数理生物学演習

第３回　個体群動態の数理モデル（１）：
離散ロジスティック成長モデル

野下 浩司（Noshita, Koji）

noshita@morphometrics.jp

https://koji.noshita.net

理学研究院 数理生物学研究室

第２回：個体群動態の数理モデル（１）：離散指数増殖，離散ロジスティック成長モデル

今回の目標

- 差分方程式を解く
- 時間発展をプロットする

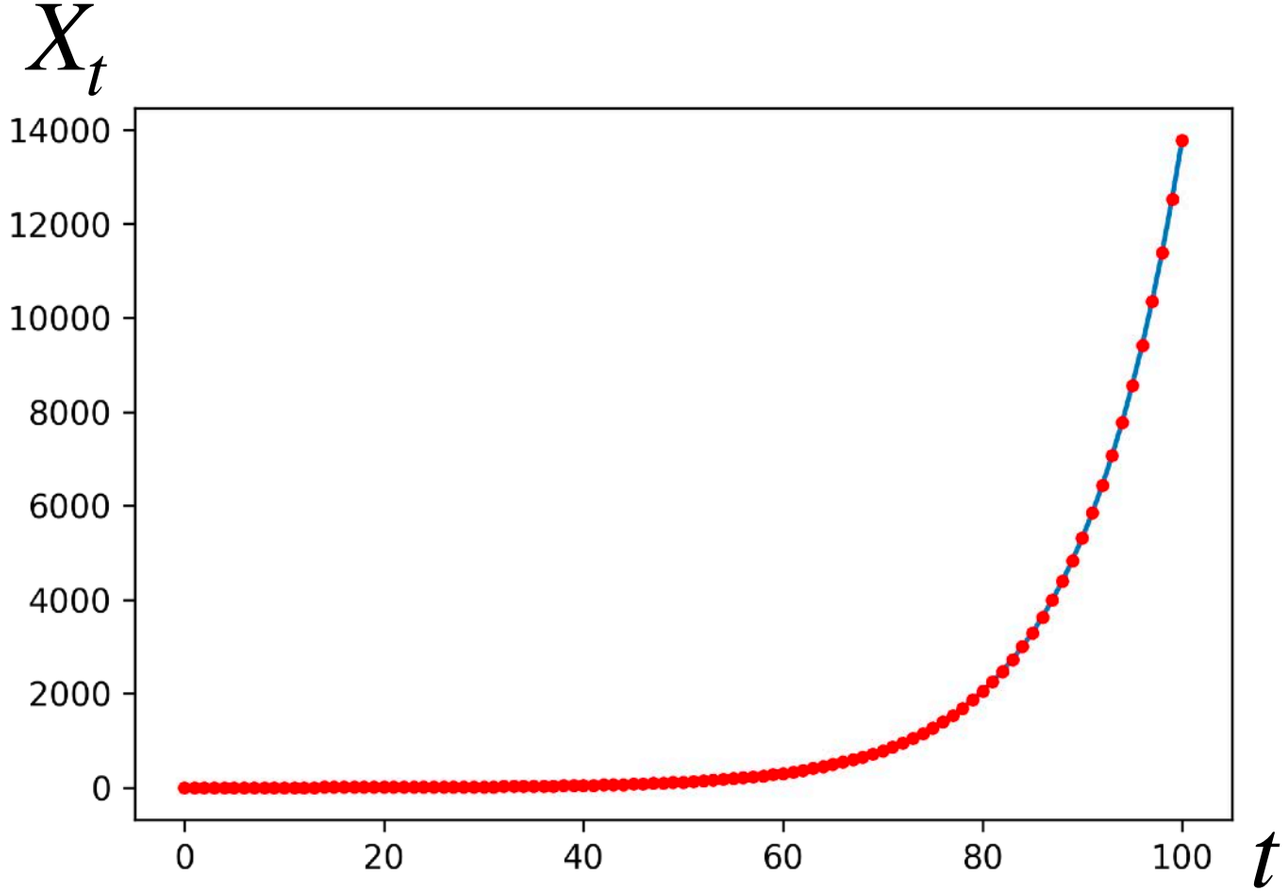
個体群動態（離散時間）

指数増殖・ロジスティック成長モデル

離散時間指数増殖モデル

$$X_{t+1} = X_t + aX_t$$

a：マルサス係数．１世代あたりの増殖率．$a \geq 0$.

