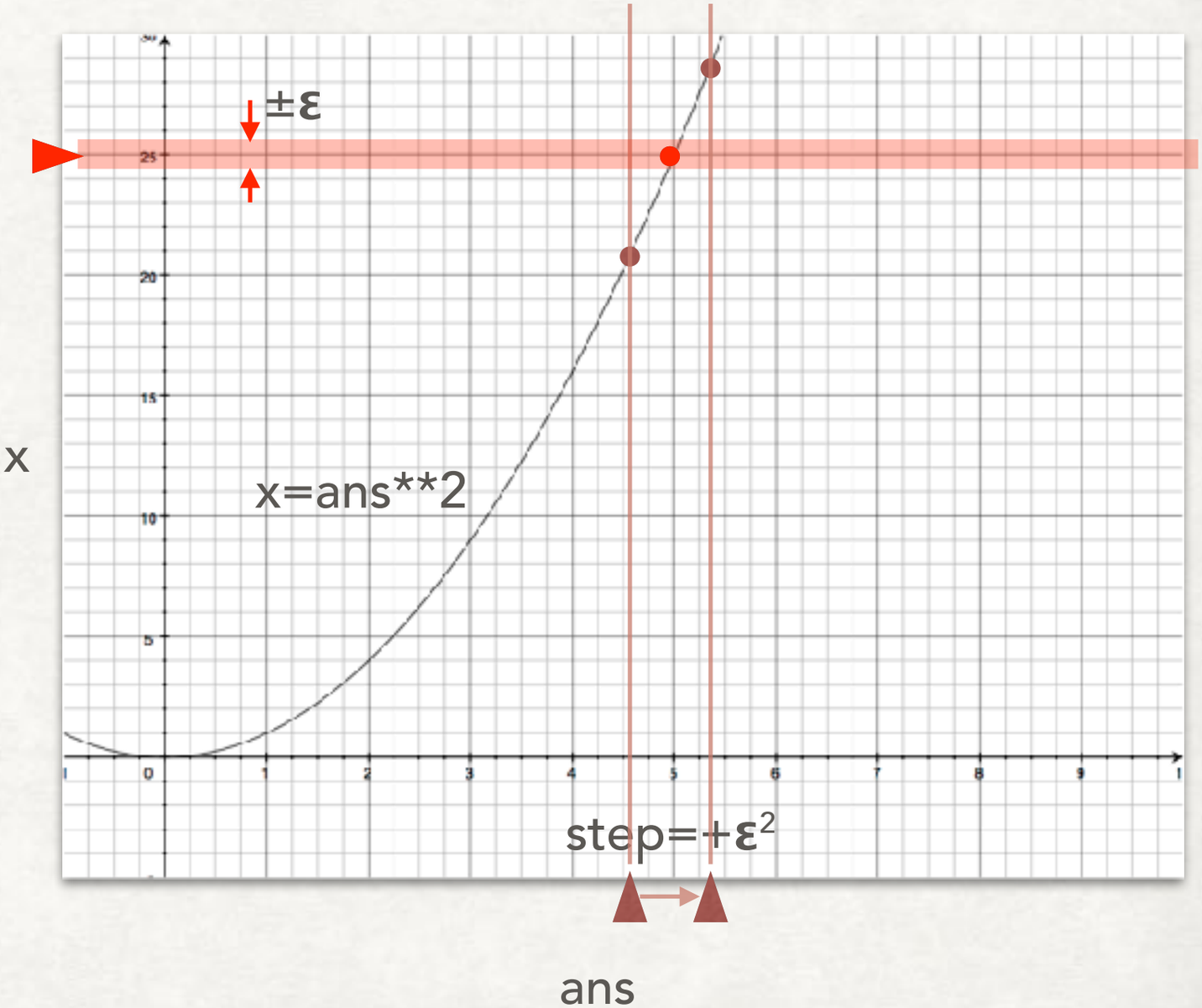
```
# -*- coding: utf-8 -*-
s = '1.23,2.4,3.123'
total = 0
state = '整数部' # 現在解析しているのが整数部か小数部か
power = 0 # 指数部
num = ''
for c in s:
    if c == ',':
        if num != '':
            total = total + int(num) * (10**power)
            num = ''
            state = '整数部'
            power = 0
    elif state == '整数部':
        if c == '.':
            state = '小数部'
        else:
            num += num + c
    else: # 小数部を解析しているとき
        num = num + c
        power -= 1

# 最後の部分文字列を処理する
if num != '':
    total = total + int(num) * (10**power)
print total
```

