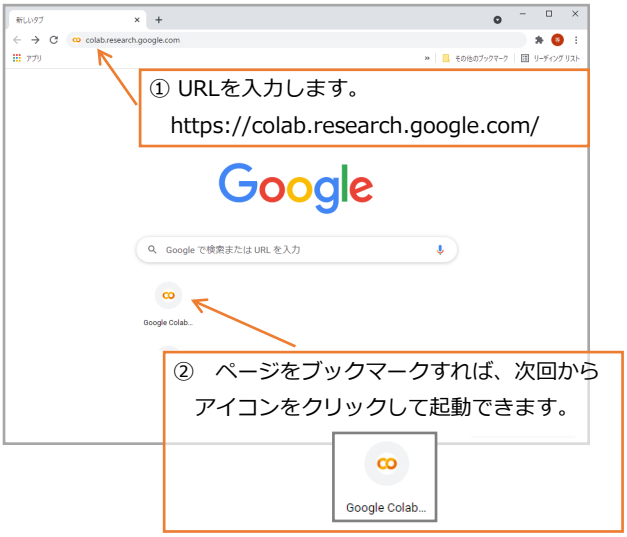
0 プログラミングの準備　ーGoogle Colab とPythonの使い方ー

このシートで学ぶこと

・Google Colabの起動　・Pythonプログラムの入力と実行　・エラー表示　・ファイルの保存と読み込み

1　Google Colab の起動

Google Chrome をインストールして、Googleアカウントも取得してください。Gmailを使用しているならば、アカウントは取得済です。Google Chrome を起動して、Google Colab のURLを入力してアクセスします。

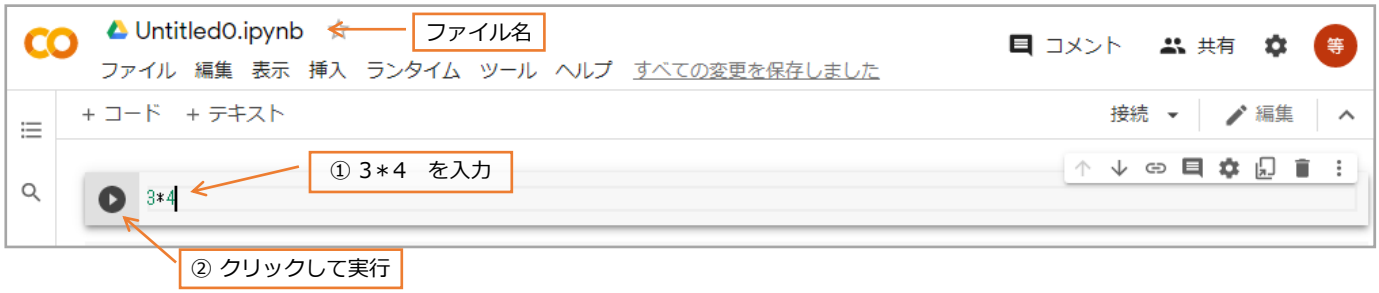


2　プログラムの入力と実行

・ Untitled0.ipynb という名前のファイルが作成され、入力待ちになります。

・ ３＊４　を入力してみます。

・ 「三角ボタン」をクリックして実行します。

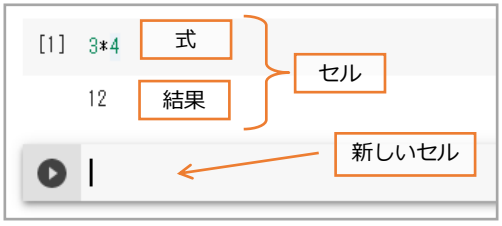


・ 計算式やプログラムはクラウドに送信して処理します。実行ボタンがしばらく回転します。

・ 処理が終わると結果が表示されます。

・ 式「３＊４」と結果「１２」のまとまりをセル（Cell）と呼びます。

・ 実行後には、新たなセルが作られ、入力待ちになっています。



・ セルの内容はいつでも編集できます。

実行すると新しい結果が表示されます。

3　エラー表示

・新しいセルに正しくない式を入力して実行してみます。

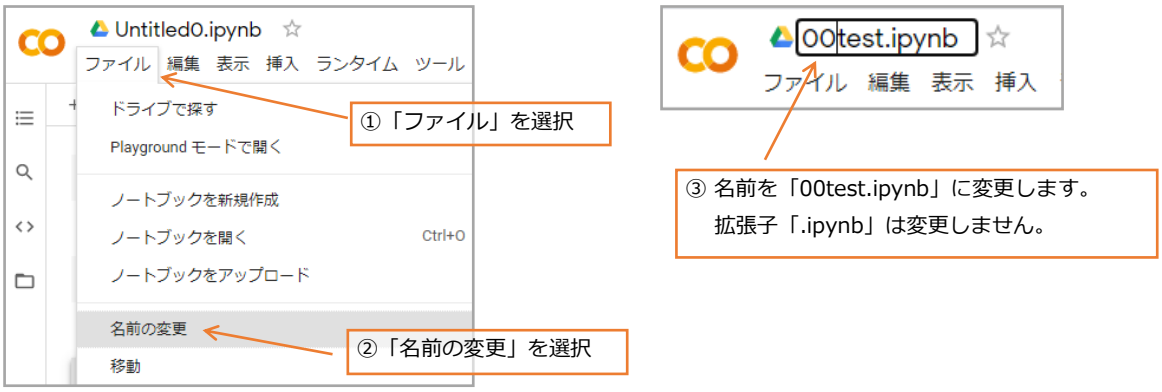
・エラーの箇所とエラーの種類が表示されます。

・修正して実行します。

4　ファイルの保存と読み込み

・ファイルは自動的に Untitled〇〇.ipynb と言う名前で、Googleドライブに保存されています。

・名前の変更と上書き保存は「ファイル」メニューから行います。



5　ファイルの読み込み

・ファイルの読み込みは、新たにGoogle Colab を開くか、「メニュー」→「ノートブックを開く」を選択すると一覧表が表示されるので、目的のファイル（ノートブック）を選択してください。

