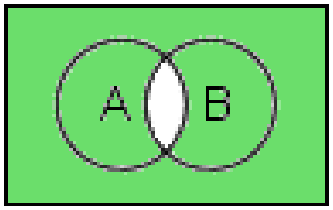
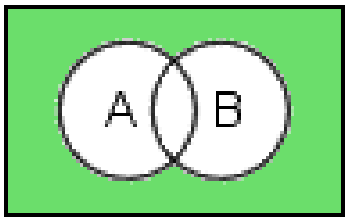
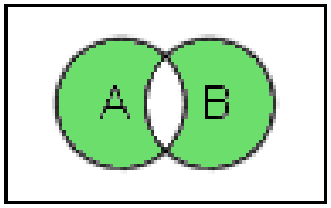
第 3 章【学習 11】 演習解答

●演習 1　P100

<table>
<tr><th></th><th>ベン図</th><th>表</th></tr>
<tr><td>（1）</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th></th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>（2）</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th></th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>（3）</td><td> 
</td><td>
<table>
<tr><th>A</th><th>B</th><th></th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

※（3）は A と B の排他的論理和といい，A XOR B と表現される。

●演習 2　P100

①AND ②NOT ③AND

●演習 3　P102

惑星探査のように長距離を航行する場合，わずかな角度の差，スピードの違いが，航行を続けるうちに大きな影響を与える。また，惑星探査には，地球重力圏からの脱出，地球以外の惑星の重力を利用してスピードをあげるスイングバイ航法，目的とする惑星に近づいた際の速度や軌道の調整など，膨大な計算が必要になるため，計算誤差が大きな問題になる。

0.28−0.27 を計算する前に，0.28 に 100 をかけて整数にし，0.27 にも 100 をかけて整数にし，28−27 を計算した後，その答えを 100 で割るなど，誤差の起こらない範囲で計算を行い，位取りを変えるなどの工夫を行う。

第 3 章【学習 12】 演習解答

●演習 1　P107

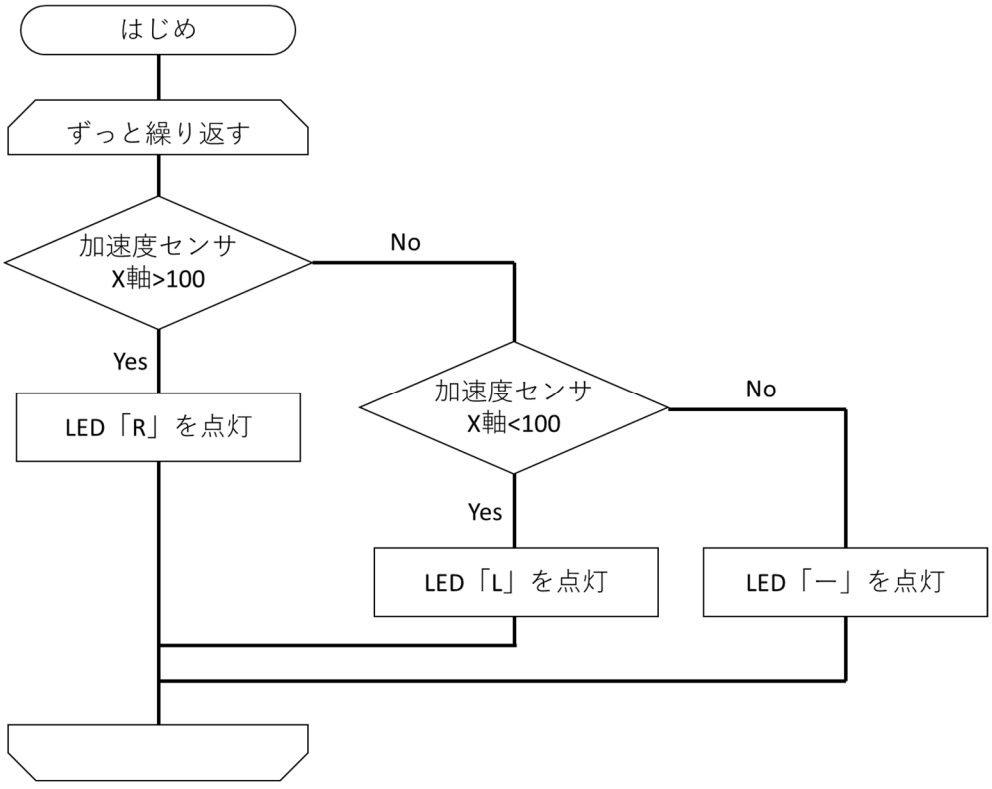
⑴洗濯機は水位センサで水位を量り，水流を発生させる羽をモータで回転させるアクチュエータが動いている。