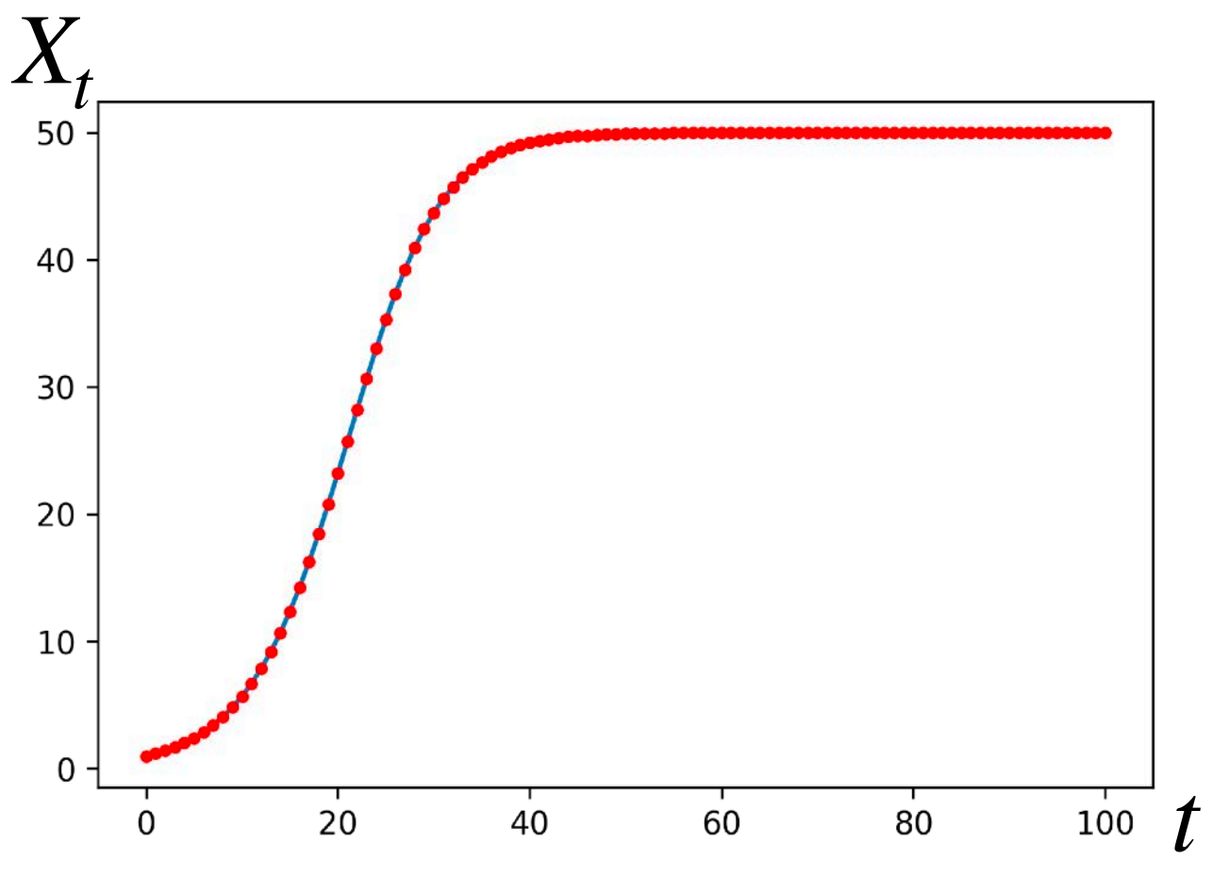


離散時間ロジスティック成長モデル

$$X_{t+1} = X_t + r\left(1 - \frac{X_t}{K}\right)X_t$$

r：内的自然増加率．個体数が十分小さい場合（$X \approx 0$）の
１世代あたりの増殖率．$r \geq 0$.

K：環境収容力．ある環境で維持されうる個体数，$K > 0$.



平衡点と局所安定性

時間的に変化しない（釣り合いが取れている）状態を見つけ，その安定性を調べる

平衡点：$X_{t+1} = X_t = \bar{X}$となるような$\bar{X}$

例．離散ロジスティック成長モデル

$$X_{t+1} = X_t + r\left(1 - \frac{X_t}{K}\right)X_t$$

モデルの式に代入すれば，

$$r\left(1 - \frac{\bar{X}}{K}\right)\bar{X} = 0.$$

これを満たすが $\bar{X}$ が平衡点になる．

よって，平衡点は $\bar{X} = 0, K$ の２つ存在する．

局所安定性：平衡点からの微小なずれが生じた際にどうなるか

$X_{t+1} = f(X_t)$とし，微小なずれ n_tを考えテイラー展開すると，

$$\begin{aligned}\bar{X} + n_{t+1} &= f\left(\bar{X} + n_t\right) \\ &= f\left(\bar{X}\right) + \frac{df}{dX}n_t + \frac{1}{2}\frac{d^2f}{dX^2}n_t^2 + \cdots\end{aligned}$$

$\bar{X}$は平衡点なので$\bar{X} = f\left(\bar{X}\right)$．また，$n_t$ は十分小さいので2次以降の項を無視すると，

$$n_{t+1} \approx \frac{df}{dX}n_t$$

よって，$\left|\frac{df}{dX}\right| < 1$で安定，$\left|\frac{df}{dX}\right| > 1$で不安定．

離散ロジスティック成長モデルの場合に２つの平衡点について計算してみよう！

関数，モジュール・パッケージ

関数（1）

復習 第2回

- Pythonおける関数とは，ある一連の処理を行うコードをまとめたもの
- これまで使ってきた，print()やtype()は関数
- 使う前に定義し，使うときに呼び出す必要がある．

関数定義

```
def 関数名():
    処理1
    処理2
    …
    処理n
```

関数定義

関数呼び出し

```
# 02-01. シンプルな関数
def simplefunc():
    print("関数を呼び出しました．")

simplefunc()
```

「関数を呼び出しました．」と表示させる関数

```
# 出力
関数を呼び出しました．
```

あまり気にしなくても良い補足
printやtype関数は予め用意されている「組み込み関数」．はじめから使える．

- 組み込み関数¦公式ドキュメント
  https://docs.python.org/ja/3/library/functions.html

関数定義

関数呼び出し

```
# 02-02. シンプルな関数 その2
def simplefunc2():
    print("1．関数を")
    print("2．呼び出し")
    print("3．ました．")

simplefunc2()
```

```
# 出力
1．関数を
2．呼び出し
3．ました．
```

何度も利用する処理を関数にまとめることで再利用性を高める

関数（2）：引数と戻り値

復習 第2回

- **引数**により，入力に応じた処理をおこなうことができる
  例．print()が表示する文字列が変わる
- 処理した結果を，**戻り値**として返すことができる
  例．a = abs(-2) # aに2が代入される

関数定義

```
def 関数名(パラメータ):
    処理1
    処理2
    …
    処理n
    return 戻り値
```

- パラメータ（仮引数）：関数内でのみ利用される変数．関数に渡す値やリストなどを入力としてもつ．
- return文：関数を終了して，戻り値を返す

入力した値とその絶対値を表示させる関数

```
# 02-03. 引数をもつ関数
def my_abs_print(x):
    y = abs(x)
    print("入力", x)
    print("絶対値", y)

my_abs_print(-9)
```

```
# 出力
入力 -9
絶対値 9
```

```
# 02-04. 引数と戻り値をもつ関数
def add(a, b):
    c = a + b
    return c

x = add(2,4)
print(x)
```

2変数の足し算

```
# 出力
6
```

モジュール・パッケージ（1）

Pythonコードをまとめたファイルやその集合

- モジュール：コードをまとめたファイル
- パッケージ：モジュールを階層的にまとめたもの

この演習ではこれらの区別はあまりしない．モジュール，パッケージ，ライブラリなど異なる名前で呼称するが，「必要なときに呼び出せる便利な機能をまとめたもの」ぐらいのニュアンスで理解しておけばOK

