プログラミング教室のテクノロ

Pythonの道

トレーニングドリル④

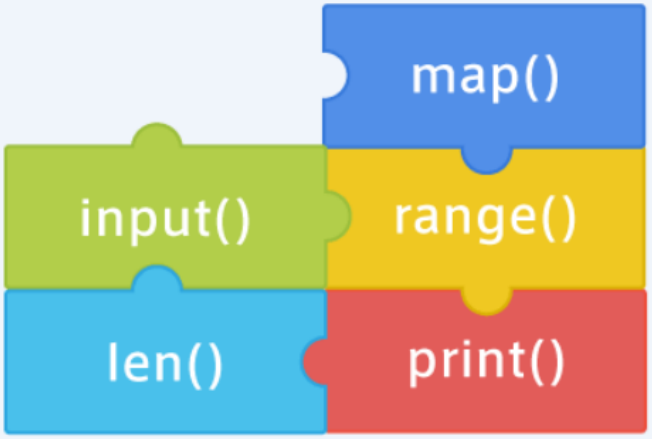
もくじ

・じゃんけんプログラムを作ろう

関数とは

関数とは、ある処理をまとめたプログラムのかたまりで、printも関数の1つである。本来、コンソールに文字を出力するためには色々なコードを書く必要がある。しかし、printという関数のおかげでprintとだけ書けばいいようになっている。printの他にもPythonにはいくつかの便利な関数が用意されており、それらを使うと様々な処理を簡単に行うことができる。

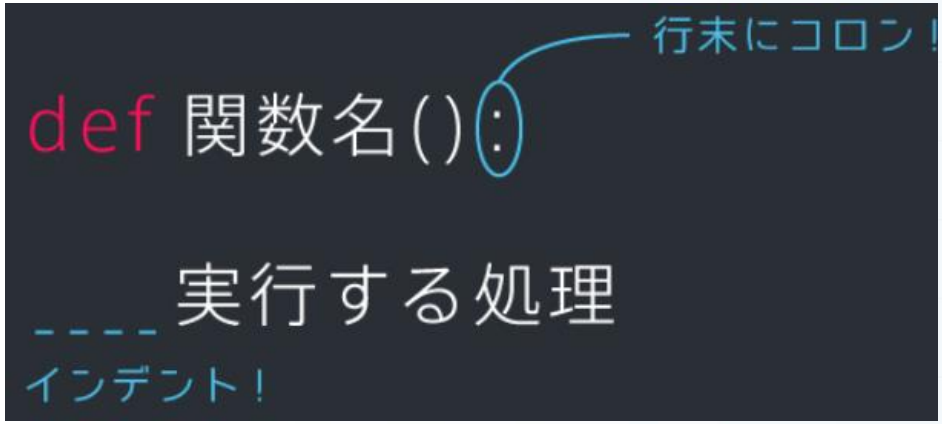
様々な関数を組み合わせてプログラミングする



関数を作ってみよう

関数は「def 関数名():」のように定義する。
関数の処理の内容は、インデントして書いていく。

関数の定義



関数の定義

```
def  hello():
      print('Hello  World')
```

関数の使い方

関数は定義しただけでは実行されないので、呼び出して実行してみよう。関数名()のように() をつけて呼び出す。ただし、関数は定義した後でしか呼び出せない。

```
def hello():
    print('Hello World')

hello()
```

関数の定義

関数の呼び出し

※ 必ず関数を定義した後、呼び出す

⇨ Hello World

練習

【問題】

関数print_handを定義してください

関数print_handを呼び出してください

練習

【答え】

```
# 関数print_handを定義してください
def print_hand():
    print('グーを出しました')

# 関数print_handを呼び出してください
print_hand()
```

引数とは

関数を呼び出す際に、関数に値を渡すことができる。
この値のことを引数といい、引数を渡すと関数の中でその値を利用することができるようになる。
関数に引数を渡せると、その値によって関数の処理結果を変えることができる。

引数を受け取る関数

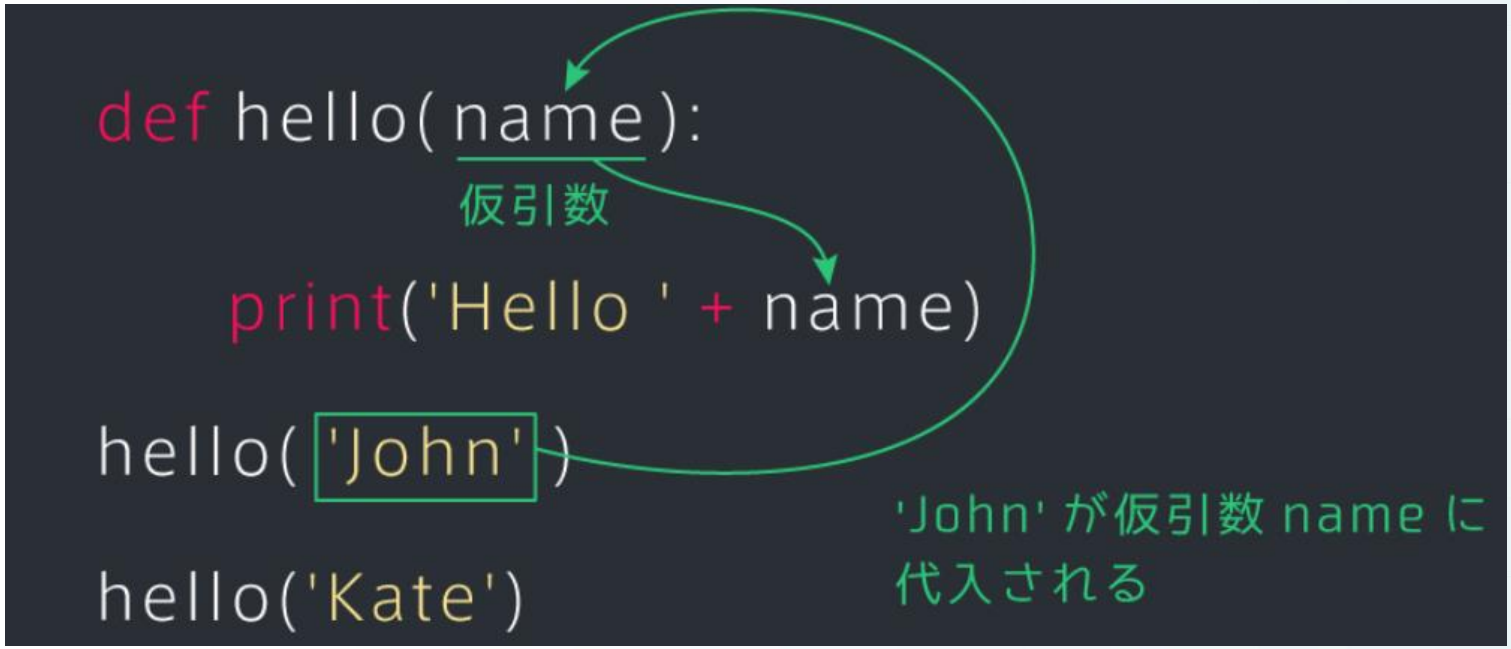
関数に引数を渡すには、まず引数を受け取れる関数を定義する必要がある。そのためには、関数の定義部分で、引数を受け取るための箱となる変数（仮引数（かりひきすう））を指定する。

```
def 関数名（仮引数）:
                引数を受け取るための仮引数を指定

    実行する処理
```

```
def hello(name):
            仮引数
    print('Hello' + name)
```