⑵鍵が貸し出し可能な時間帯以外に借りられるため，貸し出し禁止の時間に鍵がなくなったら超音波センサで感知しアラートを鳴らす。卓球部の部員が少ないため，ピンポン玉をロボットアームで回転して弾き出し，レシーブの練習を行い練習効率を上げる。　など

●演習 2　P109

解答プログラム参照

フローチャートは以下の通り



```
from microbit import *
while True:
	reading = accelerometer.get_x()
	if reading > 100:
		display.show("R")
	elif reading < 100:
		display.show("L")
	else:
		display.show("-")
```

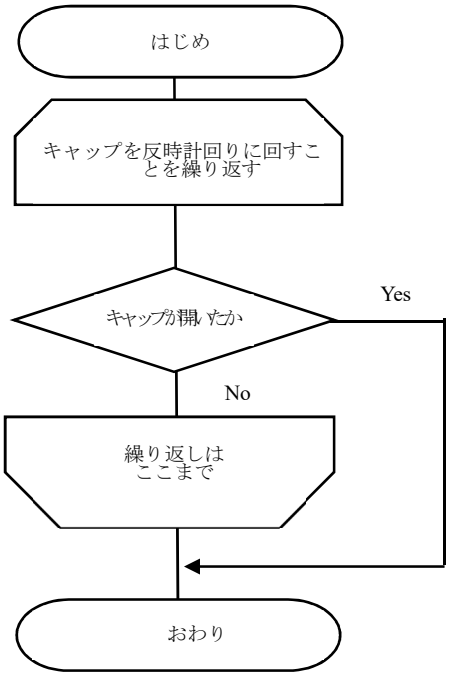
●演習 3　P110

```
from microbit import *

while True:
	if pin0.is_touched():
		display.show("A")
	else:
		display.show("B")
```

第 3 章【学習 13】 演習解答

●演習 1　P115



●演習 2　P116

```
print("おはよう")
print("こんにちは")
print("こんばんは")
print("おやすみ")
```

●演習 3　P117

(1)

```
x = 40
if x >= 60 :
        print("合格")
else :
        print("不合格")
```

(2)

```
x = 40
if x < 30 :
       print("再試験")
else :
       print("合格")
```

●演習 4　P117

(1)

```
x = 0
print("x = " , x)
for i in range(1, 6, 1):
       x = x + 10
       print("x = " , x)
       print("繰り返している回数",i)
```

(2)

```
x = 0
print("x = " , x)
for i in range(1, 11, 1):
       x = x + 10
       print("x = " , x)
```

(3)

```
x = 0
print("x = " , x)
for i in range(1, 6, 1):
       x = x + i
       print("x = " , x)
```

●演習 5　P118

```
x = 0
for i in range(1, 6, 1):
    if i%2==0:
        x = x + 10
        print("x = " , x)
```

第 3 章【学習 14】 演習解答

●演習 1　P123

```
a = [56,3,62,17,87,22,36,83,21,12]
min=a[0]
for i in range(1,10,1):
        if(min>a[i]):
                min=a[i]
print(min)
```

●演習 2　P123

```
import random
a = 5
r = random.randrange(10)
if a==r:
        print("当たり")
        print("a=",a)
        print("r=",r)
elif a>r:
        print("a の方が大きい")
        print("a=",a)
        print("r=",r)
elif a<r:
        print("a の方が小さい")
        print("a=",a)
        print("r=",r)
```