課題0-1

・ブック（00test.ipynb）に新しいセルを追加して、次の計算をしてください。

①　5×5×3.14　　②　（4＋8）÷ 6　　③　3^3　　④　100÷13のあまり

算術演算子

・算術演算と演算子（記号）の対応

（加算）a＋b　（減算）a－b　（乗算）a＊b　（除算）a／b　（aをbで割った余り）a % b　（aのb乗）a ＊＊ b

1 変数と型

このシートで学ぶこと

・変数　・型（整数型、浮動小数点型、文字列型）　・リスト　・print()　・コメント文

【はじめに】

説明のプログラムは、キーボード入力するかコピー＆ペーストして実行してみてください。＃以降はコメント文なので入力しなくても結構です。PDFファイルなどをコピー＆ペーストした時に、スペースやタブ、インデントが正しくコピーされず、エラーになる場合があります。半角スペースを再入力して書き直してください。

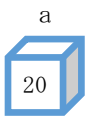
1　変数

プログラム

```
a = 20        #変数aに20を代入
b = 40        #変数bに20を代入
c = a + b     #a+bの結果をcに代入
print(c)      #cの値を表示
```

出力

```
60
```



・変数は数値や文字列を入れる箱です。変数には英数字の名前を付けます。

・変数への代入は = を使用します。= の右の値を左の変数に代入します。

・ ＃で始まる文はコメント文です。プログラムの実行には無関係です。プログラムを後で見直す時に分かりやすいようにコメントします。#の前はTAB(タブ)キーで空けると良いでしょう。

・ print()　は変数や数値、文字列を表示する関数です。数値、文字列を直接表示する場合は　“”　ダブルクォーテーションで囲みます。例　print(“こんにちは”)

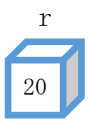
2　型

プログラム

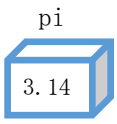
```
r = 20            #変数rに20を代入
pi = 3.14         #変数piに3.14を代入
a = r * r * pi    #20×20×3.14の結果をaに代入
ans ="答え"       #変数ansiに"答え"を代入
print(ans,a)      #ansとaの値を表示
```

出力

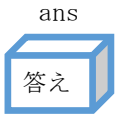
```
答え 1256.0
```



整数型
(int)



浮動小数点型
(float)



文字列型
(str)

・変数には代入できるデータにより、整数型、浮動小数点型、文字列型などがあります。

・整数型と浮動小数点型の変数を演算すると、浮動小数点の結果になります。

・数値の変数と文字列の変数は演算することはできません。

・print()文で複数の値を一度に表示する場合は　,　（カンマ）で区切ります。

コラム　数値の誤差

0.1+0.1+0.1 を計算してみましょう。答えは　0.3　になりましたか？？？

計算結果は　0.30000000000000004　になります。

これは、浮動小数点はコンピュータ内部では２進数で処理しているため、10進数の値はその数値に最も近い近似値になってしまいます。0.1を３回足しても0.3にならないので、浮動小数点を扱う場合、予想外のことが起きることに注意します。

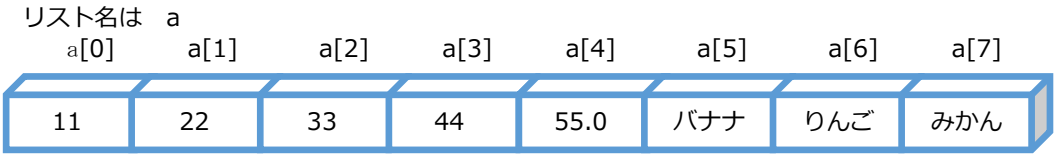
3　リスト

プログラム

```
a = [11,22,33,44,55.0,"バナナ","りんご","みかん"]    #リストaの定義
print(a[1],a[4],a[6])      #リスト1番,4番,6番の表示
```

出力

```
22 55.0 りんご
```



・ リストは複数のデータをまとめて格納する変数です。他のプログラム言語では配列変数と呼ばれます。

・ 多くのデータを変数に格納する場合、それぞれの変数に個別の変数名を付けるのは現実的ではありません。そこで、リストでは名前は一つで、添え字（インデックス）と呼ぶ数値で区別します。

・ 添え字は０から始まります。

・ データの型は何でも良く、混合しても構いません。

・ リストは、複数のデータをまとめて定義する方法　a=[, , ,]　や、ひとつづつ定義する方法　a[0] = 11 があります。

課題1-1

・ブック（01variable.ipynb）に新しいセルを追加して、次のリストｂを定義してください。

・ リストから 300 と トマト を表示する添え字を考え、print() 関数を作成してください。

リストb　[100 ,　200 , 300 , 400 ,”イチゴ”,”レモン”,”トマト”,”スイカ”]