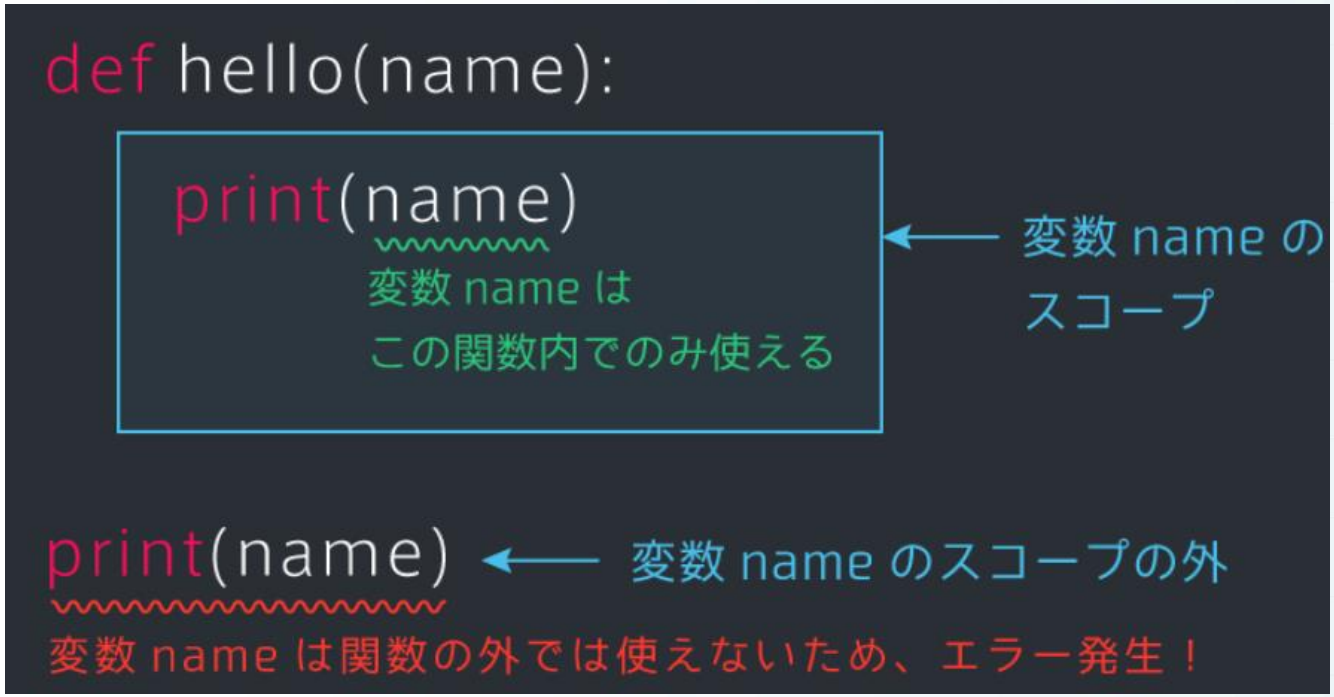
関数に引数を渡す

関数に引数を渡すには、関数名(引数)として関数を呼び出す。
渡された引数は、関数の仮引数に代入され、その値を関数の処理の中で用いることができる。



スコープ

変数には、その変数が使える範囲が存在する。その範囲のことをスコープと呼ぶ。仮引数や関数の中で定義した変数のスコープは関数の中だけとなり、その変数を関数の外で使うことはできない。




```
NameError: name 'name' is not defined
エラー発生！
```

練習

【問題】

```
# 引数を受け取れるようにしてください
def print_hand():
    # 「○○を出しました」と出力されるように書き換えてください
    print('グーを出しました')
```

```
# 引数に文字列グーを入れてください
print_hand()
```

```
# 引数を文字列パーとして関数print_handを呼び出してください
```

練習

【答え】

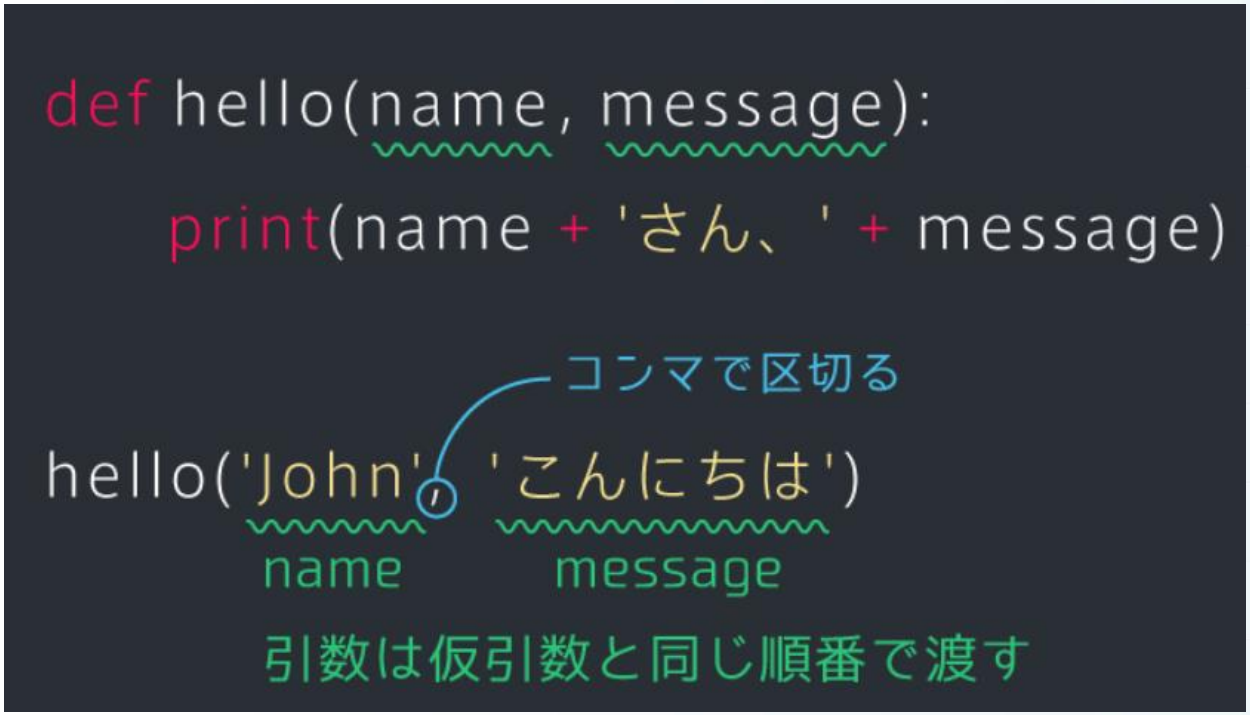
```
# 引数を受け取れるようにしてください
def print_hand(hand):
    # 「○○を出しました」と出力されるように書き換えてください
    print(hand + 'を出しました')

# 引数に文字列グーを入れてください
print_hand('グー')

# 引数を文字列パーとして関数print_handを呼び出してください
print_hand('パー')
```