使い方

- import モジュール（もしくはパッケージ）
  モジュール（もしくはパッケージ）を読み込む

```
# 01-01. mathモジュールの読み込み
import math

a = math.log(2)
print(a)
```

```
# 01-02. osパッケージの読み込み
import os

filepath = os.path.join("parent", "child", "file.txt")
print(filepath)
```

便利な機能をまとめたものを再利用することで1から作る必要がなくなる！

mathモジュール

基本的な数学関係の関数

https://docs.python.org/ja/3/library/math.html

よく使いそうな関数の例
- log：自然対数
- sqrt：平方根
- sin,cos,tan, …：三角関数関係

数学関係の定数
- pi:円周率
- e:自然対数の底

print関数の補足
- print(obj1, obj2, …)
  obj1, obj2, …を（デフォルトだと空白で）区切って表示

```
# 01-03. mathモジュール
import math

print("円周率：", math.pi)
print("自然対数の底", math.e)

print("log(2)：",math.log(2))
print("√3：", math.sqrt(3))

print(“sin(π/2)：”, math.sin(math.pi/2))
print("cos(π)：", math.cos(math.pi))
print("tan(π/4)", math.tan(math.pi/4))
```

その他の標準ライブラリ（デフォルトで使えるモジュールやパッケージ）もあるので興味のある人は使ってみよう.

- **Python 標準ライブラリ | 公式ドキュメント https://docs.python.org/ja/3/library/index.html**

さらに，Colabには標準ライブラリ以外にもデータサイエンス向けのパッケージが多数インストール済み（特に追加インストールの必要なく呼び出せる）.

モジュール・パッケージ（2）

その他の読み込み方

- from パッケージ import モジュール
  パッケージ内のモジュールを読み込む
- import モジュール（もしくはパッケージ） as 省略名
  パッケージを省略名として読み込む
- from パッケージ import モジュール as 省略名
  パッケージ内のモジュールを省略名として読み込む

```
# 01-04. matplotlibパッケージのpyplotモジュールをpltとして読み込む
import matplotlib.pyplot as plt
```

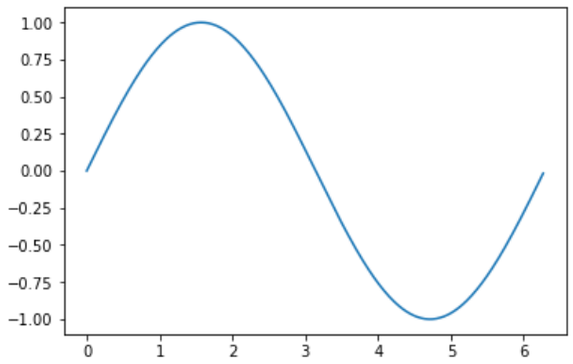
有名ライブラリの省略名はだいたい慣例があるので，それに従う（例. matplotlib.pyplot→plt）．また，自作のモジュールやパッケージを作る場合には，そうした有名ライブラリの名前や省略名との重複を避けるのが無難.

Matplotlib

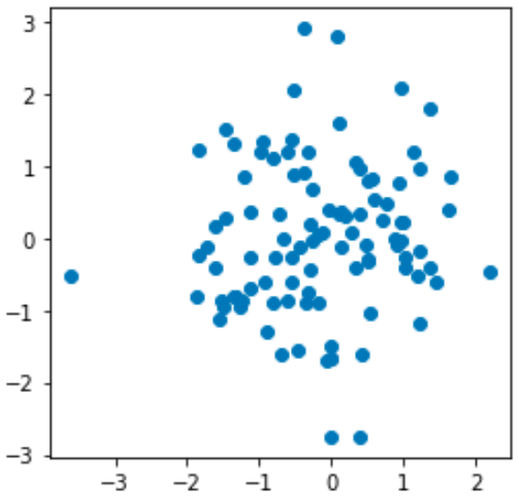
データ可視化・作図ライブラリ

https://matplotlib.org/

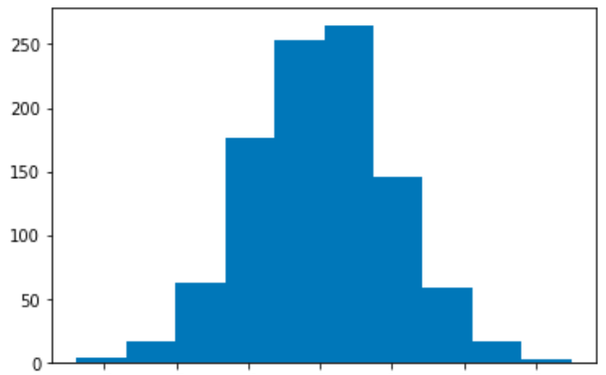
基本的なプロット



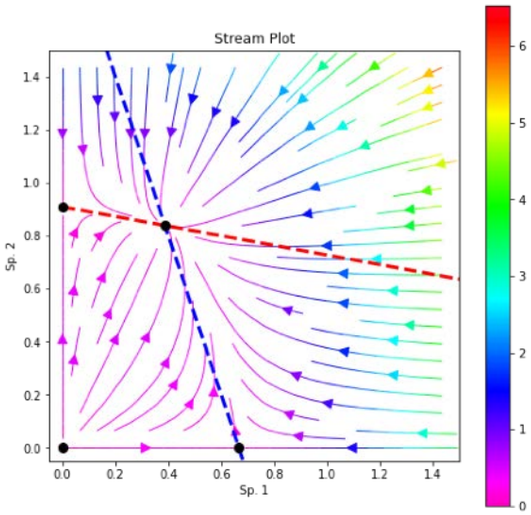
散布図



ヒストグラム



ベクトル場



ここでは例をいくつか示すだけで，個別の関数の詳細な使い方は説明しない．
例に挙げた例以外にも様々なプロットが可能．公式のサンプル集を眺めてみると，使いたいプロット方法が見つかるかも．

- **Gallery | 公式ドキュメント**
  https://matplotlib.org/gallery/index.html

プロット

結果を図として可視化する

```
# 01-05. sin関数のプロット

import matplotlib.pyplot as plt
import math

xEnd = 2*math.pi
step = math.pi/36

x_list = []
y_list = []
for i in range(0, int(xEnd/step)+1):
    x = step*i
    y = math.sin(x)
    x_list.append(x)
    y_list.append(y)

plt.plot(x_list, y_list)
```