2 逐次処理と繰り返し処理

このシートで学ぶこと

・逐次処理　・繰り返し処理　・for文　・while文

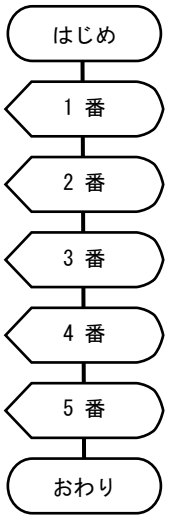
1　逐次処理

プログラム

```
print(1,"番")    #1 番と表示する
print(2,"番")    #2 番と表示する
print(3,"番")    #3 番と表示する
print(4,"番")    #4 番と表示する
print(5,"番")    #5 番と表示する
```

出力

```
1 番
2 番
3 番
4 番
5 番
```



流れ図

・ 命令を「はじめ」から「おわり」の方向へ順番に実行するのが逐次処理です。

・ 逐次処理はプログラムの基本ですが、同じような命令を数多く記述しなければなりません。

2　繰り返し処理　（　for 文　）

プログラム

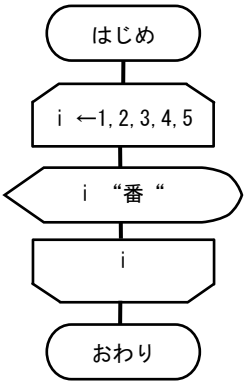
```
for i in range(1,6,1):    #変数iが1から6未満まで１づつ増加
  print(i,"番")           #変数i と 番を繰り返し表示
```

インデント（空白）が重要です

コロンを忘れない

出力

```
1 番
2 番
3 番
4 番
5 番
```



流れ図

・ for文の書式は　for 変数 in range(開始の数,終了の数＋１,増加数）：　です。

・ 変数は自由に名前を付けます。

・ 繰り返す範囲は、開始の数 ≦ 変数 ＜ 終了数　です。増加数はマイナスも指定できます。省略すると増加１になります。

・ for 文最後の：（コロン）の後が繰り返すブロックです。ブロックは半角スペース（インデント）で文字を下げます。インデントの深さ（半角スペースの数）は、１文字以上で機能しますが、推奨で 4 文字、行の文字数を少なくしたいときは２文字にするのが標準です。

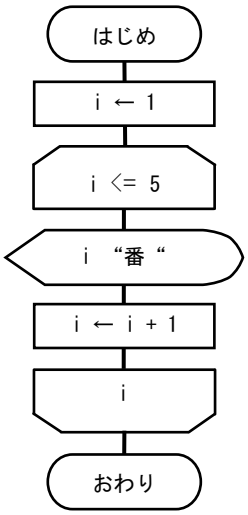
3　繰り返し処理　（　while 文　）

プログラム

```
i=1                #iの初期値を1に設定
while i<= 5:       #iが5以下の場合はブロックを繰り返す
  print(i,"番") #変数iと番 を繰り返し表示
  i+=1             #iの値に1加算
```

出力

```
1 番
2 番
3 番
4 番
5 番
```



流れ図

・while文の書式は　while　　繰り返し条件：　です。

・while文最後の：（コロン）の後が繰り返すブロックです。ブロックは１文字以上空白（ネスト）で文字を下げます。

・i +=1 は　i =i+1 と同じ意味です。この式が無いと常に繰り返し条件が整い、永久に繰り返します。

代入演算子

a = b	# a に b を代入する	a += b	# a = a + b に同じ	a -= b	# a = a - b に同じ
a *= b	# a = a * b に同じ	a /= b	# a = a / b に同じ	a %= b	# a = a % b に同じ
a **= b	# a = a ** b に同じ				

比較演算子

a == b	# a が b と等しい	a != b	# a が b と異なる	a < b	# a が b よりも小さい
a > b	# a が b よりも大きい	a <= b	# a が b 以下である		
a >= b	# a が b 以上である				

課題 2-1

・次のプログラムは、東海道新幹線の駅リストを作り、東京から順番に表示するプログラムです。

・下線の部分に数値や命令を入れて、プログラムを完成してください。

```
station = [”東京”,”品川”,”新横浜”,”小田原”,”熱海”,”三島”,”新富士”,”静岡”,”掛川”,”浜松”,”豊橋”,”三河安城”,”名古屋”,”岐阜羽島”,”米原”,”京都”,”新大阪”]
i=__                    #iの初期値を____に設定
while i<   __:          #iが____より小さい場合はブロックを繰り返す
  print(station[___])   #リストの内容を繰り返し表示
  i+=1                  #iの値に1加算
```

3 条件分岐

このシートで学ぶこと

・条件分岐　・if　else文　・比較演算子　・論理演算子

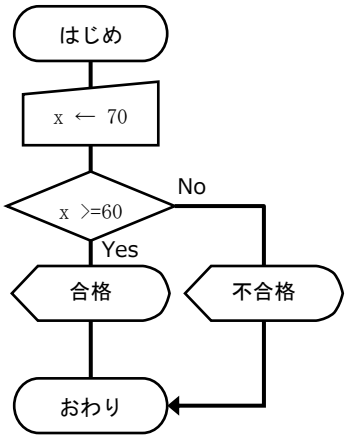
1　条件分岐

プログラム

```
x = 70                  #点数70をxに代入する
if x >= 60:             #xが60点以上の条件に合っていれば次の処理をする
  print(″合格″)         #合格を表示して分岐処理を終わる
else:                   #xが60点以上の条件に合っていなければ次の処理をする
  print(″不合格″)       #不合格を表示して条件処理を終わる
```