複数の引数を持つ関数

引数は複数渡すこともできる。関数が複数の引数を受け取るためには、仮引数をコンマ（,）で区切って定義する。引数を受け取る順番は自由に決めることができる。また、引数は左から順番に「第1引数、第2引数・・・」というように呼ぶ。

関数に複数の引数を渡す

複数の引数を渡すには以下のようにする。
引数の順番は、対応する仮引数の順番と同じにする必要がある。



⇨ johnさん、こんにちは

練習

【問題】

```
# 名前を第2引数で受け取れるようにしてください
def print_hand(hand):
    # 「○○は□□を出しました」と出力されるように書き換
      えてください
    print(hand + 'を出しました')

# 第2引数に文字列「にんじゃわんこ」を入れてください


# 第2引数に文字列「コンピューター」を入れてください


```

練習

【答え】

```
# 名前を第2引数で受け取れるようにしてください
def print_hand(hand, name):
    # 「○○は□□を出しました」と出力されるように書き換えてください
    print(name + 'は' + hand + 'を出しました')

# 第2引数に文字列「にんじゃわんこ」を入れてください
print_hand('グー', 'にんじゃわんこ')

# 第2引数に文字列「コンピューター」を入れてください
print_hand('パー', 'コンピューター')
```

引数の初期値

引数には初期値を設定することができる。図のように引数が省略されたとき、初期値が与えられていれば代わりの値として初期値が使われる。

```
def hello(name, message='こんにちは'):
                        初期値を設定できる

    print(name + 'さん、' + message)

hello('John', 'こんばんは')

hello('Kate')
        引数 message が省略された場合、
        初期値「こんにちは」が用いられる
```

Johnさん、こんばんは

引数に指定した値が代入されている

Kateさん、こんにちは

初期値「こんにちは」が代入されている

練習

【問題】

```
# 仮引数nameの初期値を設定してください
def print_hand(hand, name):
    print(name + 'は' + hand + 'を出しました’)

# 引数に文字列グーのみを入れてください
print_hand()
```

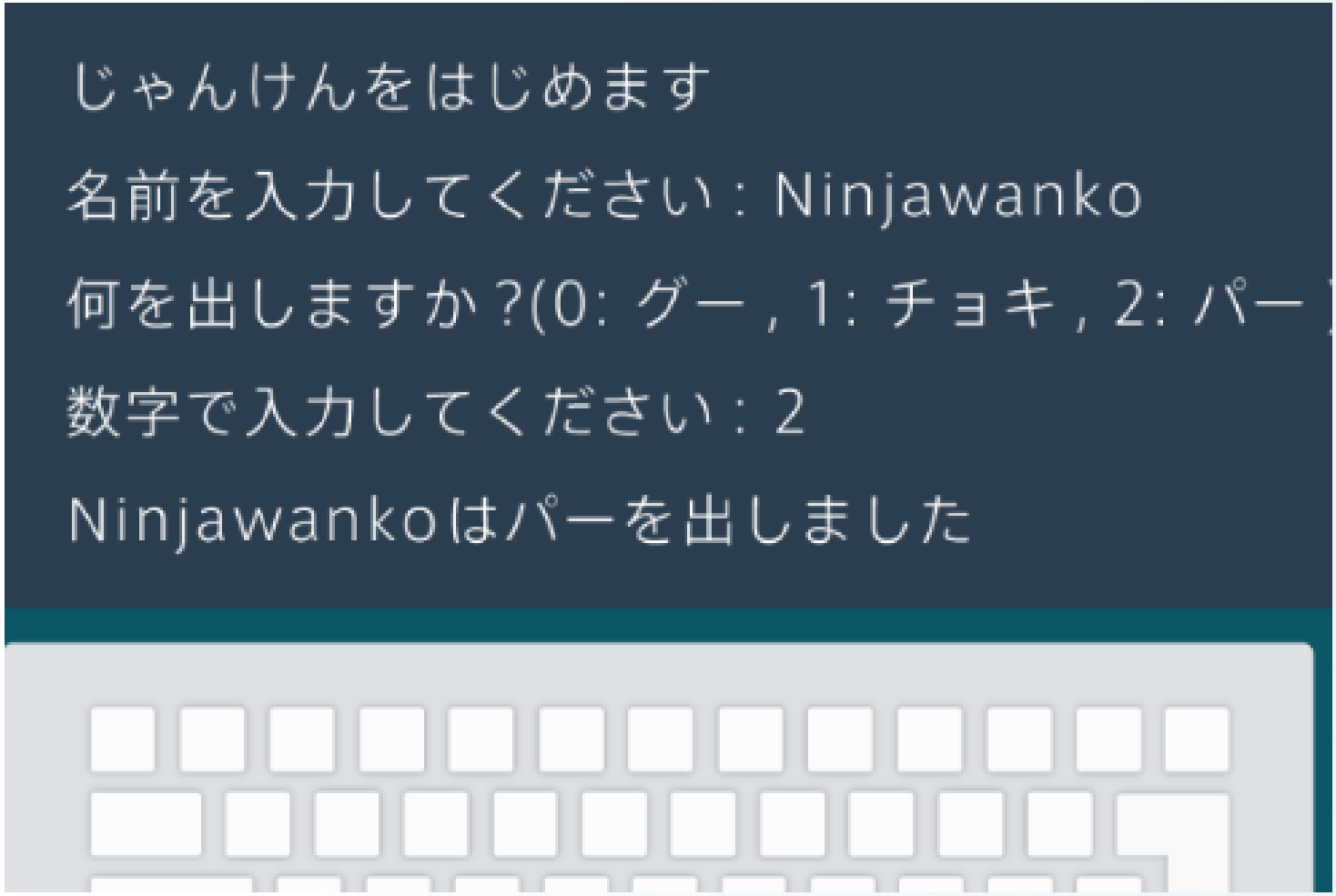
練習

【答え】

```
# 仮引数nameの初期値を設定してください
def print_hand(hand, name = 'ゲスト'):
    print(name + 'は' + hand + 'を出しました')

# 引数に文字列グーのみを入れてください
print_hand('グー')
```

