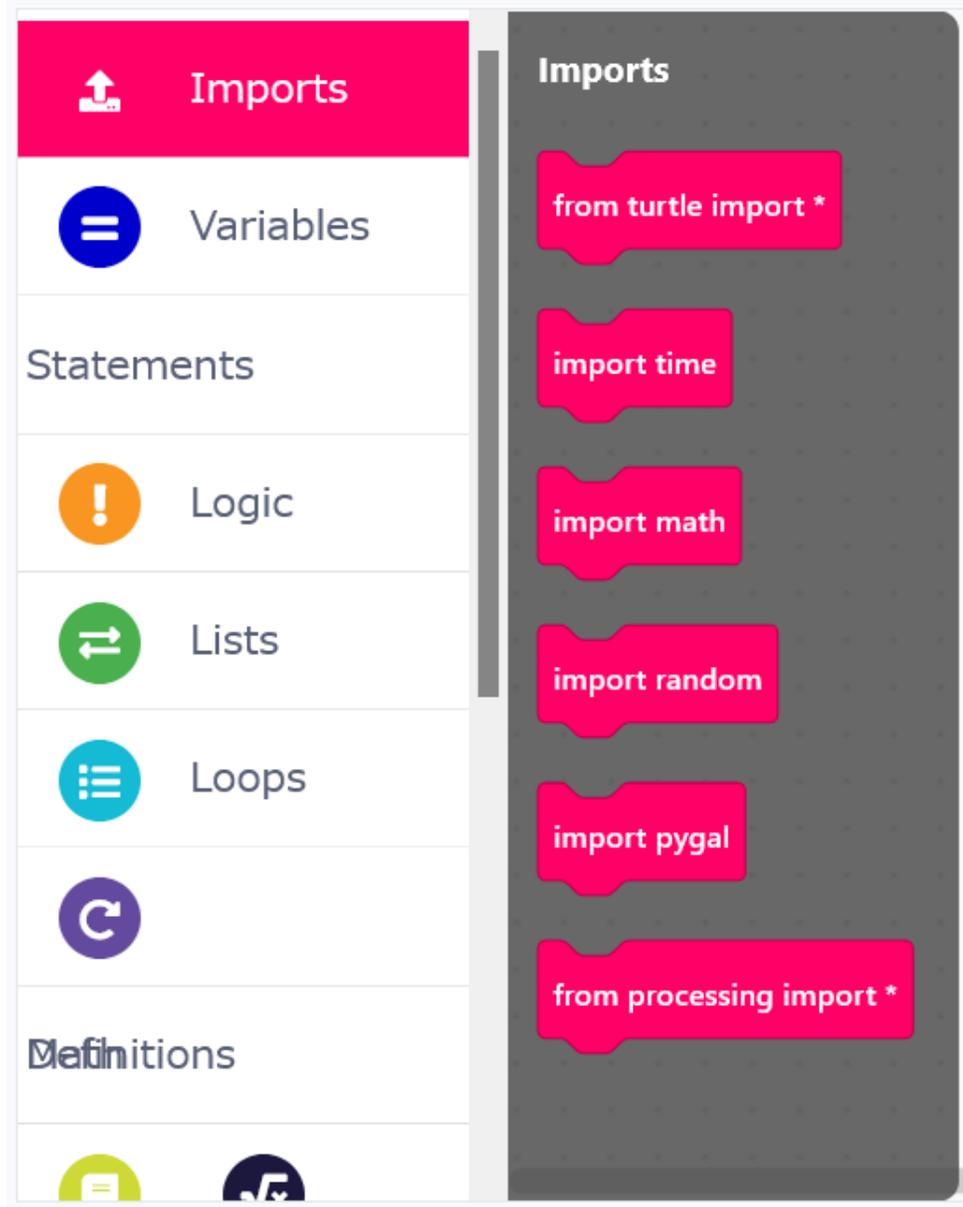


	読み方	項目 (意味)	主なブロック (機能)
 Imports	インポート	ライブラリ読み込み	ライブラリのインポート : turtle、time、math、random、pygal、processing
 Variables	変数	変数	変数の作成、初期値設定
 Statements	ステートメント	命令文	own code、関数呼び出し、print、input、int()、str()、len()、文字列連結 (+)、定数
 Logic	ロジック	論理	条件分岐 (if、elif、else)、条件判断、論理演算 (and、or、not)
 Lists	リスト	リスト	リストの生成 (ただし、名前はVariablesで作成)、リストの初期化、リストの参照
 Loops	ループス	ループス	繰返し (while)、繰返し (for)
 Definitions	デフィニションズ	定義	関数定義、クラス定義、関数呼び出し
 Math	マス	数学	算術演算 (和、差、積、商、余り、整数除算)、三角関数、対数関数、指数関数
 Turtle	タートル	タートル・グラフィックス	タートル・グラフィックスの生成、初期化、描画
 Graphs	グラフス	グラフ	線グラフ、棒グラフ、レーダーチャート
 Random	ランダム	乱数	乱数の生成、初期化、操作
 Processing	プロセシング	Processing言語	Processingプログラム
 Requests	リクエスト	リクエスト・ライブラリ	HTTPリクエストの処理、JSON処理