●演習 3　P124

```
def twogoukei(p):
        goukei = 0
        for i in range(0,p,1):
                goukei = goukei+2
        return goukei

goukei = twogoukei(10)
print(goukei)
```

●演習 4　P125

（1）

図表 11 のコード5行目 ″100-0013″ を指定したい郵便番号に変更する。

（2）

WebAPI を利用することで，番組表や現在の時刻で放送されている番組名などをプログラム中に使用することができるものや，キーワード検索の結果を利用することができるものなどもある。

第 3 章【学習 15】 演習解答

●演習 1　P129

・線形探索

```
def linsearch(a,p):
    count=0
    for i in range(0,len(a),1):
        count=count+1
        if a[i]==p:
            print(" 見つかりました")
            print(count)
            break
a = [61,15,82,77,21,32,53]
p = 82
linsearch(a,p)
```

・二分探索

```
def binsearch(a,p):
       i = 0
       j = len(a)-1
       count = 0
       while i<=j:
              m = int((i+j)/2)
              count = count+1
              if a[m]==p:
                     print("見つかりました")
                     print(count)
                     break
              else:
                     if a[m]>p:
                            j=m-1
                     else:
                            i=m+1

a = [25,33,43,51,66,71,88]
p = 43
binsearch(a,p)
```

●演習 2　P130

データ数	2	10	100	1000	10000
線形探索での最大探索回数	2	10	100	1000	10000
二分探索での最大探索回数	2	4	7	10	14

※線形探索では，1つずつデータを比較するので，最大探索数はデータ数と一致するが，二分探索では，データを二つに分けることを繰り返すので，データ数が増えても探索数は線形探索ほどには増えない。

●演習 3　P131

```
def selectionsort(a):
    c = 0
    for i in range(0,len(a),1):
        for j in range(i+1,len(a),1):
            if a[j]<a[i]:
                c = c + 1
                temp = a[i]
                a[i] = a[j]
                a[j] = temp
    print("入れ替えた回数",c)
a = [7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
print(" ソート前",a)
selectionsort(a)
print(" ソート後",a)
```

●演習 4　P131

(1)

```
def quicksort(a,start,end):
    m = int((start+end)/2)
    i = start
    j = end
    while(i<j):
        while a[i] < a[m]:
            i = i+1
        while a[j] > a[m]:
            j = j-1
        if i>=j:
            break
```

```
        temp = a[i]
        a[i] = a[j]
        a[j] = temp
        if i==m:
            m = j
        elif j==m:
            m = i
        i = i+1
        j = j-1
    print("分割終了",a,a[m])
    if start < i-1:
        quicksort(a,start,m-1)
    if end > j+1:
        quicksort(a,m+1,end)

a = [7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
print("ソート前",a)
quicksort(a,0,len(a)-1)
print("ソート後",a)
```

(2) はじめはデータの中央に位置する 40 が軸となる。その後は軸を基準に左右に分かれ，分けた左側の中央の値を軸として選択しながらソートが進んでいく。(40→13→8→2→1→7→4→10→17→16→20→34→26)