同じ高さのインデントにより
forループのブロックを表現

パッケージの読み込み

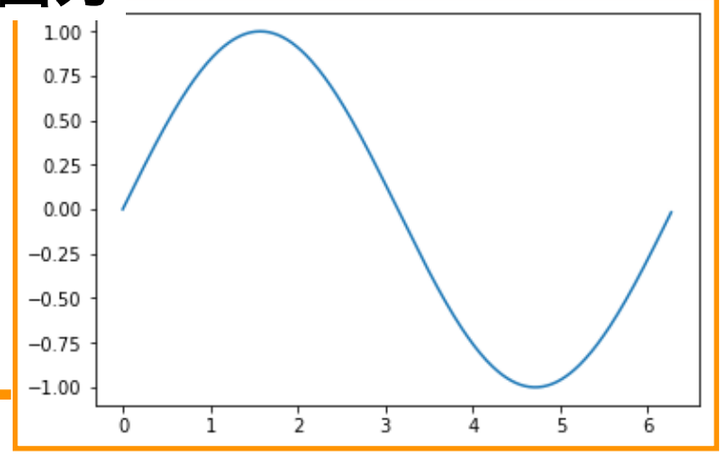
どこまで計算するか？（xの最大値）

x軸方向の刻み幅

刻み幅を変えてプロットしてみよう！

x座標，y座標の値を格納するリスト

出力



matplotlib.pyplot

- plot(横軸値リスト，縦軸値リスト)
  (横軸値，縦軸値)で与えられる座標値をプロットする

- リスト.append(要素)
  リストの末尾に要素を付け加える
- int(数値)
  数値を切り捨てて整数にする

離散指数増殖モデルの数値計算とプロット

離散指数増殖モデル（1）

```
# 02-01. 離散指数増殖モデル（1）
import matplotlib.pyplot as plt

a = 0.1
x = 1
t = 0

t_list = [t]
x_list = [x]
for i in range(100):
    t = t + 1
    x = x + a*x

    t_list.append(t)
    x_list.append(x)

plt.plot(t_list, x_list)
```