出力

```
合格
```



流れ図

・プログラム例は、点数 x が60点以上ならば、「合格」を表示し、そうでなければ「不合格」を表示します。

・if else 文の構文は、if（条件式）：[真の処理]　else：[偽の処理]　です。

条件式を判断して、条件に合っている場合は[真の処理]　、条件に合っていない場合は[偽の処理]をします。

・[偽の処理]が無い場合は　else 文を省略できます。

・条件式は、比較演算子と論理演算子を組み合わせることができます。

比較演算子

```
a == b      # a が b と等しい      a != b      # a が b と異なる      a < b      # a が b よりも小さい
a > b       # a が b よりも大きい   a <= b     # a が b 以下である
a >= b      # a が b 以上である
```

論理演算子

```
（条件式A）and （条件式B）   #条件式AB両方が真のとき真の値になる
（条件式A）or （条件式B）    #条件式ABどちかが真のとき真の値になる
not（条件式A）               #条件式Aが真のとき偽、偽のとき真の値になる
```

課題3-1

・条件分岐のプログラム例で、合格点を80点に変更して実行してください。

・点数を変更して、結果を確認してください。

課題3- 2

・身長と年齢によってジェットコースターに乗車可能か不可能か表示するプログラムを作ります。次のプログラムに適切な字句を追加して完成してください。なお、乗車条件は身長120㎝以上かつ年齢8歳以上です。

```
height = 120
age = 11
if( height >= 120 )       ( age >= 8 ):
   print("乗車できます″)
else:
   print("乗車できません″)
```

2　条件分岐と繰り返し処理の組み合わせ

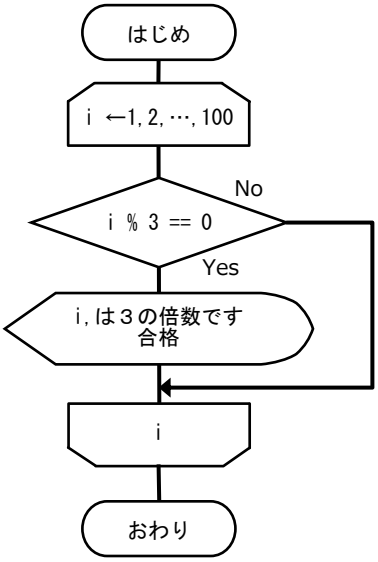
・ １から100の整数の中で、３の倍数を表示するプログラムを作りましょう。

プログラム

```
for i in range(1,101,1):      #変数iを1から100まで1づつ増加して繰り返します
   if i % 3  == 0:            #iを3で割った余りが0か判断します
      print(i,"は３の倍数です")     #3の倍数を表示します
```

出力

```
3 は３の倍数です
6 は３の倍数です
9 は３の倍数です
以下略
```



流れ図

・変数ｉを１から１００まで１づつ増加する繰り返しを作ります。

・ｉを３で割った余りが０の場合、３の倍数なので、３の倍数であることを表示します。

課題3-3

・例題をもとに、１から100の整数の中で、５の倍数または６の倍数を表示するプログラムを作ってください。

課題3-4

・1900年から2100年の間のうるう年を表示するプログラムを作りましょう。

・うるう年の条件は次のとおりです。

① 400で割れる年はうるう年です。

② ４で割れる年で、100で割れない年はうるう年です。

・流れ図を参考にして次のプログラムの＿＿に適切な字句を入れて完成してください。

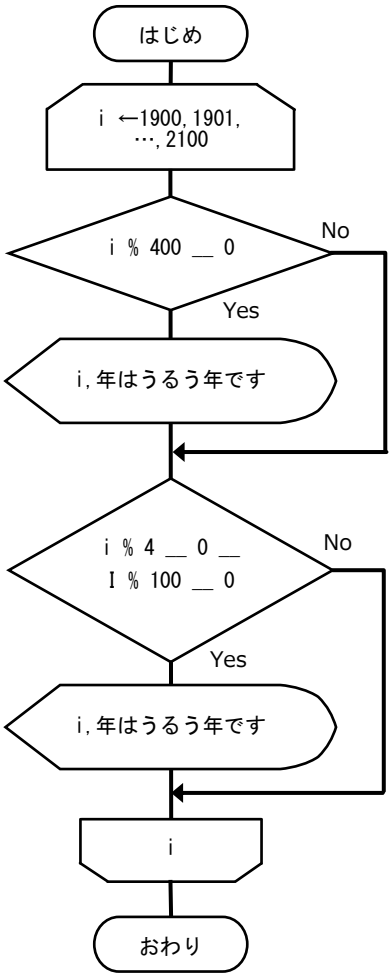
・出力結果が、1900年、2100年を除いて４で割れる年になれば正解です。

プログラム

```
for i in range(_____,_____,_____):
  if i % 400 __ 0:
     print(i,"年はうるう年です")
  if (i % 4 __ 0) ____ (i % 100 ____ 0):
      print(i,"年はうるう年です")
```

出力

```
1904 年はうるう年です
1908 年はうるう年です
1912 年はうるう年です
1916 年はうるう年です
1920 年はうるう年です
1924 年はうるう年です
以下略
```