第 3 章【学習 16】 演習解答

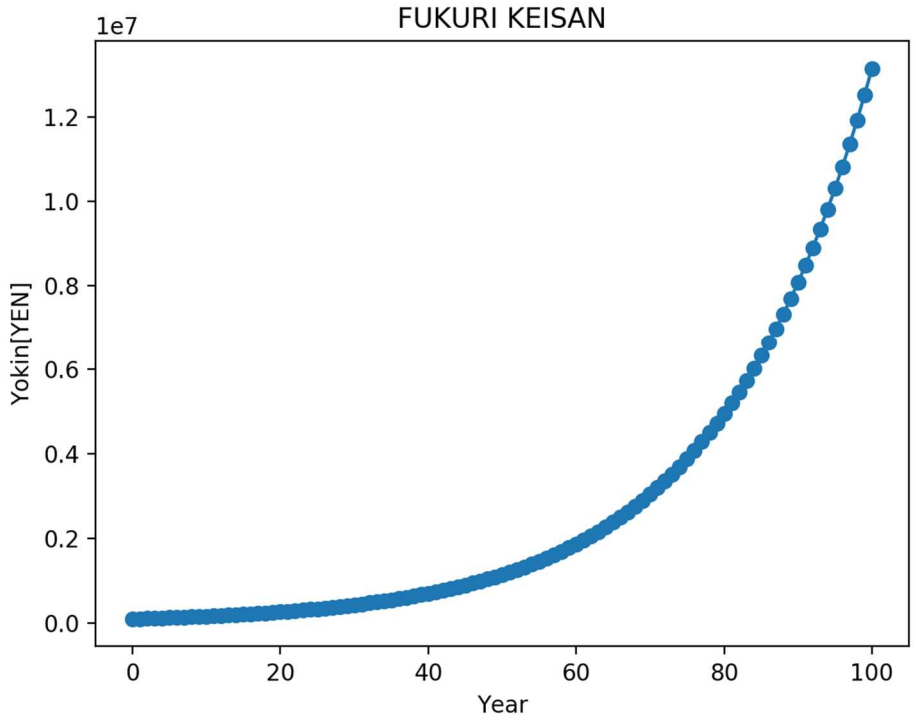
●演習 1　P138

```
import matplotlib.pyplot as plt #プロットオブジェクトをインポート
riritsu = 0.05 #利率
yokin = [100000] #預金配列の最初の値は 10 万
for i in range(100): #i の値を 0～9 まで、100 回繰り返す
        risoku = int(yokin[i]*riritsu) #利息は現在の預金額×利率(整数化)
        yokin.append(yokin[i]+risoku) #配列に計算結果を追加

plt.title("FUKURI KEISAN") #グラフのタイトル
plt.xlabel("Year") #X 軸のラベル
plt.ylabel("Yokin[YEN]") #Y 軸のラベル
plt.plot(yokin, marker="o") #グラフをプロット
plt.show()
```

複利計算は，短い年数で，利率が低い場合には単調な増加傾向しか示さないように見えるが，長い年数や高い利率で計算すると指数関数的に増加する。

※実行結果



●演習 2　P138

各グループの結果の合計値などを見ると200回ほどの試行では，公平な道具とは判断できないが，1000回以上の試行をすれば概ね公平な道具であると判断でき，1000回以上の試行で偏りが見られる場合には，そのサイコロは公平な道具とは言えないだろう。