■ Imports:ライブラリの読み込み

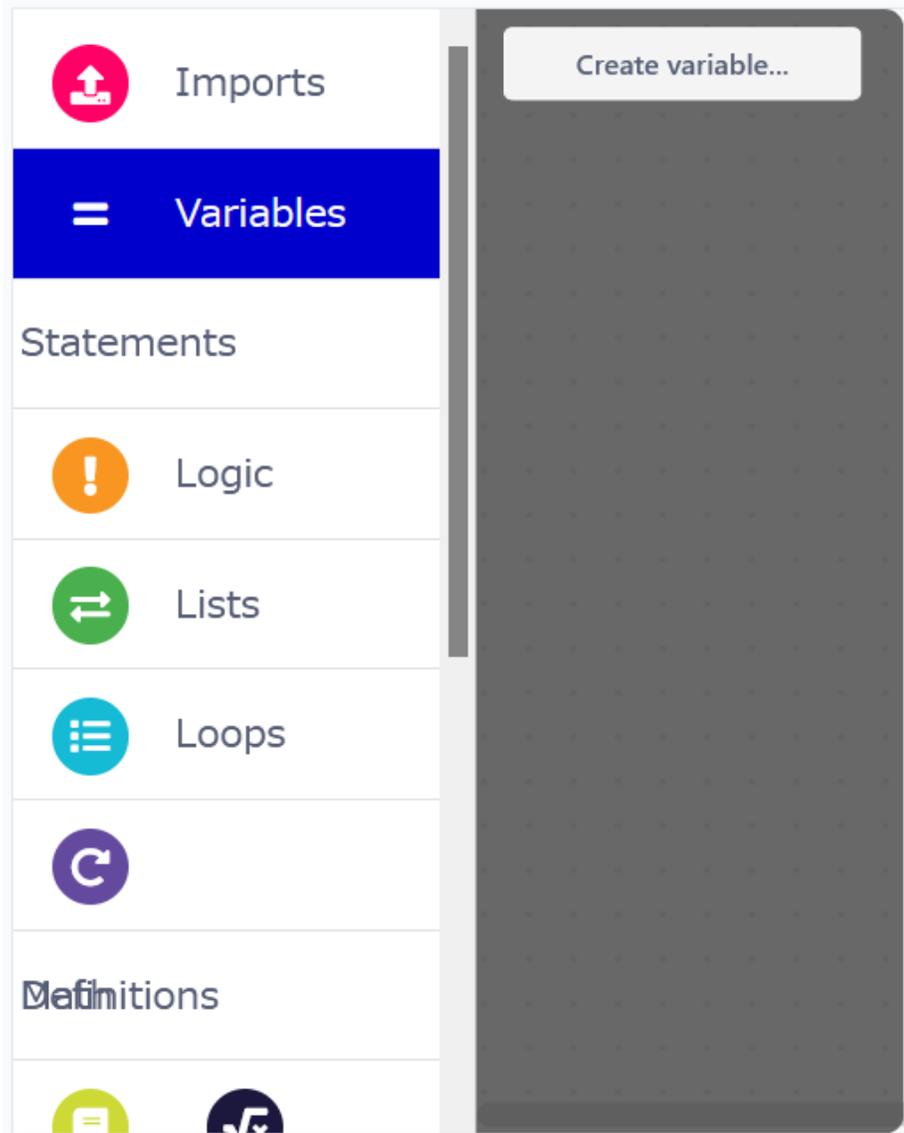


The image shows a screenshot of a code editor's 'Imports' panel. The panel is divided into two main sections: a left sidebar with navigation options and a main workspace with a list of import statements. The sidebar includes 'Imports' (highlighted in pink), 'Variables', 'Statements', 'Logic', 'Lists', 'Loops', and 'Definitions'. The main workspace displays a list of import statements, each highlighted in pink: 'from turtle import *', 'import time', 'import math', 'import random', 'import pygal', and 'from processing import *'. The background of the workspace is dark gray.

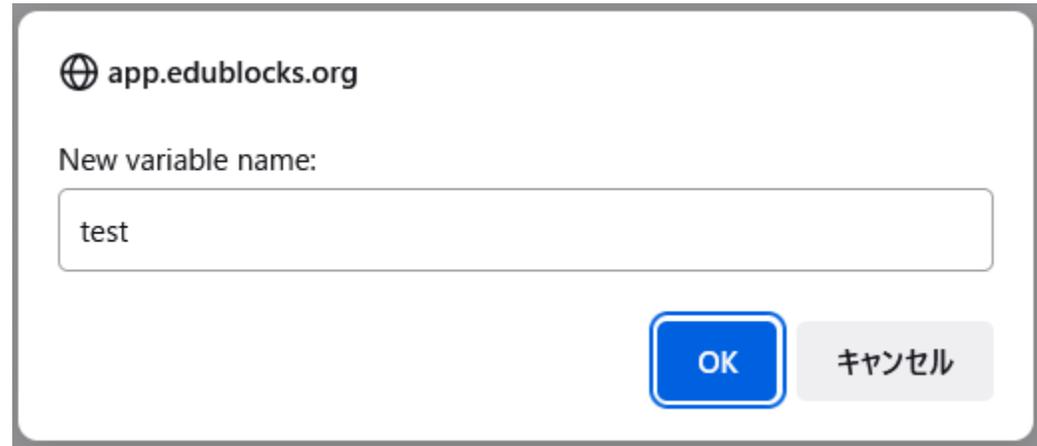
Imports

- from turtle import *
- import time
- import math
- import random
- import pygal
- from processing import *