流れ図

4 関数とスコープ

このシートで学ぶこと

・関数　・スコープ　・ライブラリ　・入力　・return　・import math　・import random　・input()

1　関数

・print()は、文字や数値を入力すると画面に表示する機能を持つ関数です。print()のように最初からプログラム言語に組み込まれている関数の他に、何回も使うプログラムの塊（かたまり）を自分で関数として定義することができます。

プログラム

```
def average(a,b):    #関数の名前をaverage、引数をaとbに定義
  c = (a + b)/2      #a,bの平均を計算してcに代入
  print(c,"平均")    #c,平均　を表示
  return c           #cの値を戻り値として定義

average(3,7)         #aberage()関数を実行①
d = average(10,20)   #aberage()関数を実行して結果をdに代入②
print(d)             #dを表示③
c,d = 150,110        #c,dに数値を代入　c=150 d=110と同じ
e = average(c,d)     #aberage()関数を実行④
```

出力

```
5.0 平均        ①
15.0 平均       ②
15.0            ③
130.0 平均      ④
```

・関数の定義は　def 関数名（引数）：　です。ここでは、関数名は average 引数は a と b です。

・関数名は　if while など最初から使われている予約語以外にします。

・関数に渡す引数（ひきすう）は、複数や、無い場合もあります。

・return 変数　は、関数の計算結果を返します。

・① average(3,7)を実行したので、5.0 平均 が表示されます。

・②　d = average(10,20)　を実行したので、15.0 平均 が表示されます。また、戻り値を d に代入します。

・③ d を表示したので　15.0 が表示されます。

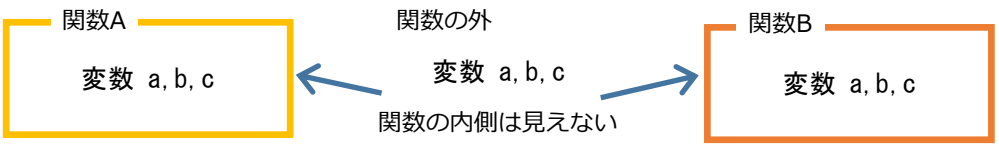
・④ c に 150 、d に 110 を引数として実行したので、130.0 平均　が表示されます。

2　スコープ

・上記の平均を求める関数では、関数内で a,b,c の変数を使用しています。また、関数の外でも　c　=　150　で c を使用しています。もし、関数内と関数外の変数が同じ変数であったら c が意図しない値になり違う答えになってしまいます。計算を正しく行うために、関数の中でどのような名前の変数を使用しているか予め調べる必要が生じてしまします。

そこで、関数は、関数内の変数名と関数外の変数名が同じでも、別の変数として機能します。このように、変数が有効な範囲をスコープと呼びます。

同じ変数名でも関数内の変数と関数外の変数は別のものです。



3　標準ライブラリ

・　プログラムの中で何回も使用する部分は、関数にして呼び出すと入力の手間が省けます。このような便利な関数を複数集めて、他のプログラムからでも呼び出せるようにしたのものがライブラリです。Pythonの使用開始時に標準で用意されているライブラリを標準ライブラリと呼び、代表的なものに、math（数学関数）、random（連数発生）、matplotlib(グラフ表示）などがあります。この他のライブラリはWebで検索して使用法を知ることができます。

プログラム

```
#三角関数
import math                             #mathライブラリを組み込みます
pi = math.pi                            #πはmath.piで定義されています
print(pi)                               #πを表示
print("sin(pi/6) = ",math.sin(pi/6))    #sin関数はmath.sin()です。引数はラジアン角です。
print("cos(pi/6) = ",math.cos(pi/6))    #cos関数はmath.sin()です。引数はラジアン角です。
print("tan(pi/6) = ",math.tan(pi/6))    #tan関数はmath.sin()です。引数はラジアン角です。
print()
#乱数発生
import random                           #rondomライブラリを組み込みます
print("0≦x＜1の実数の乱数")
print(random.random())
print("指定した範囲の実数の乱数　1.5≦x≦2.5")
print(random.uniform(1.5,2.5))
print("0≦x≦の整数の乱数")
print(random.randrange(10))
print("指定した範囲の整数の乱数　0≦x≦100で、3の倍数")
print(random.randrange(0,101,3))
```

出力

```
3.141592653589793
sin(pi/6) = 0.49999999999999994
cos(pi/6) = 0.8660254037844387
tan(pi/6) = 0.5773502691896257 0≦x＜1の実数の乱数
0.7300652333315417
 指定した範囲の実数の乱数　1.5≦x≦2.5
1.9791116231380323
 0≦x≦の整数の乱数
1
 指定した範囲の整数の乱数　0≦x≦100で、3の倍数
90
```

4　キーボード入力

・　input()　関数を使うとキーボードから文字列を入力できます。入力したデータは文字列なので、数値として使用する場合は、int() や float() 関数で数値に変換します。変換しないと文字列なので計算時にエラー表示が出ます。

プログラム

```
a = input("文字を入力してください")
print(a)
x = input("数値を入力してください")
print(int(x))
```