クラス全体の試行を合計して見ると，例えば 10 班であれば 200×10＝2000 回程度の試行となり，判断するために一定数の試行が必要なことがわかる。

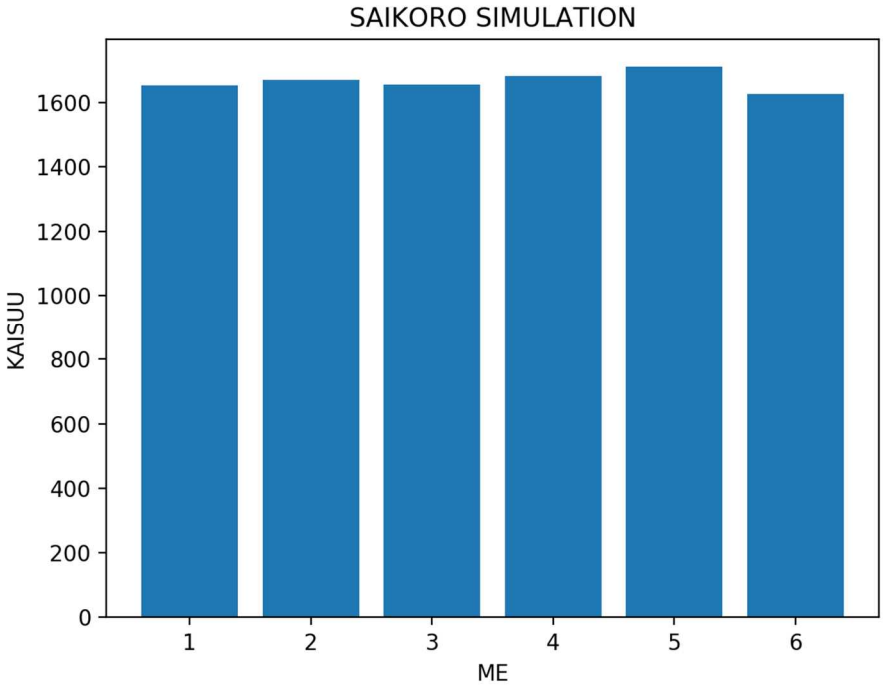
●演習 3　P139

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
saikoro = rd.randint(1, 6+1, 10000)
deme = []
for i in range(6):
    deme.append(np.count_nonzero(saikoro==i+1))

left = [1, 2, 3, 4, 5, 6]
plt.title("SAIKORO SIMULATION")
plt.xlabel("ME")
plt.ylabel("KAISUU")
plt.bar(left, deme, align="center")
plt.show()
```

試行回数が数百回程度では結果にバラツキがあるが，1000回，10000回と増やすとバラツキが相対的に減少し，コンピュータによる一様乱数の使用がサイコロと同様に扱っても良い(妥当)だと判断できる。

※実行結果

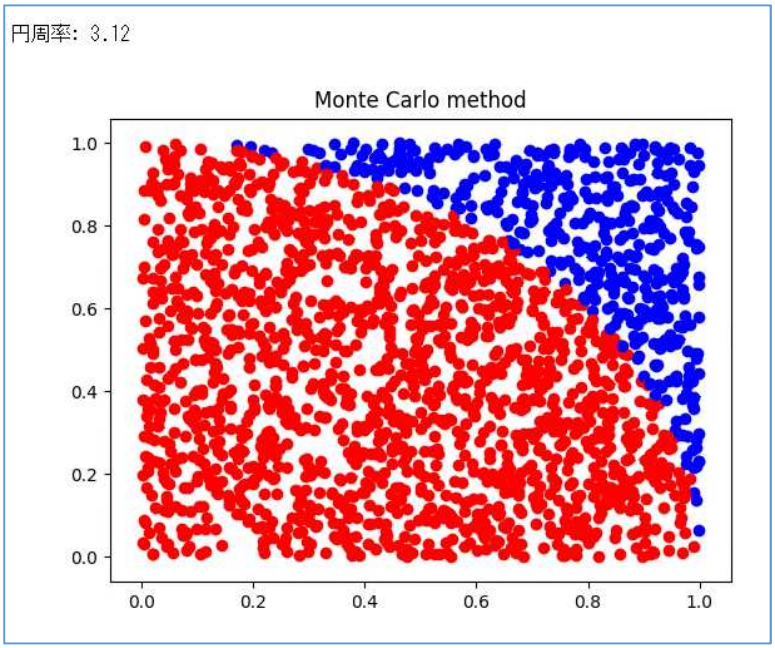


●演習４　P140