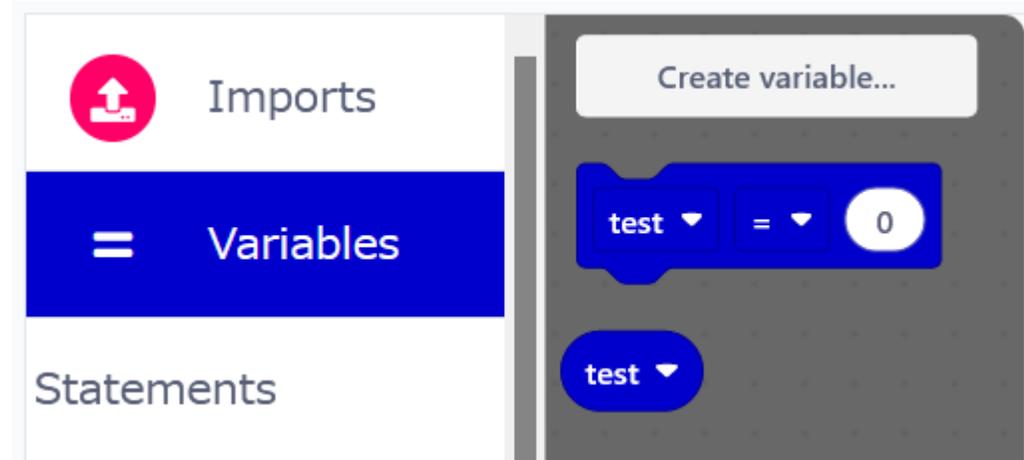
■ Variables : 変数の作成



The screenshot shows the Scratch IDE interface. On the left, a vertical menu is open to the 'Variables' section, which is highlighted in blue. Above the menu is an 'Imports' button with an upload icon. Below the menu are sections for 'Statements' (Logic, Lists, Loops) and 'Definitions'. The main workspace on the right is currently empty, with a 'Create variable...' button at the top.



The dialog box is titled 'app.edublocks.org' and contains the text 'New variable name:'. Below this is a text input field containing the word 'test'. At the bottom right, there are two buttons: a blue 'OK' button and a grey 'キャンセル' (Cancel) button.



The screenshot shows the Scratch IDE interface after the variable has been created. The 'Variables' menu is still open. In the main workspace, a 'Create variable...' button is at the top. Below it, a variable block is visible, showing 'test' in a dropdown menu, followed by an equals sign and a circle containing the number '0'. Below the variable block, there is a 'test' dropdown menu in a blue oval.

Statements : 命令文

The image shows the left sidebar of a code editor. It contains several categories with icons: Imports (pink circle with an upload icon), Variables (blue circle with an equals sign), Statements (orange bar with the text 'Statements'), Logic (orange circle with an exclamation mark), Lists (green circle with a double arrow), Loops (cyan circle with three horizontal lines), and Definitions (purple circle with a refresh icon). At the bottom, there are icons for a document and a code editor.

The image shows a vertical stack of code blocks from a block-based programming language. The blocks are: time.sleep(1), # your own code, function_name (), pass, print("Hello World"), print(Variable), input("What is your name?"), global Variable, len(1), str(1), float(1), round(1.5), len("hello"), "hello" . upper (), *1 + *1, (1), and two empty blocks.

文字列の表示

変数の表示

Statements



A vertical stack of Scratch code blocks on a dark grey background. From top to bottom: an orange 'input' block with the text 'What is your name?'; a 'global' block containing a 'variable' block; an orange 'int' block with the number '1'; an orange 'str' block with the number '1'; an orange 'float' block with the number '1'; an orange 'round' block with the number '1.5'; an orange 'len' block with the string 'hello'; and an orange block for string methods with 'hello' and 'upper' selected.