入力を促す文字列です。不要な場合は何も書きません

int()関数で文字列を整数に変換します。実数はfloat()で変換します。

出力

```
文字を入力してください ABCD
ABCD
文字を入力してください 1234
1234
```

文字列の最後にEnterキーを押すと入力できます。

亀 タートルグラフィクス

このシートで学ぶこと

・タートルグラフィクスの使用方法　・図形の描画　・描画の繰り返し

タートルグラフィックスとは

タートルグラフィックスは、亀（タートル）に移動の命令を出し、移動した軌跡を画面に描画する、プログラミング教育用ツールです。LOGOやScratchなどのプログラム言語で使用できますが、Pythonでもライブラリを読み込むことで使用できます。【注意：Python には、タートルグラフィックスの標準ライブラリ turtle が用意されていますが、Google Colaboratory では専用のライブラリを使用します。】

1　正三角形の描画

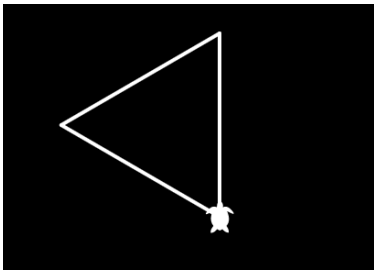
正三角形を描画して、タートルグラフィックスの基本を学びましょう。次のプログラムを実行してください。

プログラム

```
!pip3 install ColabTurtle          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtle.Turtle import *   #最初の２行は、１つのノートブックで最初に１回実行すればよい

initializeTurtle()     # 描画エリアの初期化
for i in range(3):     # ３回繰り返す
    forward(200)       # ２００進む
    left(120)          # 左に１２０度回転
```

出力



・ 描画エリアを初期化すると、亀は上を向いた状態になります。

・ 亀は forward(200) の指示で、200単位直進して止まります。

・ 亀は停止した位置で、left(120)の指示で、左に120°回転します。

・この動きを for 文で３回繰り返すと、正三角形を描画できます。

タートルグラフィックスの関数

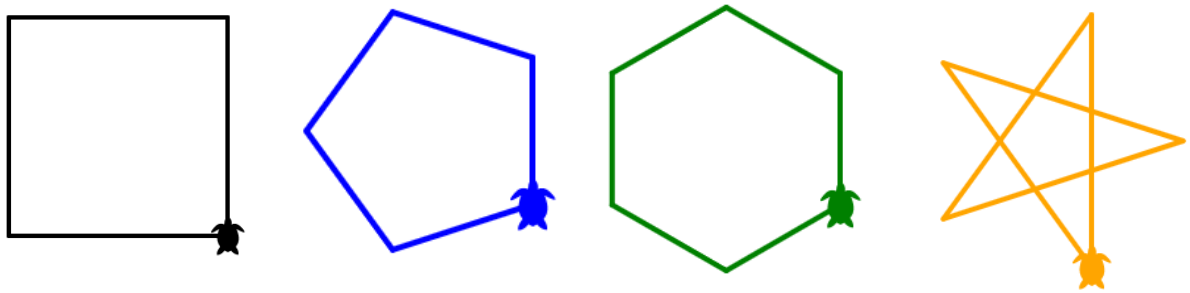
初期化	initializeTurtle( )	描画前に実行します。初期値は次のとおりです。 画面サイズ　800×500（左上 0,0 右下 800,500）　ペン(亀）位置　中央 ペン向き　上　　速さ　4　ペンサイズ　4　　ペン色　白　背景色　black
直進	forward(units)　units 前進	backward(units)　units後退
回転	right(degrees)　degrees右回転	left(degrees)　degrees　左回転
ペン状態	penup() ペンを上げる　pendown()　ペンを下げる	
描画速度	speed(speed) speedは１～10 で10が最速	
移動	goto(x,y)　座標　x,y　に移動	setx(x)　x 座標だけ移動　　sety(y)　y座標だけ移動
ペン表示	showturtle()　ペン（亀）を表示	hideturtle()　ペン(亀）を非表示
ペンサイズ	width(width)　ペンの太さ(太さは１以上の整数)	
背景色	bgcolor(color)　背景色を指定	指定できる色　'white', 'yellow', 'orange', 'red',
ペン色	color(color)　ペンの色	'green', 'blue', 'purple', 'grey', 'black'
ペン形状	shape(name)　ペン形状　指定できる形	‘turtle’, ‘circle’

課題 1

・タートルグラフィックスで次の図形を描画してください。

【アドバイス　一片の長さを調節して図形全体を描画してください。色は好みの色に変えてください。】

① 正方形　② 五角形　③ 正六角形　④ 星



2　図形の繰り返し描画

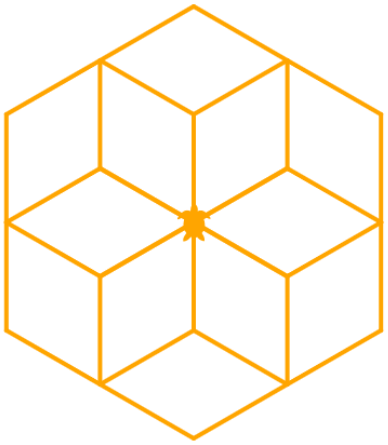
正六角形を作図するとき、開始方向を60°ずつ回転して６個描画すると、次のような図形が描けます。

プログラム

```
initializeTurtle()     # 描画エリアの初期化

speed(8)               #描画速度の指定
bgcolor("white")       #背景色の指定
color("orange")        #線の色指定
for j in range(6):     #正六角形を６個描く
 for i in range(6):    #線の描画を６回繰り返す
    forward(100)       # １００進む
    left(60)           # 左に６０度回転
 left(60)              #描画開始の方向を６０° 回転
```