```
import numpy.random as rd #乱数を発生させる関数の呼び出し
import matplotlib.pyplot as plt #グラフプロットの呼び出し
totalcount = 2000 #ランダムに打つ点の総数
incount = 0 #円に入った点の数
for i in range(totalcount):
        x = rd.random() #0-1 の範囲の値
        y = rd.random() #0-1 の範囲の値
        if x**2 + y**2 < 1.0: #単位円の中に入ったら
                incount += 1 #入ったカウンターに１を加える
                plt.scatter(x, y, c="red") #赤色でプロット
        else:
                plt.scatter(x, y, c="blue") #青色でプロット
print("円周率:", incount * 4.0 / totalcount) #求まった円周率
plt.title("Monte Carlo method") #グラフのタイトル
plt.show()
```

シミュレーションの回数が少ないと，実行結果にばらつきが見られ，円周率の結果も正しくない事が多いが1000回を超えるような回数であれば結果がある程度正しくなってくる。シミュレーション回数を増やすと時間がかかり，回数が少ないと結果のバラツキが大きくなる。妥当性の判断は，どの程度の正確さを求めるのかによって異なってくる。

人工知能の基礎となる機械学習の分野，金融のリスクマネジメント，科学実験のシミュレーションなど幅広く使われている。



※実行結果

円周率: 3.04

（実行結果については，乱数による計算の為，毎回異なる値が出力される。回数が多いほどより正確な円周率（3.14）に近い値となるが，回数が少ない場合にも 3.14 に近い値が出力される場合がある。）

第 3 章【学習 17】 演習解答

●演習 1　P146

```
import math as math #数値計算ライブラリ
import matplotlib.pyplot as plt #グラフ描画ライブラリ

#１回目(最も遠くに飛ぶ)
dt = 0.01 #微小時間(時間間隔)
v0 = 30 #初速度
g = 9.8 #重力加速度
x = [0] #水平位置の初期値 0
y = [0] #鉛直位置の初期値は 0
angle = 45.0 * math.pi / 180.0 #投げ上げ角度
vx = [v0*math.cos(angle)] #水平方向の初速度
vy = [v0*math.sin(angle)] #鉛直方向の初速度
for i in range(1000):
        vx.append(vx[i]) #微小時間後の水平方向の速度
        vy.append(vy[i]-g*dt) #微小時間後の鉛直方向の速度
        x.append(x[i]+vx[i]*dt) #微小時間後の水平位置
        y.append(y[i]+(vy[i]+vy[i+1])/2.0*dt) #微小時間後の鉛直位置
        if y[i] < 0 : #もし鉛直位置が 0 を下回ったら
                break #ループ中断
plt.plot(x,y) #位置の配列をプロット
plt.title("parabollic motion") #グラフのタイトル
plt.xlabel("distance") #x 軸ラベル
plt.ylabel("height") #y 軸ラベル

#２回目(最も高く飛ぶ)
dt = 0.01 #微小時間(時間間隔)
v0 = 30 #初速度
g = 9.8 #重力加速度
x = [0] #水平位置の初期値 0
y = [0] #鉛直位置の初期値は 0
angle = 90.0 * math.pi / 180.0 #投げ上げ角度
vx = [v0*math.cos(angle)] #水平方向の初速度
```

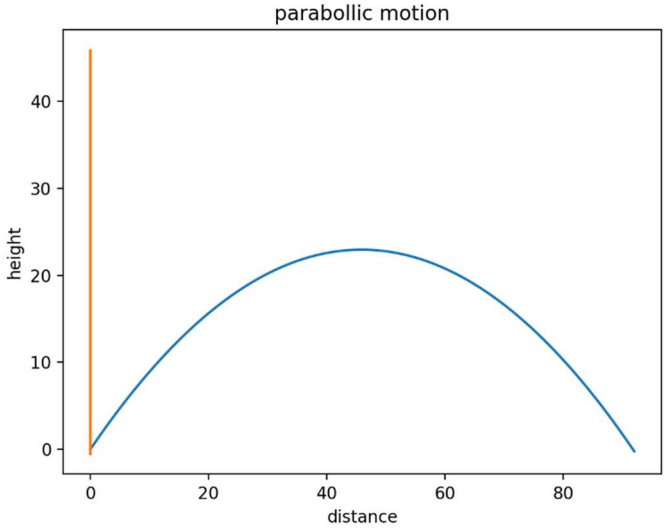
```
vy = [v0*math.sin(angle)] #鉛直方向の初速度
for i in range(1000):
        vx.append(vx[i]) #微小時間後の水平方向の速度
        vy.append(vy[i]-g*dt) #微小時間後の鉛直方向の速度
        x.append(x[i]+vx[i]*dt) #微小時間後の水平位置
        y.append(y[i]+(vy[i]+vy[i+1])/2.0*dt) #微小時間後の鉛直位置
        if y[i] < 0 : #もし鉛直位置が 0 を下回ったら
                break #ループ中断
plt.plot(x,y) #位置の配列をプロット
plt.title("parabollic motion") #グラフのタイトル
plt.xlabel("distance") #x 軸ラベル
plt.ylabel("height") #y 軸ラベル