平方根の近似解法

Python勉強会@HACHINOHE

- 総当たり法
- ansをstepずつ増やして解を探索する
  - step = ε^2 = 0.0001
  - 探索範囲は0からx
    (x以上になったら終了)
- ans^{2}が精度の範囲内(x±ε)になったら、近似解が見つかったと判定
  - 精度 ε = 0.01
- stepが小さければ見つけられるが、時間はかかる



平方根を総当たり法で見つけるコード

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
x = 25 # 平方根を求めたい数

epsilon = 0.01 # 精度
step = epsilon**2 # 探索の刻み
numGuesses = 0 # 試行回数
ans = 0 # 近似解
while abs(ans**2 - x) >= epsilon and ans <= x:
    ans += step
    numGuesses += 1
print '試行回数=' + str(numGuesses)

if abs(ans**2 - x) >= epsilon:
    print str(x) + 'の平方根を見つけられませんでした'
else:
    print str(x) + 'の平方根の近似解は' + str(ans) +'です'
```

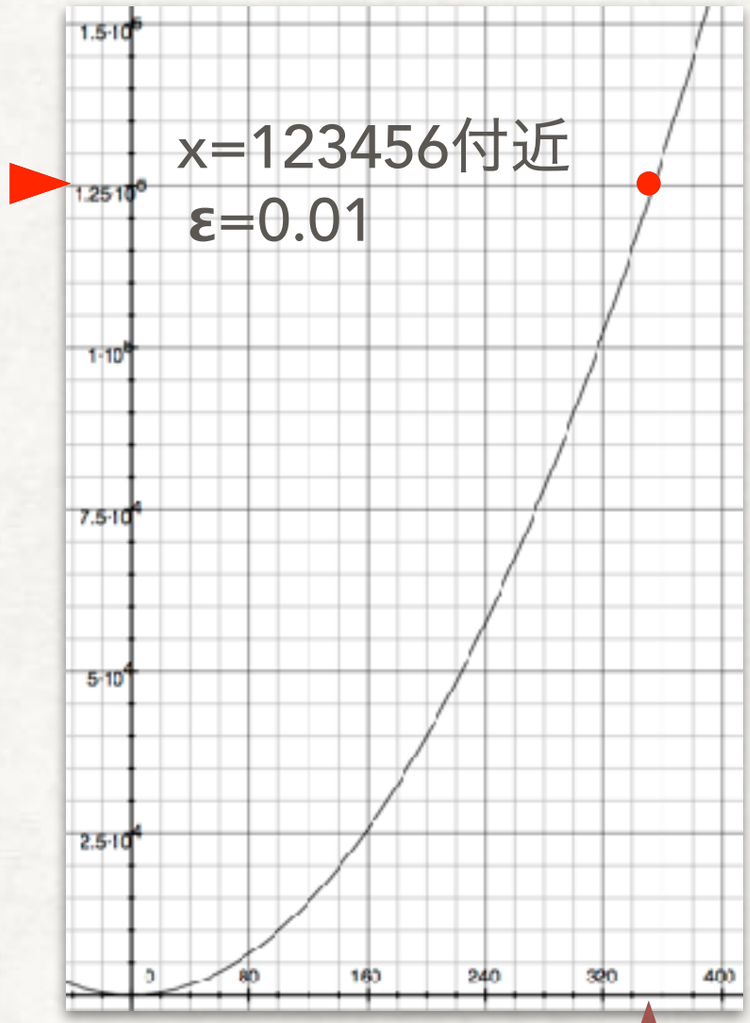
総当たりのつもりなのに見つからない近似解

Python勉強会@HACHINOHE

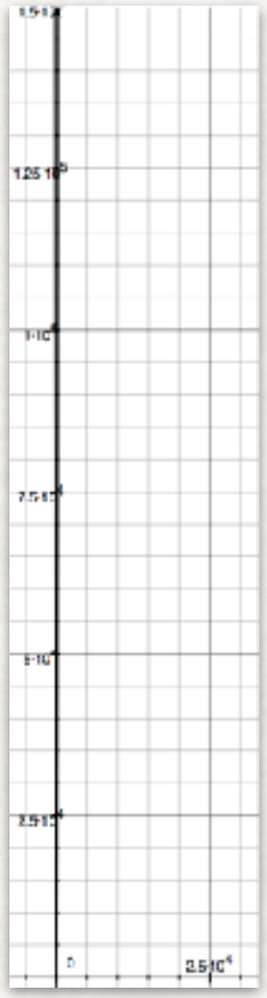
- x=123456のとき、 ε = 0.01、
  step = 0.0001なのに見つからない
- ans=350付近で、
  ansが0.0001変化すると…
  xは0.07くらい変化する