パラメータ・初期値の設定

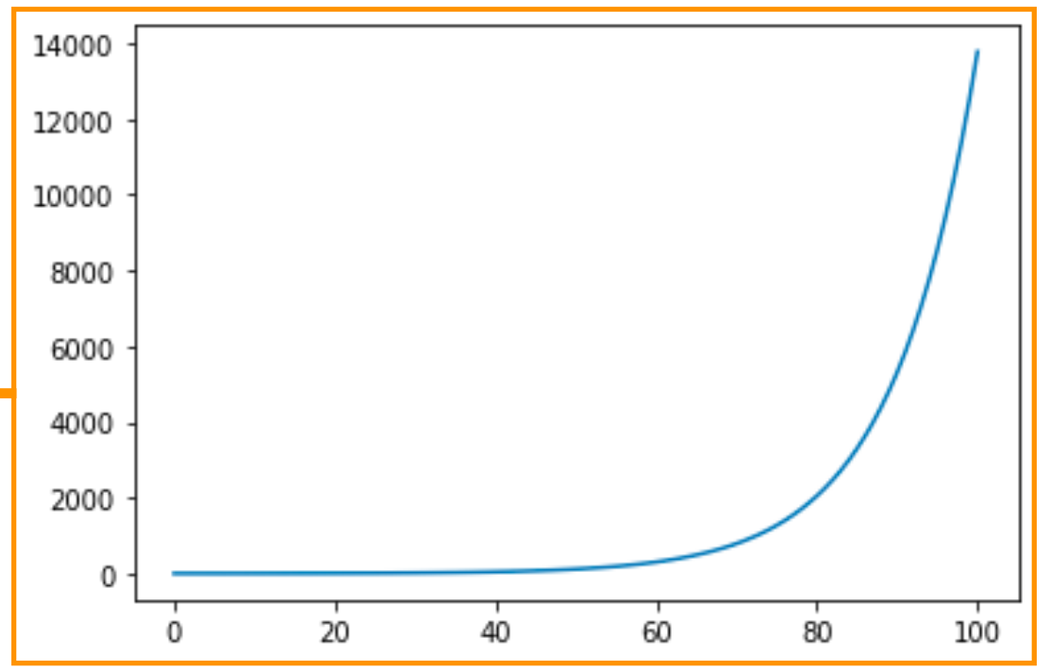
時刻（t）と個体数（x）を記録するリスト

100世代のループ

$X_{t+1} = X_t + aX_t$

更新された値をリストに格納

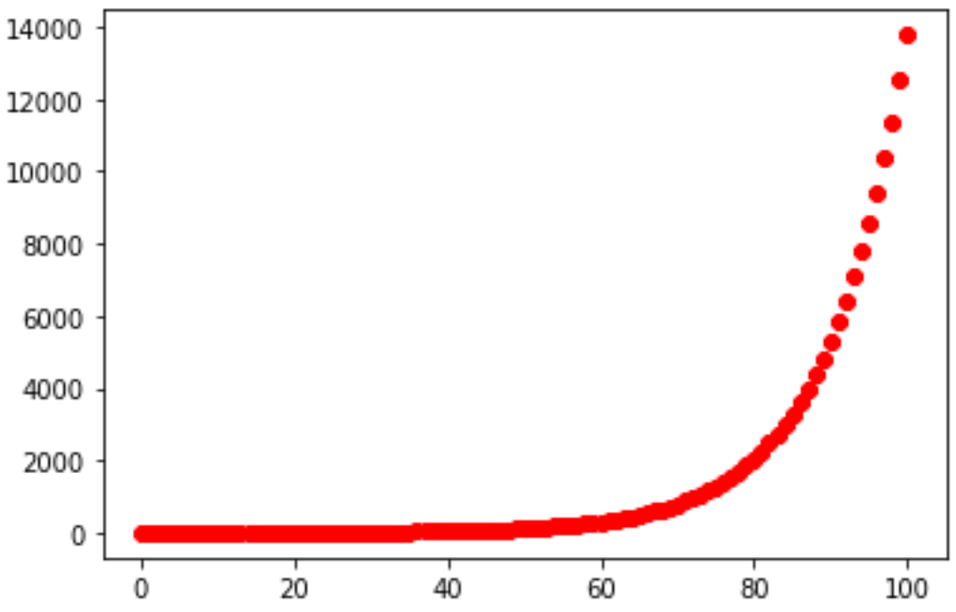
同じ高さのインデントによりforループのブロックを表現



プロットを鍛える（1）

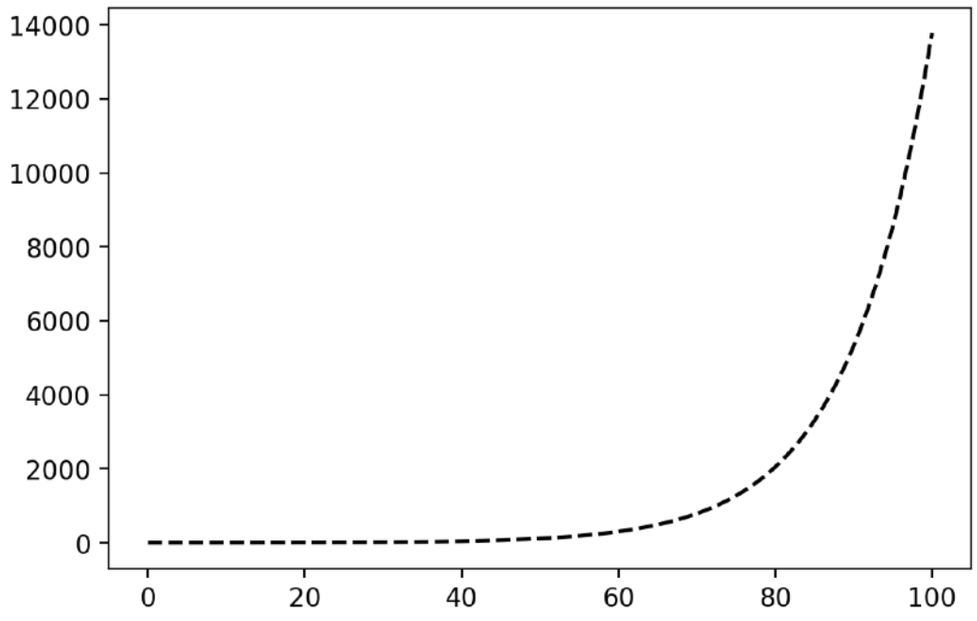
matplotlib.pyplot
- plot(横軸値リスト，縦軸値リスト，フォーマット)
  (横軸値，縦軸値)で与えられる座標値を指定されたフォーマットでプロットする

```
# 02-02. フォーマットの変更1
plt.plot(t_list, x_list, "ro")
```



フォーマットの例
r: 赤，o: circleマーカー

```
# 02-03. フォーマットの変更2
plt.plot(t_list, x_list, "k--")
```



フォーマットの例
k: 黒，--: ダッシュライン

指定できるフォーマットの詳細は公式ドキュメントを参照.

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

プロットを鍛える（2）

matplotlib.pyplot
- plot(横軸値リスト1，縦軸値リスト1，フォーマット1，横軸値リスト2，縦軸値リスト2，フォーマット2，…)(横軸値，縦軸値)で与えられる座標値を指定されたフォーマットでプロットする（フォーマットを指定しなくても良い）．

```
# 02-04. 複数のデータのプロット1
plt.plot(t_list, x_list, "-", t_list, x_list, "r.")
```

```
# 02-05. 複数のデータのプロット2
x_list_list = []

for a in [0.1, 0.11, 0.12]:
  x = 1
  t = 0
  t_list = [t]
  x_list = [x]

  for i in range(100):
    t = t + 1
    x = x + a*x

    t_list.append(t)
    x_list.append(x)

  x_list_list.append(x_list)

plt.plot(t_list, x_list_list[0],
         t_list, x_list_list[1],
         t_list, x_list_list[2])
```