plt.show()
```

最も遠くに飛ぶのは投げ上げ角度 45 度の時，最も高く飛ぶのは投げ上げ角度0度の時。

※45 度は空気抵抗を考慮しない場合の角度である。応用課題として空気抵抗を考慮した計算をさせてもよい。

※実行結果



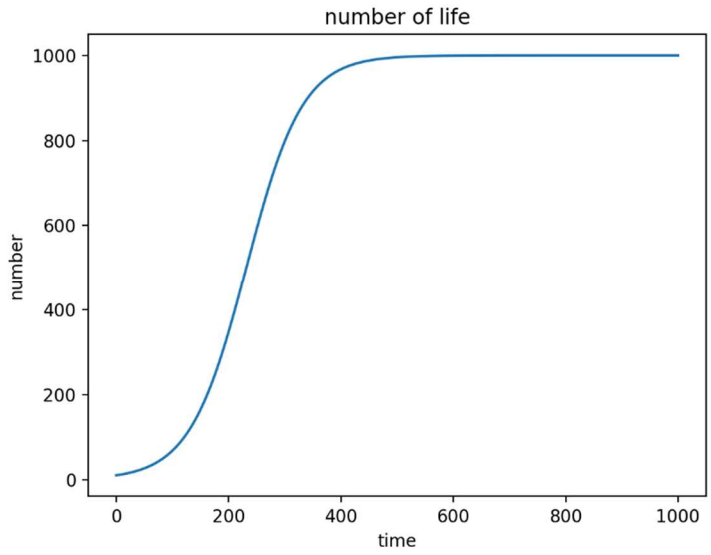
●演習 2　P147

```
import matplotlib.pyplot as plt
#グラフ描画ライブラリ
zouka = 0.02 #増加率
capacity = 1000 #環境収容数
n = [10] #最初の個体数
for i in range(1000):
        zoukasuu = n[i]*zouka #増加数
        gensyousuu = n[i]*(n[i]/capacity)*zouka #減少数
        n.append(n[i]+(zoukasuu - gensyousuu)) #個体数

plt.plot(n) #グラフにプロット
plt.title("number of life") #グラフのタイトル
plt.xlabel("time") #x 軸のラベル
plt.ylabel("number") #y 軸のラベル
plt.show()
```

※同じ時間であれば，増加数を増やすほど個体数の増加は急角度で立ち上がる。同じ増加数であれば，時間を増やせば，個体の増加が急角度で立ち上がるように見える。

※実行結果



●演習 3　P148

```
import matplotlib.pyplot as plt #グラフ描画ライブラリ
import numpy.random as rd #乱数のためのライブラリ
x=[0] #X の初期値を 0 にする
y=[0] #Y の初期値を 0 にする
for i in range(1000):
        x.append(x[i]+rd.random() - 0.45) #X 方向(※片側に寄り気味)
        y.append(y[i]+rd.random() - 0.5) #Y 方向

trace = [x, y] #XY 平面状の物体の座標を trace という名前の配列に代入
plt.plot(*trace) #配列要素のグラフ描画
plt.axis("equal") #グラフの縦横比を等しく
plt.title("random walk") #グラフのタイトルをつける
plt.show() #グラフを表示
```

※乱数の範囲を-0.5 から+0.5 にする事で，X 軸の左右方向，Y 軸の上下方向の現在位置から一様な乱数で移動させることができるから。この値を例えば-0.45 とすれば，乱数の範囲が-0.49 から+0.51 となり，X 軸であればより右方向へ，Y 軸であればより上方向へ移動するような特性を持たせられる。

※実行結果