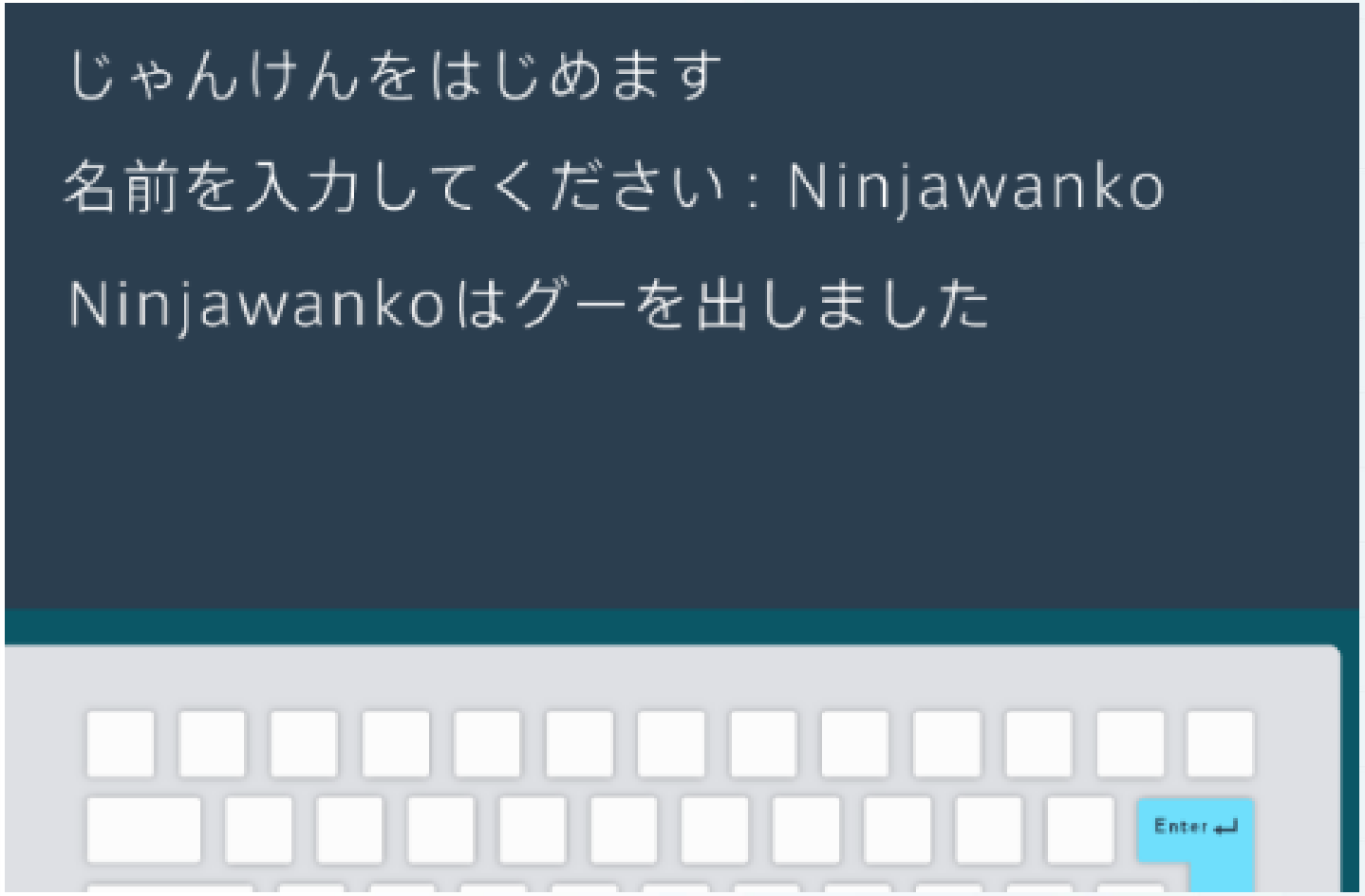
じゃんけんをしよう

今までに作った関数を使って、図のように入力に応じた処理ができるようにする。



名前を受け取ろう

ここでは名前を受け取って表示できるようにする。



練習

【問題】

```
def print_hand(hand, name='ゲスト'):
    print(name + 'は' + hand + 'を出しました')

print('じゃんけんをはじめます')

# inputを用いて入力を受け取り、変数player_nameに代入してください

# 変数player_nameの値によって関数print_handの呼び出し方を変更してください
print_hand('グー')
```

練習

【答え】

```
def print_hand(hand, name='ゲスト'):
    print(name + 'は' + hand + 'を出しました')

print('じゃんけんをはじめます')

# inputを用いて入力を受け取り、変数player_nameに代入してください
player_name = input('名前を入力してください：')

# 変数player_nameの値によって関数print_handの呼び出し方を変更してください
if player_name == ' ':
    print_hand('グー')
else:
    print_hand('グー', player_name)
```

手を選べるようにしよう

グー、チョキ、パーを要素とするリストを用意し、それぞれ要素のインデックス番号に対応する数字を入力することで、どの手を出すか選べるようにする。インデックス番号は、関数に渡す仮引数（hand）で受け取る。

```
def print_hand(hand):
    hands = ['グー', 'チョキ', 'パー']
              0       1       2
                インデックス番号は入力した数字と対応している

    print(hands[hand] + 'を出しました')
          インデックス番号を用いて
          リストの要素(= じゃんけんの手)を取り出す
```

手を選べるようにしよう

入力した数字でプレーヤーの手を判別

0を入力したとき

→ インデックス番号0に
対応する要素＝「グー」

1を入力したとき

→ インデックス番号1に
対応する要素＝「チョキ」

2を入力したとき

→ インデックス番号2に
対応する要素＝「パー」

練習

【問題】

```
def print_hand(hand, name='ゲスト'):
    # 変数handsに、複数の文字列を要素に持つリストを代入してください
    
    # リストhandsを用いて、選択した手が出力されるように書き換えましょう
    print(name + 'は' + hand + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
# 「何を出しますか？（0: グー, 1: チョキ, 2: パー）」と出力してください

# inputを用いて入力を受け取り、数値に型変換してから変数player_handに代入してください

if player_name == ' ' :
    # 第1引数を変数player_handに書き換えてください
    print_hand('グー' )
else:
    # 第1引数を変数player_handに書き換えてください
    print_hand('グー', player_name)
```