出力

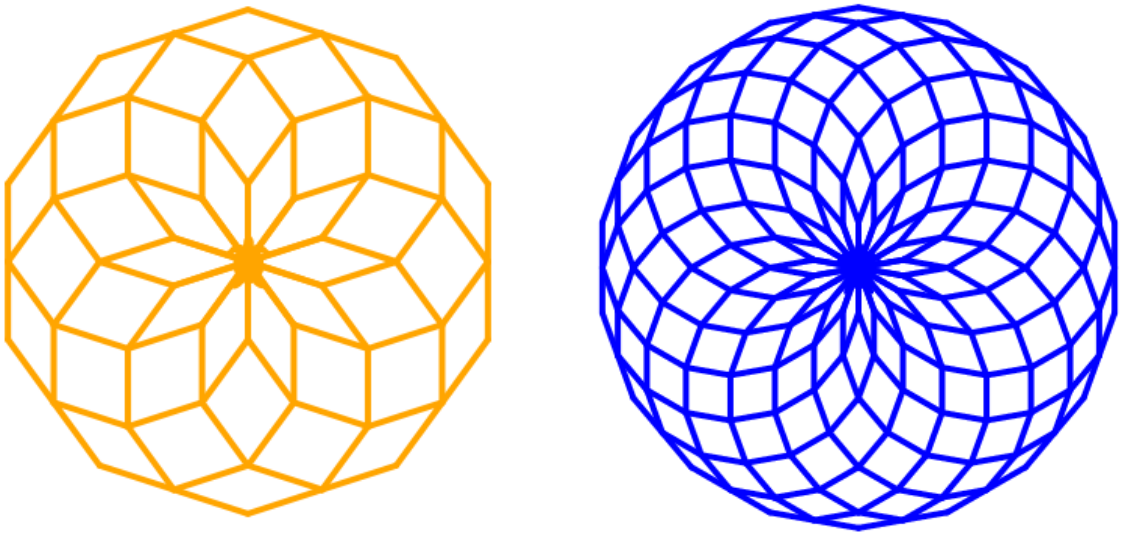


課題 2

・タートルグラフィックスで次の図形を描画してください。

【アドバイス　一片の長さを調節して図形全体を描画してください。色は好みの色に変えてください。】

①　②

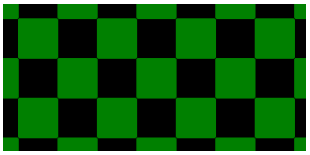


連続文様を描いてみよう

このシートで学ぶこと

・タートルグラフィクスの応用　・要求や条件の分析　・プログラムの分割　・アルゴリズムの工夫

タートルグラフィックスで伝統的な連続文様を描いてみましょう。

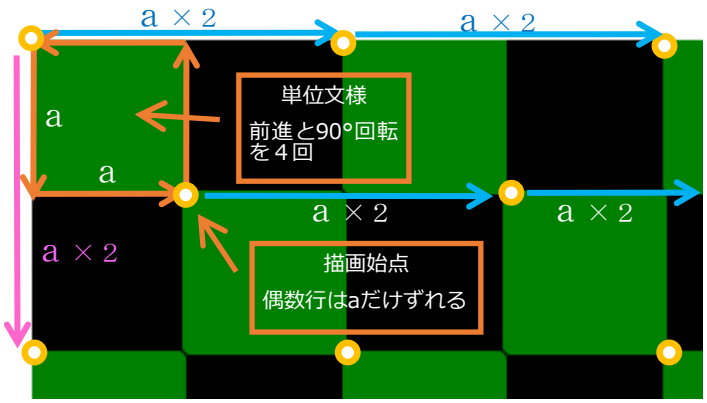


例題　市松模様

手順１　単位文様の決定

連続文様は、小さな単位文様を縦、横に並べて配置しています。描画のし易さ、連続の仕方、タートルグラフィックスの機能などを考えながら、単位文様を決定します。

ここでは、緑の正方形を単位文様とします。黒い正方形は、緑の正方形を飛び飛びに配置した時に、背景色（地の色）でできる文様と考えます。



手順２　単位文様の寸法と描画方法の方針決定

ここでは、正方形一片の長さを基準 a とします。描画方法は、 a 進んで90°回転を４回繰り返すことにします。なお、正方形は緑色で塗られていますが、Colabo　のタートルグラフィックスには塗りの命令がありません。そこで、同じ位置で大きさが少しずつ異なる正方形を描いて、塗った正方形にみえるようにします。

手順３　単位文様の配置位置と繰り返し方法の決定

緑の正方形は、横方向に一片の２倍（基準ａ×２）の間隔で並べればよさそうです。この時、奇数の行と偶数の行では、最初の緑の正方形の位置を横方向に一片分（基準ａ）だけずらします。縦方向には、奇数行と偶数行の２行を集まりとして配置するので、一片の２倍（基準ａ×２）の間隔で並べればよさそうです。

手順４　プログラミングの方針

プログラムの先頭には、背景色や辺の長さなどの初期設定をします。単位文様の描画と配置は、他のプログラムでも利用しやすいように分離し、単位文様の描画は関数としてまとめます。