個体数（x）を記録するリスト
（x_list）を記録するリスト

3通りのaの値について
ループを回す

それぞれのaの値について
100世代分のループを回す

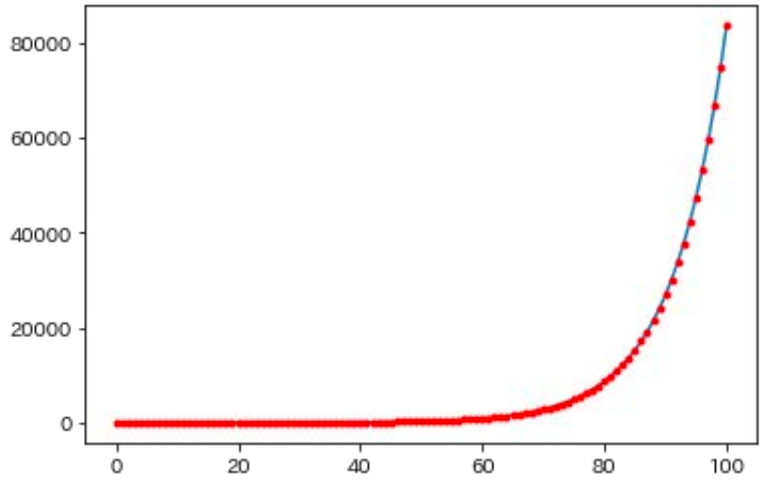
a=0.1の結果

a=0.11の結果

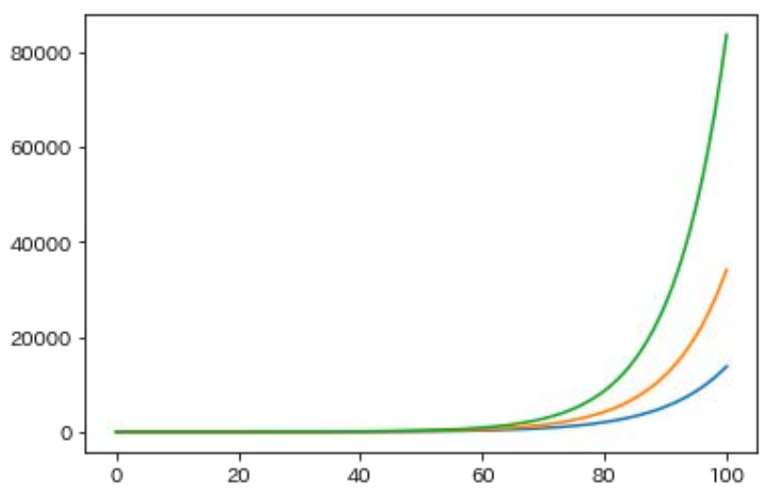
a=0.12の結果

forのネスト（2重ループ）

02-04. 同じ結果を異なるフォーマットで重ねてプロット



02-05. 異なるaに対する結果をまとめてプロット

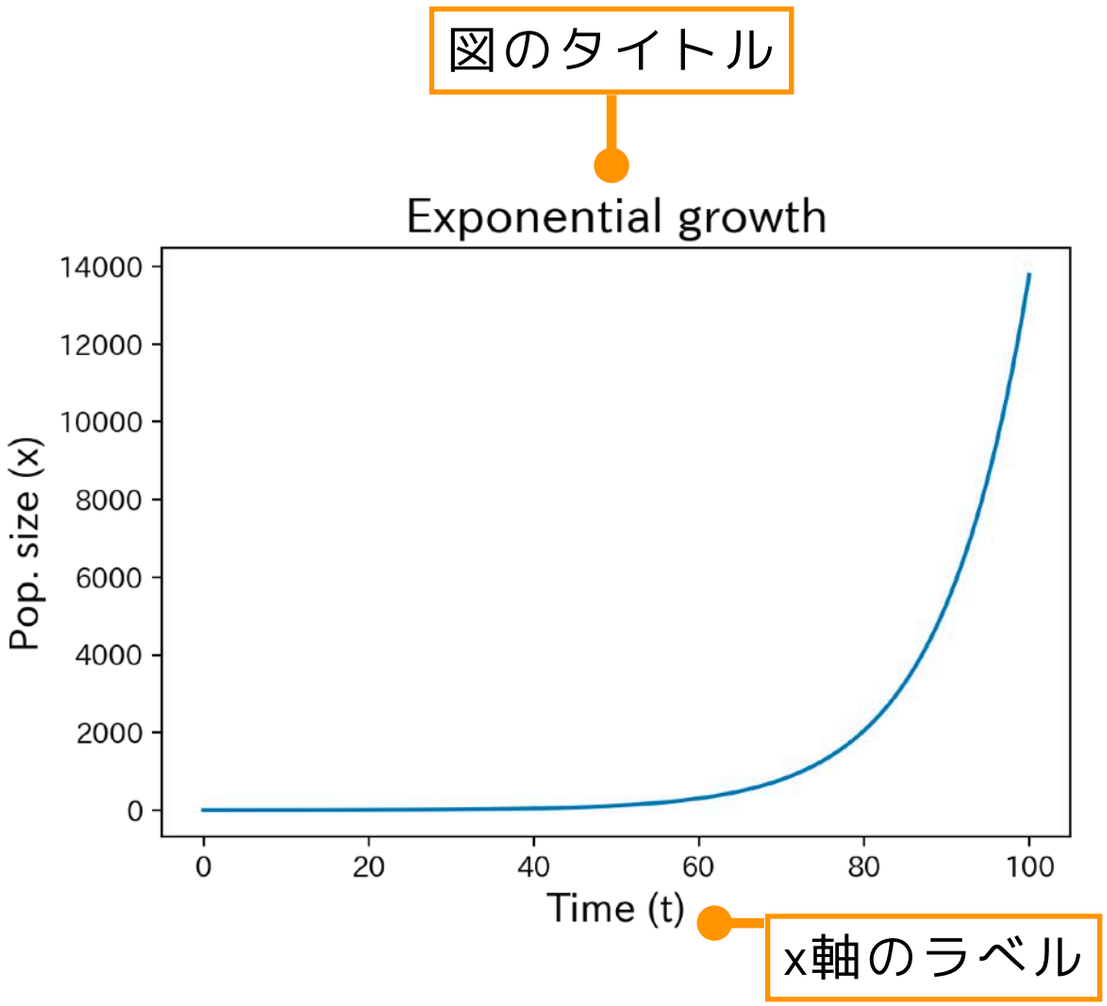


プロットを鍛える（3）

matplotlib.pyplot
- title(タイトル)：プロットにタイトルをつける．
- xlabel(x軸ラベル),ylabel(y軸ラベル)：各軸にラベルをつける．

```
# 02-06. タイトル・軸ラベル1

plt.plot(t_list, x_list)
plt.title("Exponential growth")
plt.xlabel("Time (t)")
plt.ylabel("Pop. size (x)")
```



注意：デフォルトだと，日本語フォントをタイトルやラベルに使うと豆腐化する（日本語フォントが□になってしまう）．基本はアルファベットがおすすめ．どうしても日本語化したい人は後述の日本語化を試してみよう．

```
# 02-07. タイトル・軸ラベル2

plt.plot(t_list, x_list)
plt.title("Exponential growth", fontsize="xx-large")
plt.xlabel("Time (t)",fontsize="x-large")
plt.ylabel("Pop. size (x)",fontsize="x-large")
```

フォントサイズを指定できる．整数もしくは'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', ‘xx-large'のいずれか．

プロットを鍛える（3）：おまけ

matplotlibで日本語を使う

```
# matplotlibで日本語フォントを使う準備１
# japanize-matplotlibのインストール

!pip install japanize-matplotlib
```

matplotlibで日本語表示できるようにするパッケージ
japanize-matplotlib
- https://github.com/uehara1414/japanize-matplotlib
- https://pypi.org/project/japanize-matplotlib/

をインストール．!はOS上でシェルコマンドを実行するためのマジックコマンド

```
# matplotlibで日本語フォントを使う準備２
# matplotlibとともにjapanize_matplotlibを読み込む

import matplotlib.pyplot as plt
import japanize_matplotlib
```