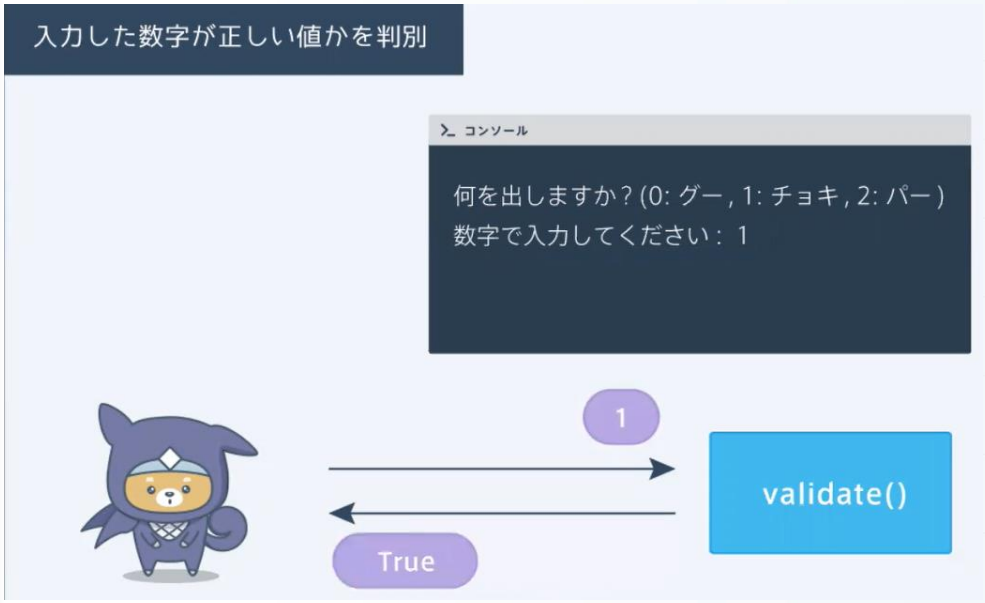
練習

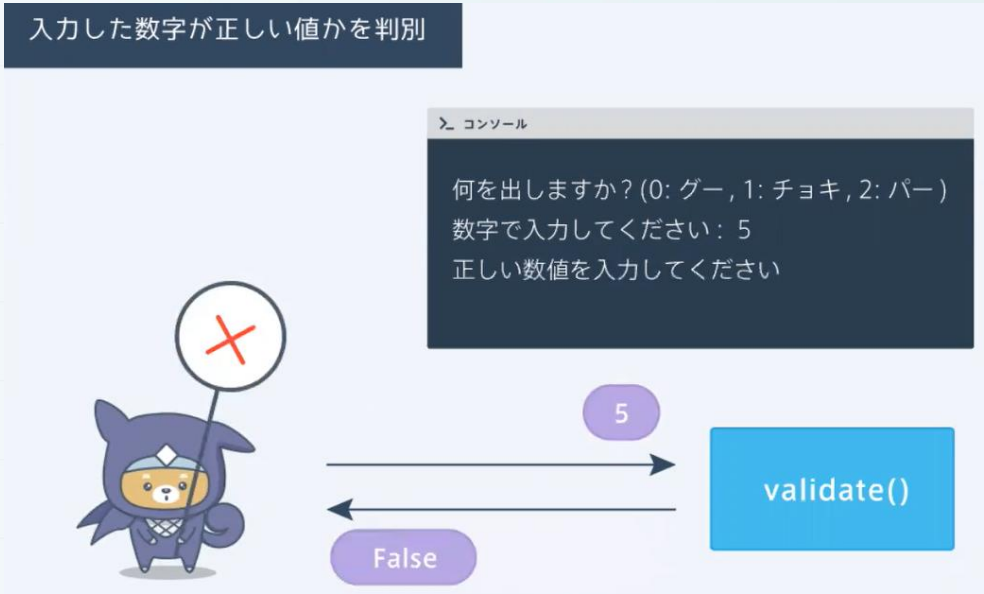
【答え】

```
def print_hand(hand, name='ゲスト'):
    # 変数handsに、複数の文字列を要素に持つリストを代入してください
    hands = ['グー', 'チョキ', 'パー']

    # リストhandsを用いて、選択した手が出力されるように書き換えましょう
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
# 「何を出しますか？（0: グー, 1: チョキ, 2: パー）」と出力してください
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')

# inputを用いて入力を受け取り、数値に型変換してから変数player_handに代入してください
player_hand = int(input('数字で入力してください：'))

if player_name == ' ':
    # 第1引数を変数player_handに書き換えてください
    print_hand(player_hand)
else:
    # 第1引数を変数player_handに書き換えてください
    print_hand(player_hand, player_name)
```

入力値が正しいか判別しよう

入力した値が0,1,2以外の数字だった場合はエラーが起きてしまう。そこで入力された値が正しい数値かどうか判別する関数を作る。関数の結果によって処理を変えたいので戻り値というものを学習する。





戻り値とは

関数の処理結果を関数の呼び出し元で使いたい場合、戻り値を使う。2つの引数を受け取って足し算を行い、結果を返す関数addを例に考えてみよう。関数addは「3」と「7」を受け取ると、処理結果の「10」を呼び出し元に返す。この「10」にあたるのが戻り値という。

戻り値のイメージ

戻り値のある関数

戻り値を呼び出し元に返すには、関数の中で「return」を使う。「return 戻り値」と書くことで戻り値を呼び出し元に返すことができる。

戻り値のある関数

```
def 関数名():
    return 戻り値
```

呼び出し元に値を返す

コード例

```
def add(a,b):
    return a + b
```

aとbを足した値が戻り値として呼び出し元に返される

戻り値を受け取ろう

戻り値がある場合、関数の呼び出し部分がそのまま値に置き換わる。そのため、関数の呼び出し部分を変数に代入することもできる。

```
def add(a,b):

    return a + b
```

「1 + 3」の結果を呼び出し元に返す

```
sum = add(1,3)
```

4

```
print(sum)
```

⇨ 4

練習

【問題】

```
# 関数validateを定義してください
def validate(hand):
    # handの値によって条件分岐してください
```

```
def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

# 関数validateの戻り値がTrueの場合、以下のif~else文が実行されるようにしてください
```

```
# 関数validateの戻り値がFalseの場合「正しい数値を入力してください」と出力してください
```

練習

【答え】

```
# 関数validateを定義してください
def validate(hand):
    # handの値によって条件分岐してください
    if hand < 0 or hand > 2:
        return False
    else:
        return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

# 関数validateの戻り値がTrueの場合、以下のif~else文が実行されるようにしてください
if validate(player_hand):
    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)
# 関数validateの戻り値がFalseの場合「正しい数値を入力してください」と出力してください
else:
    print('正しい数値を入力してください')
```

もし、関数validateがTrue(0,1,2)なら中の処理に進む。Falseなら外側の処理に進む

returnの性質

returnは戻り値を呼び出し元に返すだけでなく、関数内の処理を終了させる性質も持っている。そのため、return以降の関数処理が実行されることはない。

```
def add(a,b):

    return a + b
    print(‘ こんにちは’ )
```

return以降の処理は実行されない

```
sum = add(1,3)
print(sum)
```

4

「こんにちは」は表示されない

複数のreturn

条件分岐と組み合わせると複数のreturnを用いることができる。nameの値が「ゲスト」なので、「名前を教えてください」という文字列がreturnされた時点で処理を終了している。

```
def hello(name = ' ゲスト' ):
    if name ==' ゲスト' :
        return ' 名前を教えてください'
    return name + ' さん、ようこそ！'
print(hello())
```

```
名前を教えてください
>>>
```

練習

【問題】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    # elseを消してインデントを直してください
    else:
        return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if validate(player_hand):
    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)
else:
    print('正しい数値を入力してください')
```

練習

【答え】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    # elseを消してインデントを直してください
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if validate(player_hand):
    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)
else:
    print('正しい数値を入力してください')
```