```
(350+0.0001)**2
=350**2+2*350*0.0001+0.0001**2
```

- これはεの数倍でかい
- 横軸の値がでかくなると、
  グラフの傾き(x'=2*ans)が
  大きいのが見つからない原因
- とりあえずstep=ε^3にすると
  近似解が見つかるが、
  もっと値がでかくなると結局…



縦軸、横軸同じスケールで描くと
左右がつぶれる
くらい急なカーブ

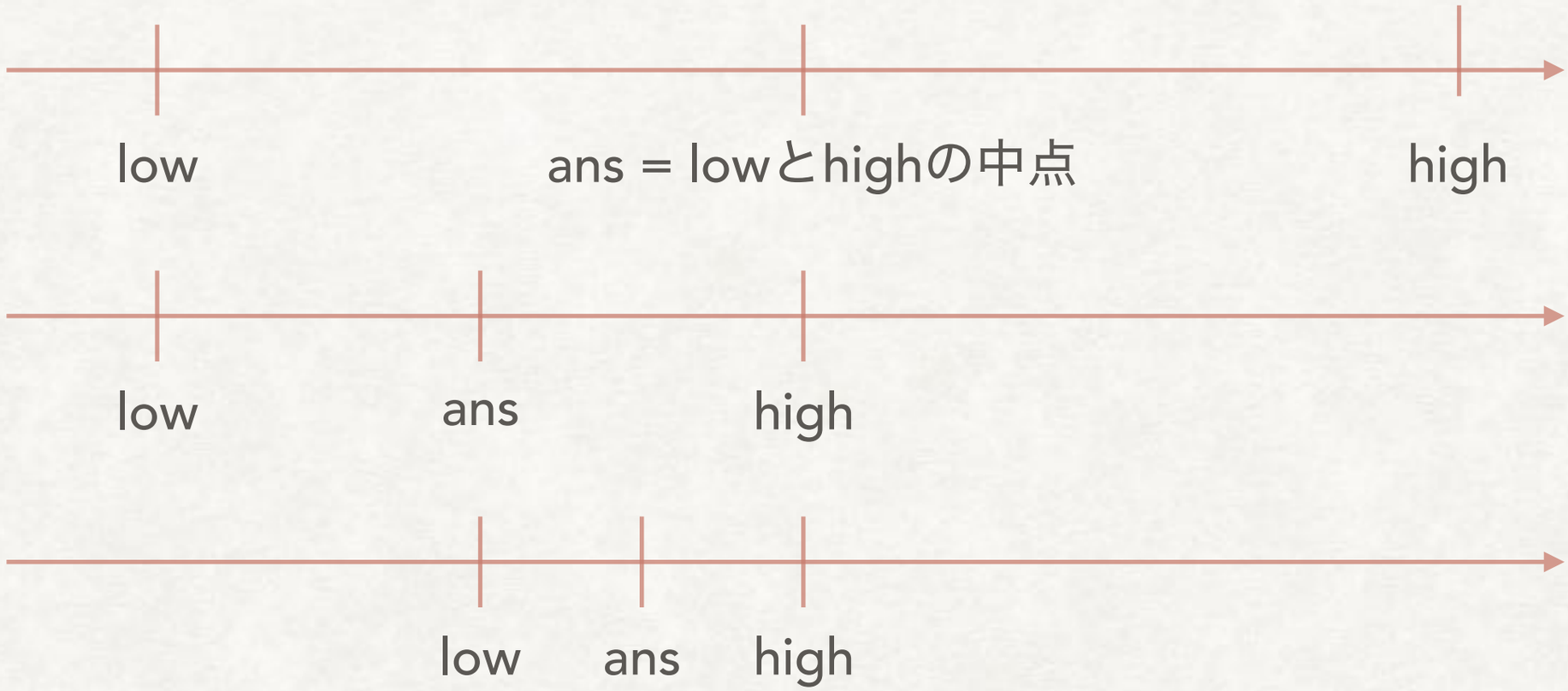


ans=350付近
step=0.0001

二分法

Python勉強会@HACHINOHE

- 探索範囲を半分ずつにしていく



- 探索範囲に解があれば見つけられる
- 最大値max(集合)、最小値min(集合)

コード3.4 平方根の近似解のための二分法

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
x = 25
epsilon = 0.01
numGuesses = 0
low = 0.0
high = max(1.0, x) # xが1以下なら平方根はxより大きくなるから
ans = (high + low) / 2.0
while abs(ans**2 - x) >= epsilon:
    print 'low =', low, 'high =', high, 'ans =', ans
    numGuesses += 1
    if ans**2 < x:
        low = ans
    else:
        high = ans
    ans = (high + low) / 2.0

print '試行回数 =' + str(numGuesses)
print str(x) + 'の平方根の近似解は' + str(ans)
```