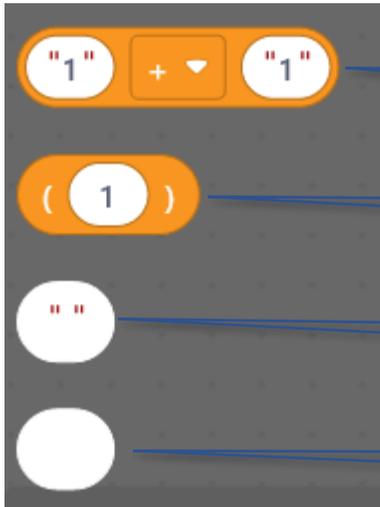
キーボード入力

整数化（小数点以下切り捨て）

文字列に変換

文字列の長さ（文字数）を取得

Statements



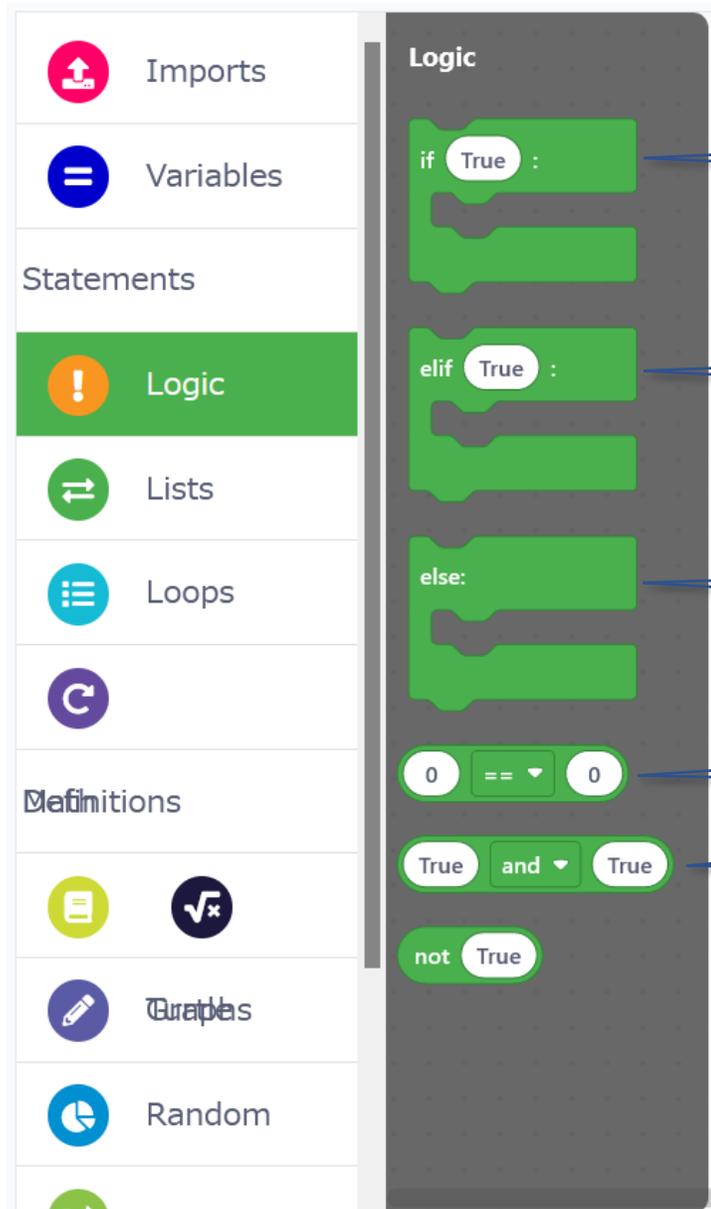
文字列連結

カッコで囲む (演算の優先)

文字列定数を作る

数値定数を作る

Logic : 論理



The image shows the Scratch Logic block palette. On the left, there are categories: Imports, Variables, Statements, Logic (highlighted), Lists, Loops, and Definitions. Under the Logic category, there are several block types: 'if True:', 'elif True:', 'else:', a comparison block with '0 == 0', a logical operator block with 'True and True', and a 'not True' block.

条件分岐 (もし~なら)

条件分岐 (そうでなく、もし~なら)

条件分岐 (そうでなければ)

条件判断 (==、!=、<、<=、>、>=)

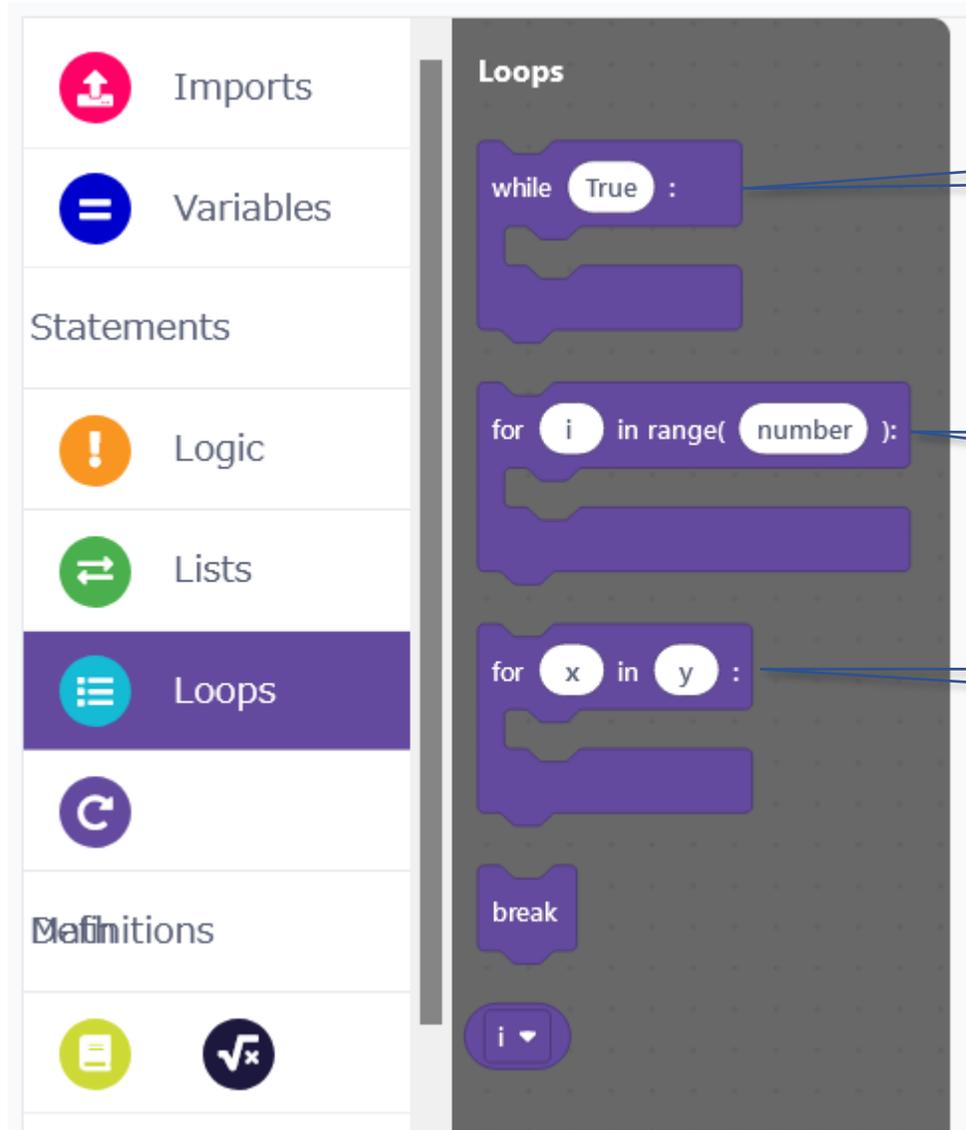
論理演算 (and、or)