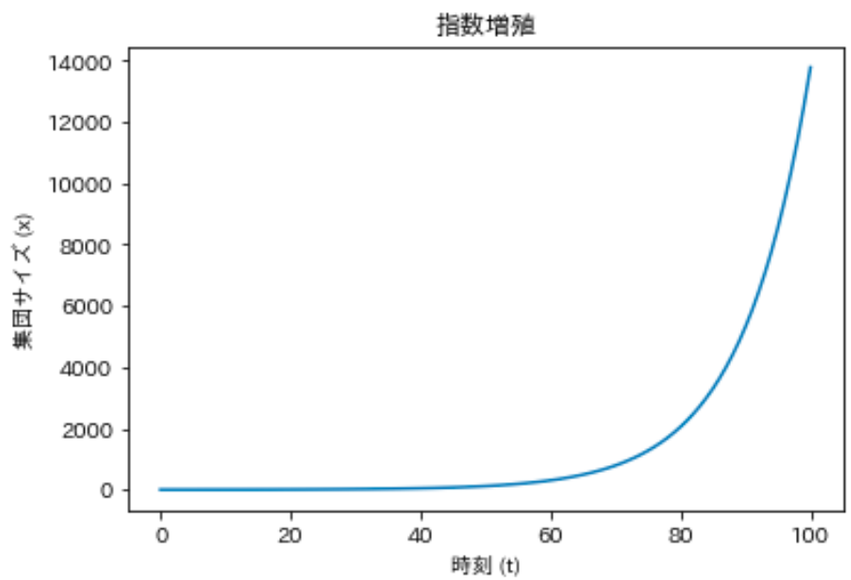
matplotlibを読み込んだ後で，japanize_matplotlibをインポートする．これによりmatplotlibのフォント設定を弄り，日本語フォントを利用可能な状態にしている．
参考：pip install して import するだけで matplotlib を日本語表示対応させる ¦ Qiita https://qiita.com/uehara1414/items/6286590d2e1ffbf68f6c

```
# タイトル・軸ラベル（日本語）

plt.plot(t_list, x_list)
plt.title("指数増殖")

plt.xlabel("時刻 (t)")

plt.ylabel("集団サイズ (x)")
```



japanize-matplotlibのインストールはインスタンスが再起動された場合には，再度行う必要あり！

プロットを鍛える（4）

matplotlib.pyplot
- figure(dpi=解像度)：図の解像度を指定する．デフォルトは100（dpi）．
- figure(figsize=[幅, 高さ])：図のサイズ（幅と高さ）をインチで指定する．デフォルトは[6.4, 4.8]

```
# 02-08. 解像度の変更

plt.figure(dpi = 200)
plt.plot(t_list, x_list)
```

```
# 02-09. プロットサイズの変更

plt.figure(figsize = [5,7])
plt.plot(t_list, x_list)
```

他にも様々な調整ができるので，詳しく知りたい人は
- 公式ドキュメント https://matplotlib.org/
- DataCampチュートリアル Matplotlib Tutorial: Python Plotting https://www.datacamp.com/community/tutorials/matplotlib-tutorial-python

などを参照．また，matplotlib以外やmatplotlibを拡張するプロット用のライブラリがあるので，興味のある人は探してみよう．

離散指数増殖モデル（2）

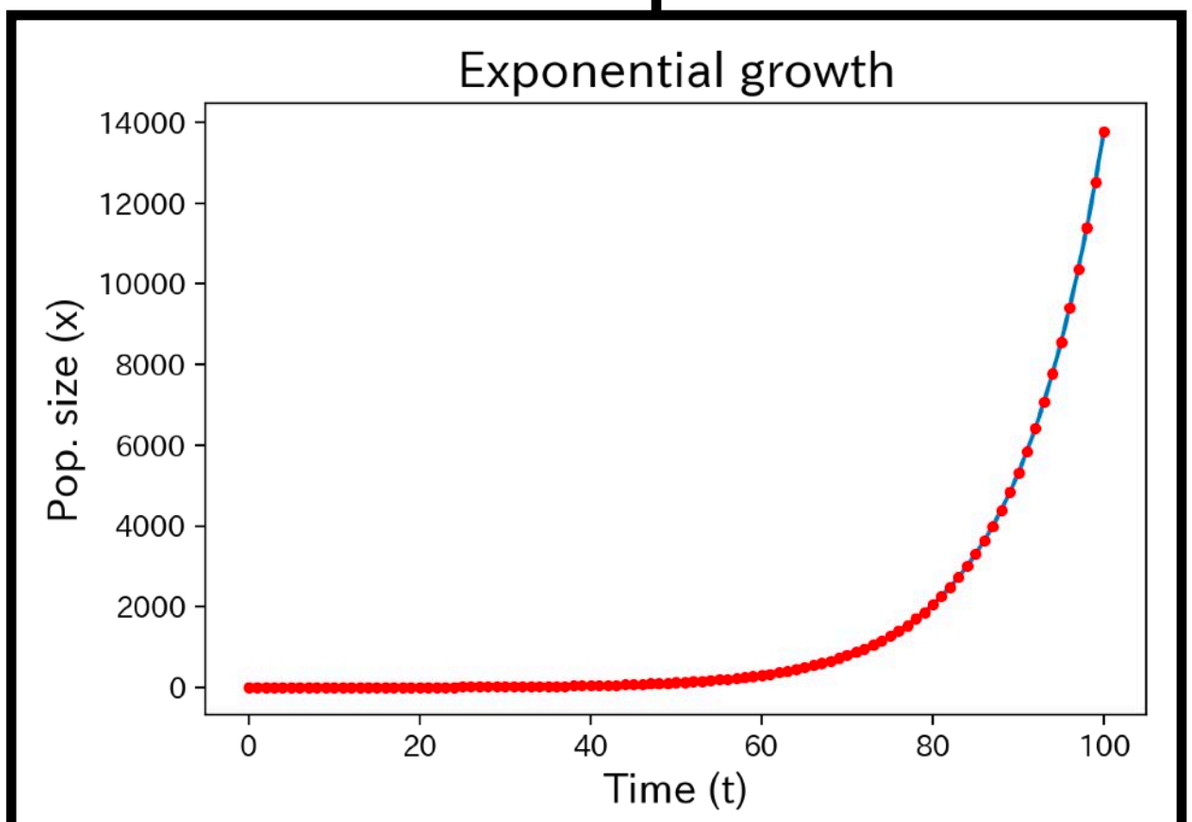
```python
# 02-10. 離散指数増殖モデル（2）
import matplotlib.pyplot as plt

a = 0.1
x = 1
t = 0

t_list = [t]
x_list = [x]
for i in range(100):
    t = t + 1
    x = x + a*x

    t_list.append(t)
    x_list.append(x)

plt.figure(dpi = 200)
plt.plot(t_list, x_list, "-", t_list, x_list, "r.")
plt.title("Exponential growth", fontsize="xx-large")
plt.xlabel("Time (t)",fontsize="x-large")
plt.ylabel("Pop. size (x)",fontsize="x-large")
```