指練習その1

Python勉強会@HACHINOHE

- x=-257の場合: 平方根は虚数
  - 範囲が0〜1 → 0〜0.5 → 0〜0.25 → 0〜0.125 → …
  - 解が虚数なので、探索範囲にない。abs(ans**2 - x) >= epsilonは絶対成り立たず、止まらない
- 実行例

```
low = 0.0 high = 1.0 ans = 0.5
low = 0.0 high = 0.5 ans = 0.25
low = 0.0 high = 0.25 ans = 0.125
...
low = 0.0 high = 0.0 ans = 0.0
low = 0.0 high = 0.0 ans = 0.0
low = 0.0 high = 0.0 ans = 0.0
...
```

指練習その2

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
x = 25
epsilon = 0.01
numGuesses = 0
low = min(-1.0, x)
high = max(1.0, x)
ans = (high + low) / 2.0
while abs(ans**3 - x) >= epsilon:
    print 'low =', low, 'high =', high, 'ans =', ans
    numGuesses += 1
    if ans**3 < x:
        low = ans
    else:
        high = ans
    ans = (high + low) / 2.0

print 'numGuesses =', numGuesses
print str(x) + 'の立方根の近似解は' + str(ans)
```

10進数と2進数と指練習

Python勉強会@HACHINOHE

- 10進数

100の位　10の位　1の位

302

$$302 = 3\times10^2 + 0\times10^1 + 2\times10^0 = 3\times100 + 0\times10 + 2\times1$$

- 2進数

16の位　8の位　4の位　2の位　1の位

10011

$$10011 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 1\times16 + 0\times8 + 0\times4 + 1\times2 + 1\times1 = 19$$

浮動小数点数の内部表現

Python勉強会@HACHINOHE

- コンピュータ内部で、浮動小数点数は仮数(かすう)部+指数部で表現
- 10進数

$$1.949 = \overset{\text{仮数}}{1949} \times 10^{\overset{\text{指数}}{-3}} \rightarrow (1949, -3)$$

- 2進数
  - 有効数字1桁 $0.1 \approx \frac{1}{2^4} = \frac{1}{16} \rightarrow (1, -100)$
  - 有効数字4桁 $0.1 \approx \frac{25}{2^8} = \frac{25}{256} \rightarrow (1101, -1000)$

2進数で0.1

Python勉強会@HACHINOHE

- 有効数字?桁　$0.1 \approx \frac{0}{2^1} = \frac{0}{2} \rightarrow (0, -1)$
- 有効数字?桁　$0.1 \approx \frac{0}{2^2} = \frac{0}{4} \rightarrow (0, -10)$
- 有効数字?桁　$0.1 \approx \frac{0}{2^3} = \frac{0}{8} \rightarrow (0, -11)$
- 有効数字1桁　$0.1 \approx \frac{1}{2^4} = \frac{1}{16} \rightarrow (1, -100)$
- 有効数字2桁　$0.1 \approx \frac{3}{2^5} = \frac{3}{32} \rightarrow (11, -101)$
- 有効数字3桁　$0.1 \approx \frac{6}{2^6} = \frac{6}{64} \rightarrow (110, -110)$
- 有効数字4桁　$0.1 \approx \frac{12}{2^7} = \frac{12}{128} \rightarrow (1100, -111)$
- 有効数字5桁　$0.1 \approx \frac{25}{2^8} = \frac{25}{256} \rightarrow (11001, -1000)$
- 有効数字6桁　$0.1 \approx \frac{51}{2^9} = \frac{51}{512} \rightarrow (110011, -1001)$
- 有効数字7桁　$0.1 \approx \frac{102}{2^{10}} = \frac{102}{1024} \rightarrow (1100110, -1010)$

小数を丸める

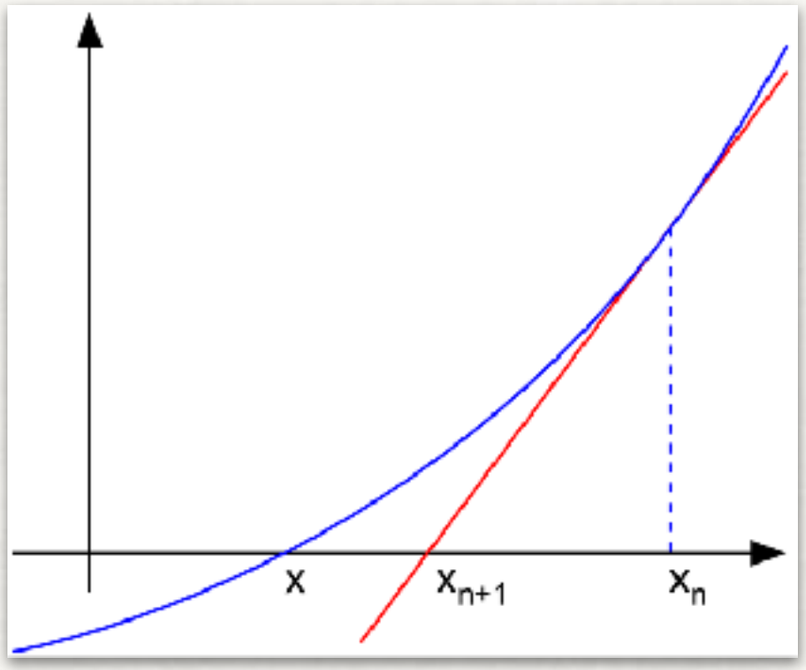
Python勉強会@HACHINOHE

- round(小数, 小数点以下の桁数)

Newton-Raphson法

Python勉強会@HACHINOHE

- 関数f(x)がx軸と交わる箇所を見つける
- 接線をヒントにして、よりよい近似値を効率よく推測する
- 1次のテイラー展開に相当
- x_nよりもよい近似値x_{n+1}は、
  点(x_n, f(x_n))での接線がx軸と交わる点



https://ja.wikipedia.org/wiki/ニュートン法

接線の方程式

$$y - f(x_n) = f'(x_n)(x - x_n)$$

x軸との交点

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

コード3.5 Newton-Raphson

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
# 平方根を求めるためのニュートン=ラフソン法
# x**2 - kがepsilon以内になるようなxを求める

k = 24.0 # 平方根を求めたい数

epsilon = 0.01 # 精度
guess = k / 2.0 # 試行解
while abs(guess * guess - k) >= epsilon:
    guess = guess - (((guess**2) - k) / (2*guess))

print str(k) + 'の平方根はおよそ' + str(x) +'です'
```