じゃんけんゲームを作ろう

ここからはじゃんけんゲームを完成させていく。
コンピュータとじゃんけんし、勝敗を判定できるようにする。

```
じゃんけんをはじめます
名前を入力してください：Ninjawanko
何を出しますか？(0: グー , 1: チョキ , 2: パー )
数字で入力してください：1
Ninjawankoはチョキを出しました
コンピューターはグーを出しました
結果は負けでした
```


```
じゃんけんをはじめます
名前を入力してください：Ninjawanko
何を出しますか？(0: グー , 1: チョキ , 2: パー )
数字で入力してください：2
Ninjawankoはチョキを出しました
コンピューターはチョキを出しました
結果は引き分けでした
```


```
じゃんけんをはじめます
名前を入力してください：Ninjawanko
何を出しますか？(0: グー , 1: チョキ , 2: パー )
数字で入力してください：2
Ninjawankoはチョキを出しました
コンピューターはパーを出しました
結果は勝ちでした
```

練習

【問題】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))
```

次のページに続く

練習

```
if validate(player_hand):
    # 変数computer_handに数値1を代入してください


    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)

    # 第1引数をcomputer_hand、第2引数を文字列「コンピューター」
      として関数print_handを呼び出してください


else:
    print('正しい数値を入力してください')
```

練習

【答え】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))
```

次のページに続く

練習

```
if validate(player_hand):
    # 変数computer_handに数値1を代入してください
    computer_hand = 1

    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)

    # 第1引数をcomputer_hand、第2引数を文字列「コンピューター」
      として関数print_handを呼び出してください

    print_hand(computer_hand, 'コンピューター')

else:
    print('正しい数値を入力してください')
```

じゃんけんの結果を判定しよう

じゃんけんの結果を判定する関数を作ろう。

練習

【問題】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

# 関数judgeを定義してください
def judge(player, computer):
    # playerとcomputerの比較結果によって条件を分岐してください
```

次のページに続く

練習

```
print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if validate(player_hand):
    computer_hand = 1

    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)

    print_hand(computer_hand, 'コンピューター')

    # 変数resultに関数judgeの戻り値を代入してください


    # 変数resultを出力してください


else:
    print('正しい数値を入力してください')
```

練習

【答え】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

# 関数judgeを定義してください
def judge(player, computer):
    # playerとcomputerの比較結果によって条件を分岐してください
    if player == computer:
        return '引き分け'
    elif player == 0 and computer == 1:
        return '勝ち'
    elif player == 1 and computer == 2:
        return '勝ち'
    elif player == 2 and computer == 0:
        return '勝ち'
    else:
        return '負け'
```

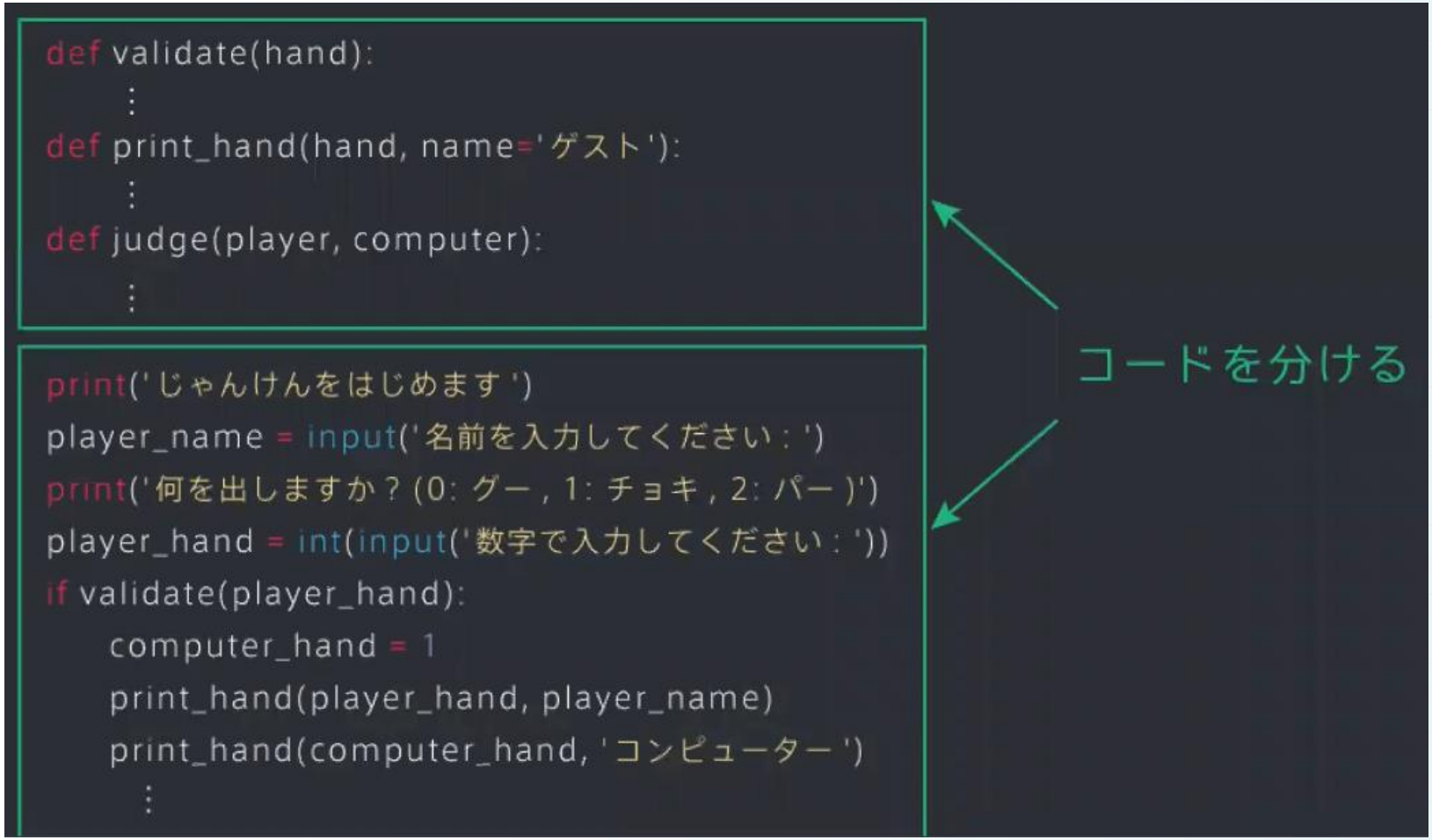
次のページに続く

練習

```
print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if validate(player_hand):
    computer_hand = 1

    if player_name == '':
        print_hand(player_hand)
    else:
        print_hand(player_hand, player_name)

    print_hand(computer_hand, 'コンピューター')

    # 変数resultに関数judgeの戻り値を代入してください
    result = judge(player_hand, computer_hand)
    # 変数resultを出力してください
    print('結果は' + result + 'でした')
else:
    print('正しい数値を入力してください')
```