離散ロジスティック成長モデルの数値計算とプロット

課題 ノーマル 3

プログラムの流れ
離散ロジスティックモデルの時間発展

必要なパッケージ（matplotlib）の読み込み

↓

初期値（x0）・パラメータ（r, K）の定義

↓

時間tと個体数xのリスト（t_list, x_list）を作成

↓

forループ

100世代のループ

差分方程式

$$X_{t+1} = X_t + r\left(1 - \frac{X_t}{K}\right)X_t$$

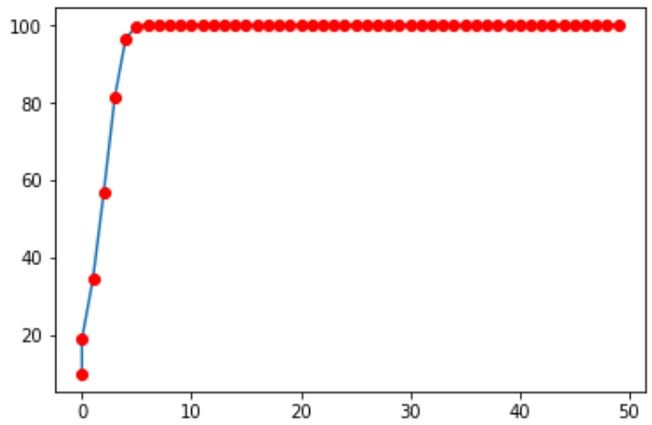
を使い，t+1ステップ目を計算する.

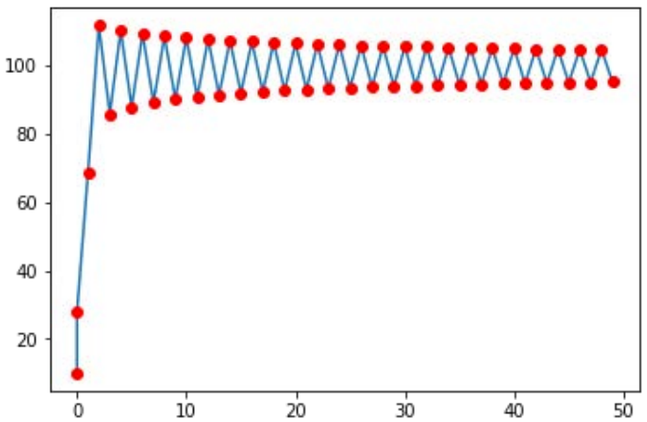
```
x = x+r*(1-x/K)*x
```

結果をリストに追加

↓

プロット





局所安定性解析の結果から推察される様々な挙動が再現できるとGood！

離散指数増殖モデルを参考にしてみよう

```python
# 02-10. 離散指数増殖モデル（2）
import matplotlib.pyplot as plt

a = 0.1
x = 1
t = 0

t_list = [t]
x_list = [x]
for i in range(100):
    t = t + 1
    x = x + a*x

    t_list.append(t)
    x_list.append(x)

plt.figure(dpi = 200)
plt.plot(t_list, x_list, "-", t_list, x_list, "r.")
plt.title("Exponential growth", fontsize="xx-large")
plt.xlabel("Time (t)",fontsize="x-large")
plt.ylabel("Pop. size (x)",fontsize="x-large")
```

必要なパッケージ（matplotlib）の読み込み

↓

初期値（x0）・パラメータ（a）の定義

↓

時間tと個体数xのリスト（t_list, x_list）を作成

↓

forループ

100世代のループ

差分方程式

$$X_{t+1} = X_t + aX_t$$

を使い，t+1ステップ目を計算する.

```
x = x + a*x
```

結果をリストに追加

↓

プロット

離散指数増殖モデルを参考にしてみよう

```python
# 02-10. 離散指数増殖モデル（2）
import matplotlib.pyplot as plt

a = 0.1
x = 1
t = 0

t_list = [t]
x_list = [x]
for i in range(100):
    t = t + 1
    x = x + a*x

    t_list.append(t)
    x_list.append(x)

plt.figure(dpi = 200)
plt.plot(t_list, x_list, "-", t_list, x_list, "r.")
plt.title("Exponential growth", fontsize="xx-large")
plt.xlabel("Time (t)",fontsize="x-large")
plt.ylabel("Pop. size (x)",fontsize="x-large")
```

必要なパッケージ（matplotlib）の読み込み

↓

初期値（x0）・パラメータ（a）の定義

↓

時間tと個体数xのリスト（t_list, x_list）を作成

↓

forループ

100世代のループ

差分方程式

$$X_{t+1} = X_t + aX_t$$

を使い，t+1ステップ目を計算する.

```
x = x + a*x
```

結果をリストに追加

↓

プロット

この辺を変更すれば良さそう！

+プロット時のタイトルも

第３回 課題 ノーマル

1. 離散ロジスティックモデルの平衡点を求めよ．
   また，その安定性を調べよ．
2. 離散ロジスティックモデルの時間発展を様々な
   rに対してプロットせよ（$0.5 \leq r \leq 3$ぐらいの範
   囲がおすすめ）．
3. 質問，意見，要望等をどうぞ．

課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること

ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．03_n.ipynb

第３回 課題 ハード

1．離散ロジスティックモデルの分岐図を描け．

課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること

ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．03_h.ipynb

ハードに関連する内容は
発展的内容の資料を参照.

次回予告

第４回：指数成長・ロジスティック成長

５月１０日

復習推奨

- 指数成長
- ロジスティック成長
- 微分方程式（変数分離型）の解法