指練習: 3.5節の1

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
# 平方根を求めるためのニュートン=ラフソン法
# x**2 - kがepsilon以内になるようなxを求める

k = 24.0 # 平方根を求めたい数

epsilon = 0.01 # 精度
numGuess = 0 # 試行回数
guess = k / 2.0 # 試行解
while abs(guess * guess - k) >= epsilon:
    numGuess += 1
    guess = guess - (((guess**2) - k) / (2*guess))

print '試行回数:' + str(numGuess)
print str(k) + 'の平方根はおよそ' + str(guess) + 'です'
```

指練習: 3.5節の2

Python勉強会@HACHINOHE

```
# -*- coding: utf-8 -*-
# 平方根を求めるための二分法とニュートン=ラフソン法の比較
# x**2 - kがepsilon以内になるようなxを求める

k = 24.0 # 平方根を求めたい数
epsilon = 0.01 # 精度

numGuesses = 0
low = 0.0
high = max(1.0, k)
ans = (high + low) / 2.0
while abs(ans**2 - k) >= epsilon:
    numGuesses += 1
    if ans**2 < k:
        low = ans
    else:
        high = ans
    ans = (high + low) / 2.0
print '二分法'
print '試行回数:' + str(numGuesses)
print str(k) + 'の平方根はおよそ' + str(ans)

numGuess = 0
guess = k / 2.0
while abs(guess * guess - k) >= epsilon:
    numGuess += 1
    guess = guess - (((guess**2) - k) / (2*guess))
print 'ニュートン=ラフソン法'
print '試行回数:' + str(numGuess)
print str(k) + 'の平方根はおよそ' + str(guess) + 'です'
```

まとめ

Python勉強会@HACHINOHE

- 「簡単」な算術プログラムでした
- 試行錯誤法を、多項式の根をテーマに学んだ
- 「解を探す範囲」と「解に近づく方法」がポイント
- 解に近づく方法
  - 総当たり
    - ある刻みで片っ端から探索する
  - 二分法
    - 整列された対象(探索範囲内で単調増加か減少)に対し、探索範囲を半分ずつに減らして効率的に見つける
  - Newton-Raphson
    - よりよい解を接線を参考に探す。微分できる必要がある