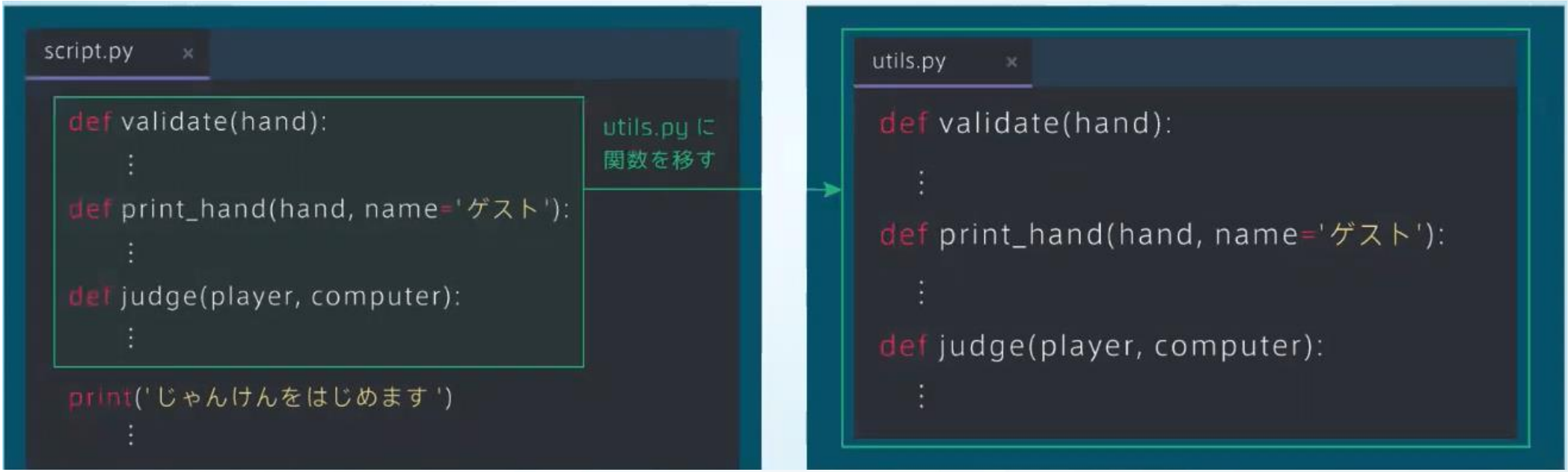
コードを分けよう

コードが増えて長くなると、コードが読みにくくなったり、何をしているかが分かりづらくなってくる。また、予想していないようなバグを引き起こしやすくなる。そこでコードを分ける方法を学習しよう。



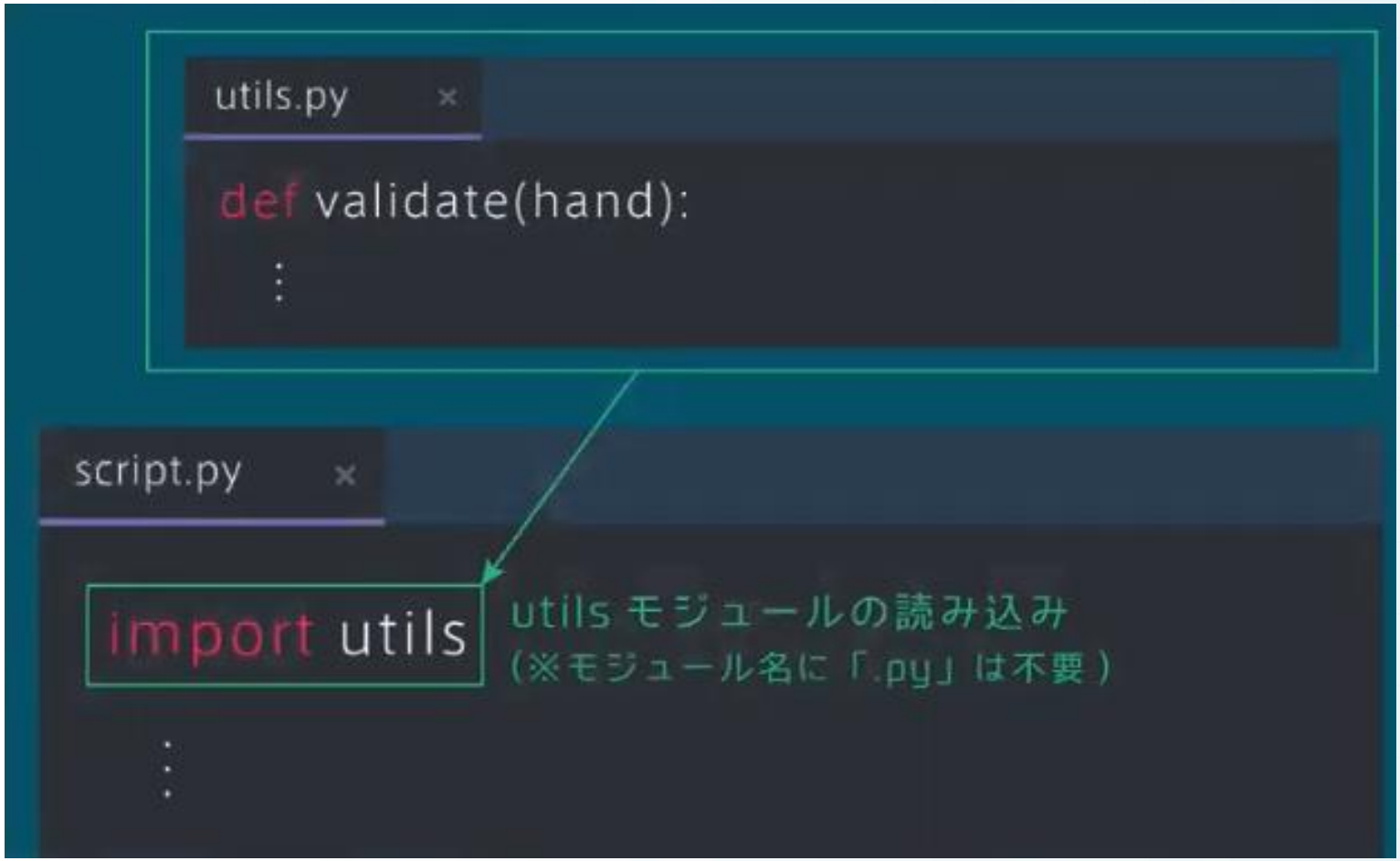
モジュール

モジュールとはPythonのコードが書かれたファイルのことである。別ファイルをモジュールとして読み込むことでそこに書かれたコードを利用することができる。



モジュールを使う

importを使うことでモジュールを読み込むことができる。
モジュールを読み込んで使いたいファイルに「import モジュール名」と書くことで読み込むことができる。モジュール名はファイル名から拡張子(.py)を取り除いたもの。