手順５　プログラミングと改良

プログラミングの方針に従い、途中で動作を確認しながらプログラムを作成します。また、既存のプログラムがあれば、参考にして、改良しながら作成するのも良いでしょう。

プログラム例を実行してみましょう。

プログラム例

```
!pip3 install ColabTurtle          #Claboratory用タートルグラフィックスモジュールの読み込み
from ColabTurtle.Turtle import *  #最初の２行は、１つのノートブックで最初に１回実行すればよい

# 市松模様
initializeTurtle()              # 描画エリアの初期化
# 初期設定
speed(10)                       # 描画速度(1-10)
bgcolor("black")                # 背景色
color("green")                  # 描画色
width(10)                       # 線の太さ
a=80                            # 一片の長さ
left(180)                       # ペンの初期角度（下向き）
```

つづく

```
# 単位文様を描画する関数の定義
def pattern(x, y, a):              # 描き始めの座標と一片の長さを指定
 # 描き始め位置へ移動
 penup()                           # ペンアップ
 goto(x, y)                        # 移動
 pendown()                         # ペンダウン
 # 描画
 for j in range(10):               # 繰り返し１０回（面として塗るために大きさを変えて描画）
  for i in range(4):               # 繰り返し４回（正方形を描画）
    forward(int(a*j/10))           # 線描画（だんだん長くなる）
    left(90)                       # 左回転９０°

# 単位文様の描画位置指定と描画
for y in range(0, 500, a*2):       # 上端からＹ方向に一片の長さ×2移動
  for x in range(0, 800, a*2):     # 左端からＸ方向に一片の長さ×2移動（奇数の列）
     pattern(x, y, a)              # 単位文様描画
  y += a                           # Ｙ方向に一片分移動
  for x in range(a, 800, a*2):     # 左端から右に一片分の位置からＸ方向に一片の長さ×2移動（偶数の列）
      pattern(x, y, a)             # 単位文様描画
```

課題

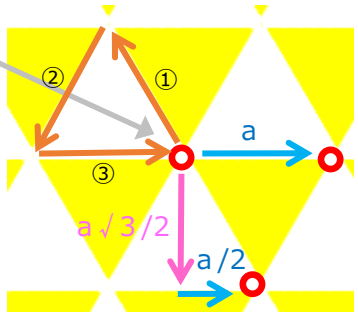
・例題を参考にて次の連続文様を描いてください。

（１）鱗文様

単位文様　①②③

始点は左30°回転方向一片進んだら120°左回転

描画のヒント



・Y方向の描画位置は、奇数偶数行合わせて、三角形の高さ（a√3/2)の2倍移動する。

for y in range(0,500,int(a*1.73)):

・奇数行から偶数行へは

y=y+int(a*1.73/2) 移動する。

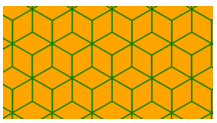
・X方向の描画位置は、一片の長さだけ移動する。

for x in range(0,800,a)

・偶数行は一片の長さの1/2ずれて移動する。

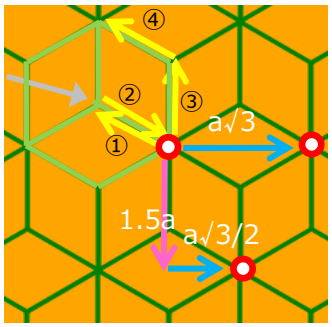
for x in range(int(a/2),800,a)

（２）毘沙門亀甲文様



単位文様①②③④

始点は上０°。①左60°回転してa進んだら②バックして左-60°して0°に戻る。③a進んで120°④a進んで120°。これを３回繰り返す。



・Y方向の描画位置は、奇数偶数行合わせて、六角形の高さ（1.5a)の2倍移動する。

for y in range(0,500,int(1.5*2*a)):

・奇数行から偶数行へは

y=y+int(a*1.5) 移動する。

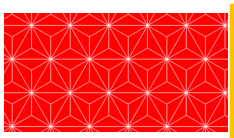
・X方向の描画位置は、六角形の幅a√3だけ移動する。

for x in range(0,800,int(1.76*a))

・偶数行は六角形の幅a√3/2だけずれて移動する

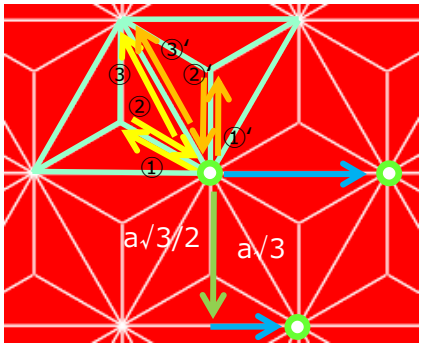
for x in range(int(1.73＊a/2),800,int(1.73*a))

（３）麻の葉文様



単位文様①②③ ①'②'③'

始点は上30°。①左30°回転してa/√3進んだら②バックして左-30°して0°に戻る。③a進んで120°。これを３回繰り返す。　逆三角形①'②'③'も同様。



・Y方向の描画位置は、奇数偶数行合わせて、三角形の高さ（a√3/2)の2倍移動する。

for y in range(0,500,int(a*1.73)):

・奇数行から偶数行へは

y=y+int(a*1.73/2) 移動する。

・X方向の描画位置は、三角形の幅aだけ移動する。

for x in range(0,800,a)

・偶数行は三角形の幅の1/2 a/2だけずれて移動する

for x in range(int(a/2),800,a)