■ Lists : リスト

The image shows the Scratch Lists block palette on the left and a workspace on the right. The workspace contains five list-related blocks, each with a callout box explaining its function:

- list = [1, 2, 3, 4]**: リストの初期値設定 (Initial value setting of the list)
- list**: リスト全体の表示 (Display the entire list)
- list [3]**: リストの1つの要素の表示 (Display one element of the list)
- list [3]**: リストの1つの要素の指定 (Specify one element of the list)
- list . append (3)**: リストへの要素追加 (Add element to list)

Loops : ループ



The image shows the Scratch Loops palette on the left, which includes sections for Imports, Variables, Statements, Logic, Lists, Loops (highlighted), and Definitions. The main workspace on the right displays three loop blocks: a 'while True' block, a 'for i in range(number)' block, and a 'for x in y' block. Below these are 'break' and 'i' blocks. Callout boxes on the right provide Japanese descriptions for each loop type.

繰返し (条件が成り立つ間繰り返す)

繰返し (指定回数の繰り返し)

繰返し (リスト要素の繰り返し)

■ Definitions : 定義

The image shows the Scratch 'Definitions' palette on the left, which is organized into several categories: Imports, Variables, Statements, Logic, Lists, Loops, and Math. The 'Definitions' category is highlighted in green. On the right, a dark grey workspace contains several green code blocks. Blue callout boxes with white text point to specific parts of these blocks: '関数の定義' (Function definition) points to a 'def function ():' block; '戻り値の指定' (Return value specification) points to a 'return function' block; 'クラスの定義' (Class definition) points to a 'class class :' block; and '関数の呼び出し' (Function call) points to a 'function ()' block. Below these, there is also a 'self. i = 0' block.

Imports

Variables

Statements

Logic

Lists

Loops

Math

Definitions

関数の定義

戻り値の指定

クラスの定義

関数の呼び出し

■ Math : 数学

math.degrees()

math.exp()

math.fabs()

math.factorial()

math.floor()

math.hypot()

math.log()

math.log10()

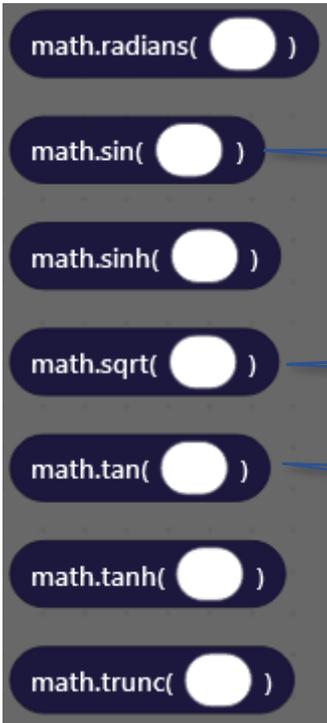
math.pow()

指数関数

対数関数

べき乗 (累乗)

■ Math : 数学

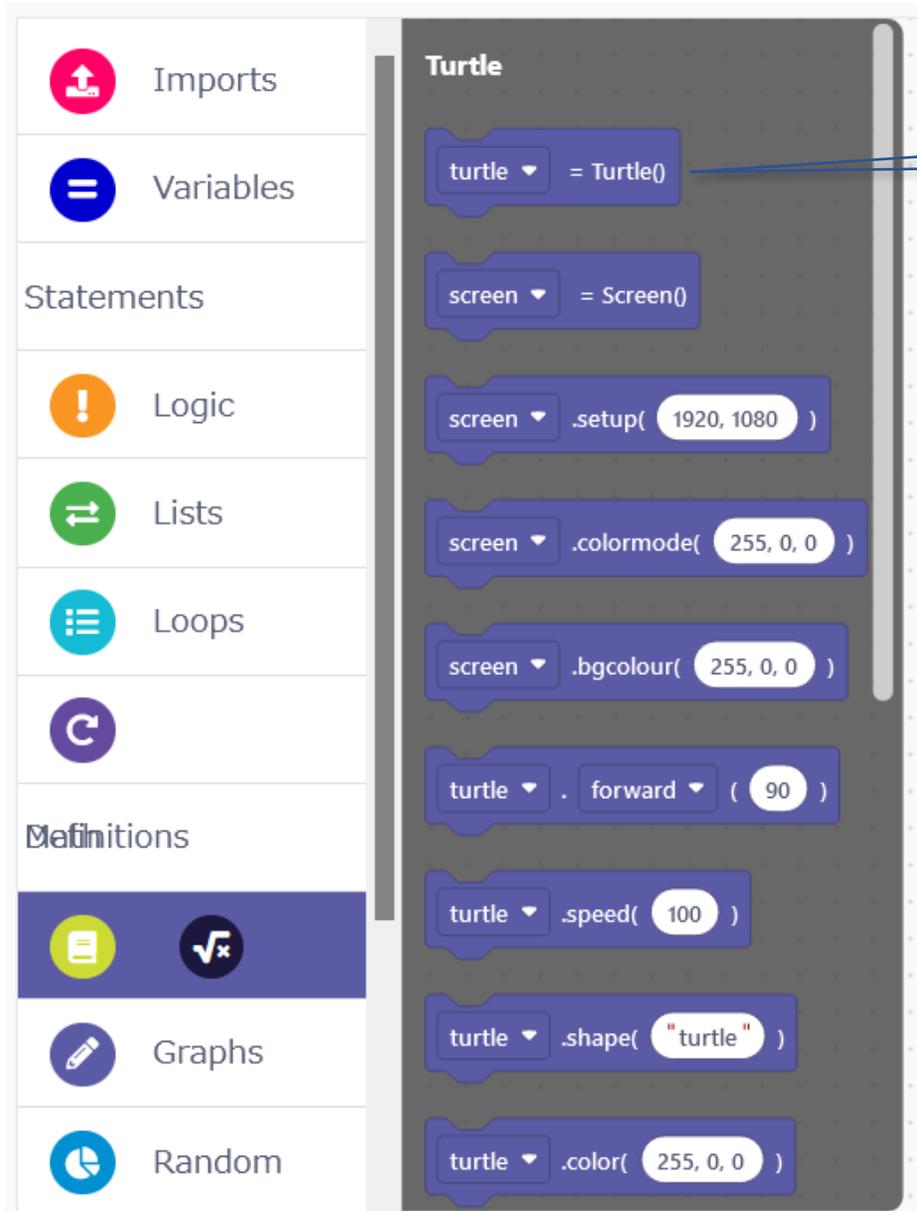


三角関数 : sin

平方根

三角関数 : tan

■ Turtle : タートル・グラフィックス



The image shows a Scratch script for Turtle Graphics. The script is titled "Turtle" and contains the following blocks:

- Imports:** A red "Import" button.
- Variables:** A blue "Variables" button.
- Statements:**
 - Logic: An orange "!" button.
 - Lists: A green "↔" button.
 - Loops: A cyan "☰" button.
 - Definitions: A purple "↻" button.
- Code Blocks:**
 - `turtle = Turtle()`
 - `screen = Screen()`
 - `screen .setup(1920, 1080)`
 - `screen .colormode(255, 0, 0)`
 - `screen .bgcolor(255, 0, 0)`
 - `turtle .forward (90)`
 - `turtle .speed(100)`
 - `turtle .shape("turtle")`
 - `turtle .color(255, 0, 0)`

タートルの生成

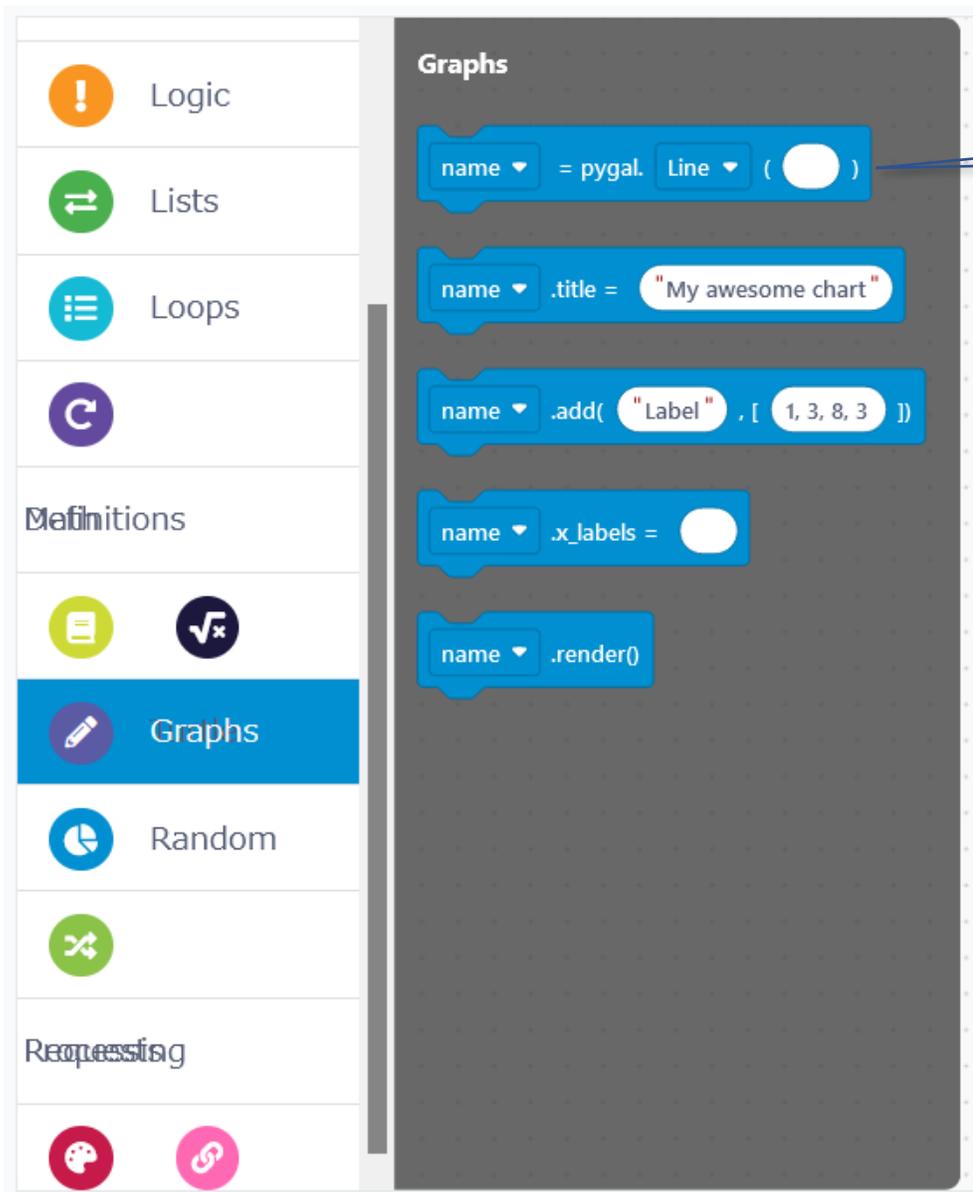
■ Turtle : タートル・グラフィックス



```
turtle .pencolor( 255, 0, 0 )
turtle . begin_fill ( )
turtle .fillcolor( 255, 0, 0 )
turtle .pen up ( )
turtle .width( 20 )
turtle .circle( 50 )
turtle .goto( 5, 5 )
```

The image shows a vertical stack of seven Scratch code blocks for the turtle library. Each block starts with a 'turtle' dropdown menu. The blocks are: 1. '.pencolor(255, 0, 0)' - sets the drawing color to red. 2. '. begin_fill ()' - starts filling the shape. 3. '.fillcolor(255, 0, 0)' - sets the fill color to red. 4. '.pen up ()' - lifts the pen so the next movement doesn't draw a line. 5. '.width(20)' - sets the line thickness to 20 pixels. 6. '.circle(50)' - draws a circle with a radius of 50 pixels. 7. '.goto(5, 5)' - moves the turtle to the coordinates (5, 5).

■ Graphs : グラフ



The image shows the Scratch 'Graphs' block palette on the left and the workspace on the right. The palette includes categories: Logic, Lists, Loops, Definitions, Graphs (selected), Random, and Requests. The workspace contains five blue 'Graphs' blocks:

- name ▾ = pygal. Line ▾ ()
- name ▾ .title = "My awesome chart"
- name ▾ .add("Label" , [1, 3, 8, 3])
- name ▾ .x_labels =
- name ▾ .render()

グラフの生成

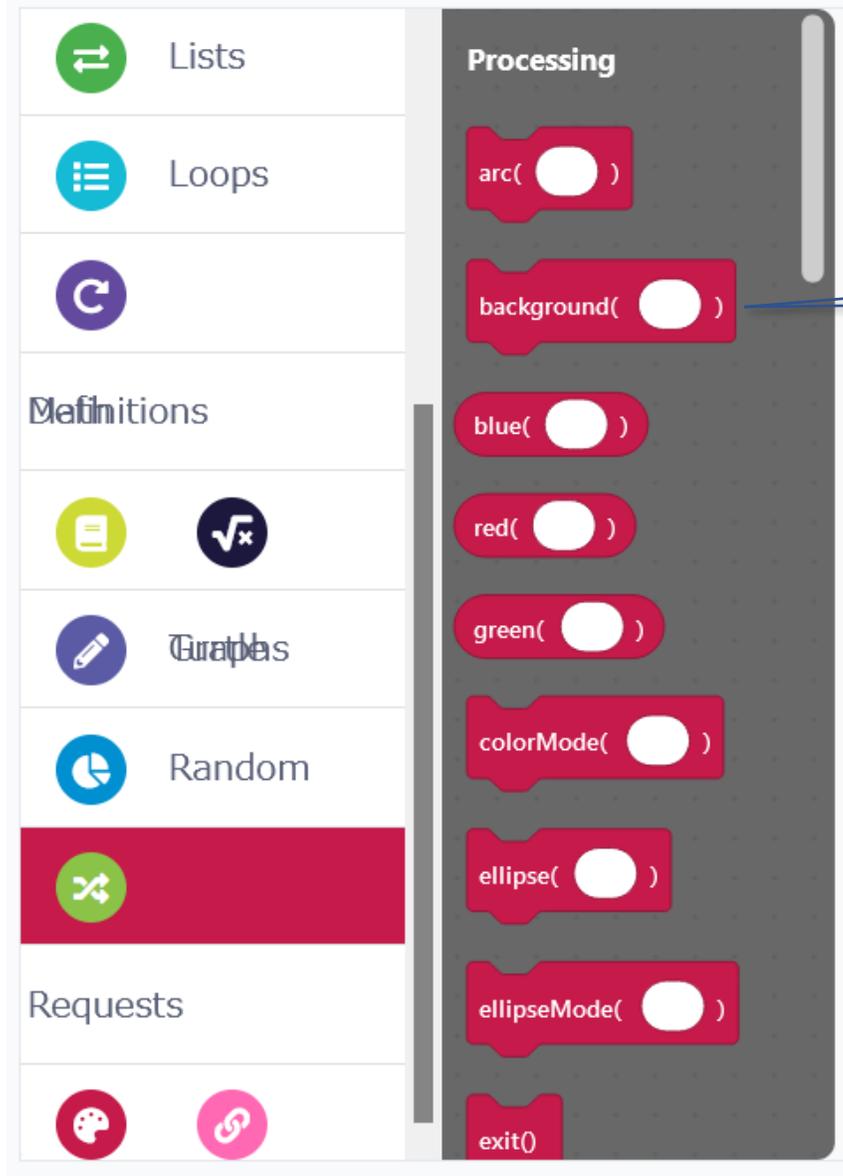
Random : 乱数

The image shows a software interface with a sidebar on the left and a main panel on the right. The sidebar contains several categories: Logic, Lists, Loops, Definitions, and Repeating. The 'Random' category is highlighted in green. The main panel displays a list of random number generation functions, each with a green button and a white toggle switch:

- random.choice()
- random.randint()
- random.random()
- random.randrange()
- random.seed()
- random.shuffle()
- random.uniform()

乱数の生成

■ Processing : Processing言語

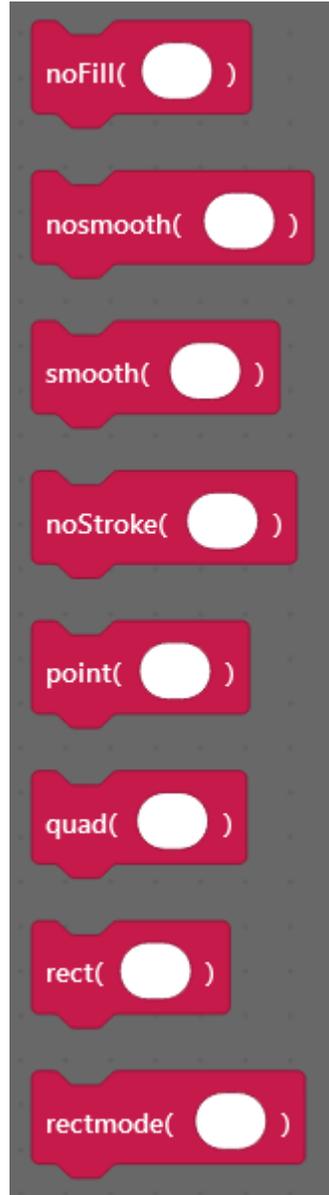


背景の指定

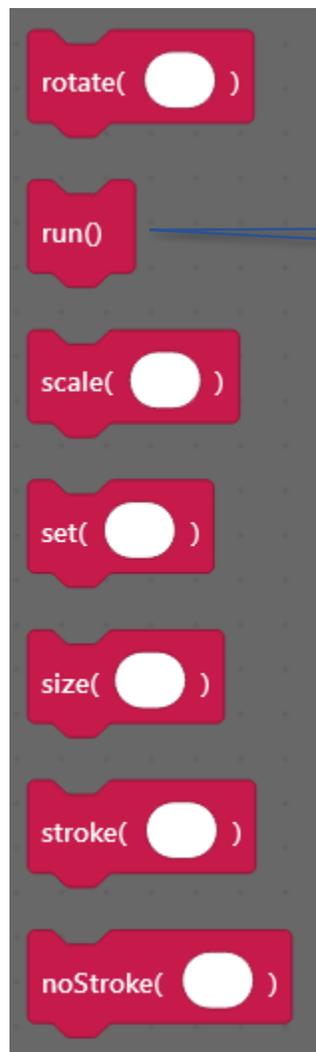
■ Processing : Processing言語



■ Processing : Processing言語

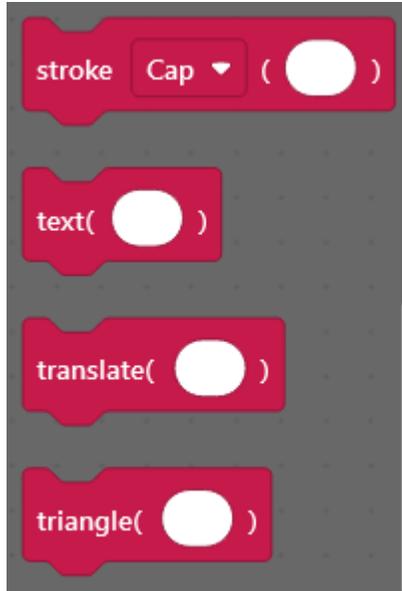


■ Processing : Processing言語



プログラムの実行

■ Processing : Processing言語



■ Requests : リクエスト・ライブラリ

```
import requests

import json

r = requests.get( "https://api.github.com/events" )

r = requests.post( 'https://httpbin.org/post', data = {'key':'value'} )

r = requests.put( 'https://httpbin.org/put', data = {'key':'value'} )

r = requests.delete( 'https://httpbin.org/delete' )

r = requests.head( 'https://httpbin.org/get' )

r = requests.options( 'https://httpbin.org/get' )
```

HTTPリクエスト

■ Requests : リクエスト・ライブラリ

```
r .url  
r .url ()  
r .cookies[ 'example_cookie_name' ]  
r .json()[ "name" ]  
jar = requests.cookies.RequestsCookieJar()  
jar .set( 'tasty_cookie', 'yum', domain='httpbin.org', path='/cookies' )  
r .headers.get( 'content-type' )  
r .raise_for_status()
```

JSONの指定