モジュールの使い方

読み込んだモジュールを使ってみよう。「モジュール名.関数名」と書くことでモジュールの関数を実行することができる。引数がある場合、関数を使用するときと同様に()の中に書く。

script.py

```
import utils
  ⋮
if utils.validate(player_hand):
    computer = 1
    utils.print_hand(player_hand, player_name)
      ⋮
```

ドットでつなぐ

「モジュール名.関数名」で
そのモジュールの関数を実行できる

ドットでつなぐ

utils.py

```
def validate(hand):
    ⋮
def print_hand(hand, name='ゲスト'):
    ⋮
def judge(player, computer):
    ⋮
```

練習

【問題：script.py 】

```
# 3つの関数のコードをutils.pyに移してください（こちらのコードは消してください）
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

def judge(player, computer):
    if player == computer:
        return '引き分け'
    elif player == 0 and computer == 1:
        return '勝ち'
    elif player == 1 and computer == 2:
        return '勝ち'
    elif player == 2 and computer == 0:
        return '勝ち'
    else:
        return '負け'
```

次のページに続く

練習

```
# utils.pyをモジュールとして読み込んでください


print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

# utilsモジュール内の関数validateを呼び出してください

    computer_hand = 1

    if player_name == '':
        # utilsモジュール内の関数print_handを呼び出してください

    else:
        # utilsモジュール内の関数print_handを呼び出してください


    # utilsモジュール内の関数print_handを呼び出してください
```

練習

【問題：utils.py 】

3つの関数のコードを貼り付けてください

練習

【答え：script.py 】

```
# 3つの関数のコードをutils.pyに移してください（こちらのコードは消してください）

# utils.pyをモジュールとして読み込んでください
import utils

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

# utilsモジュール内の関数validateを呼び出してください
if utils.validate(player_hand):
    computer_hand = 1

    if player_name == '':
        # utilsモジュール内の関数print_handを呼び出してください
        utils.print_hand(player_hand)
    else:
        # utilsモジュール内の関数print_handを呼び出してください
        utils.print_hand(player_hand, player_name)

    # utilsモジュール内の関数print_handを呼び出してください
    utils.print_hand(computer_hand, 'コンピュータ')
```

次のページに続く

練習

```
    # utilsモジュール内の関数judgeを呼び出してください
    result = utils.judge(player_hand, computer_hand)
    print('結果は' + result + 'でした')
else:
    print('正しい数値を入力してください')
```

練習

【答え：utils.py】

```
# 3つの関数のコードを貼り付けてください
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

def judge(player, computer):
    if player == computer:
        return '引き分け'
    elif player == 0 and computer == 1:
        return '勝ち'
    elif player == 1 and computer == 2:
        return '勝ち'
    elif player == 2 and computer == 0:
        return '勝ち'
    else:
        return '負け'
```

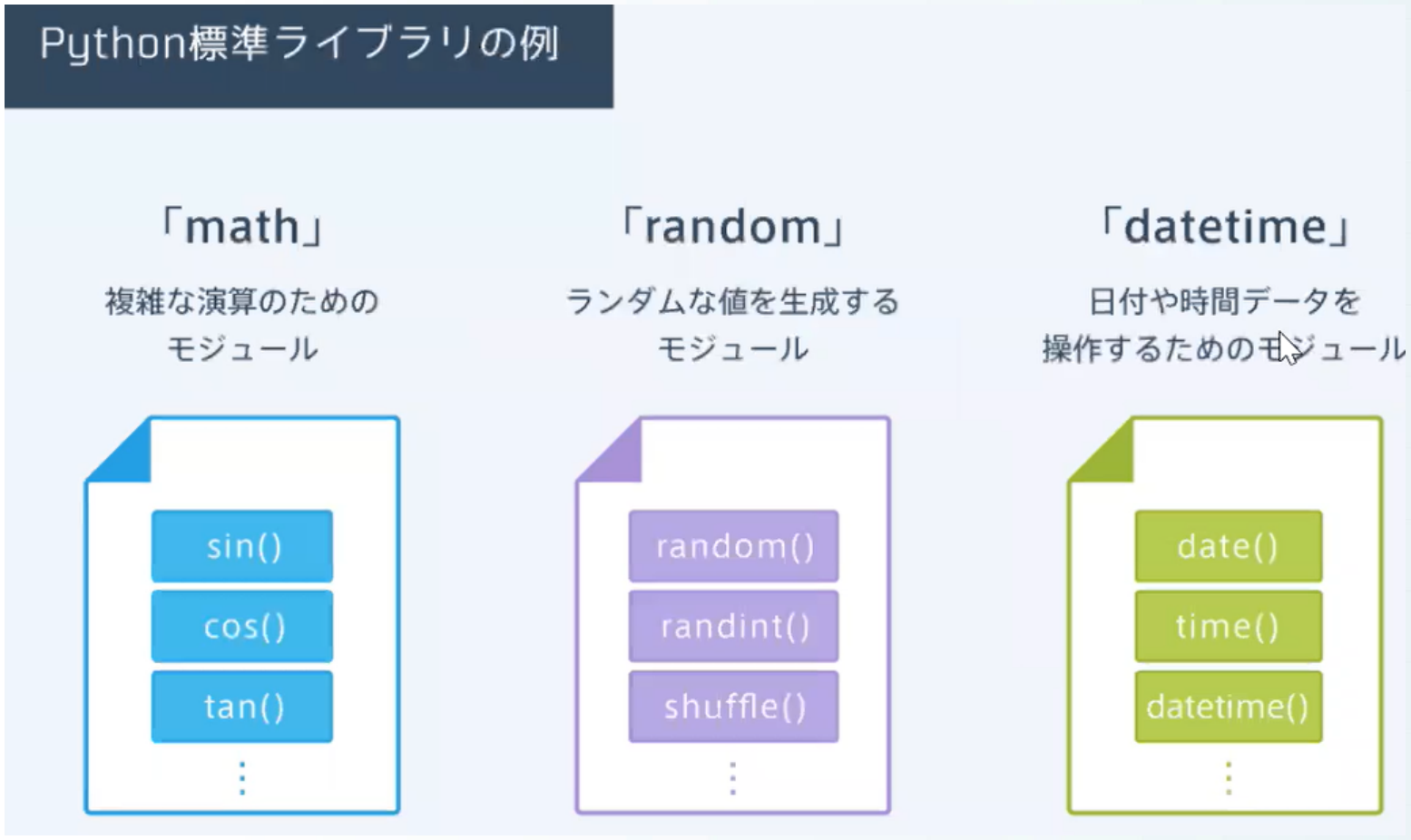
じゃんけんゲームを完成させよう

コンピュータの出す手をランダムにして、じゃんけんゲームを完成させる。

computerの手をランダムにする

ライブラリ

Pythonには便利なモジュールがたくさん用意されている。これらのあらかじめ用意されているモジュールは標準ライブラリと呼ばれ、importを用いて読み込むことで便利な関数を利用できるようになる。



randomモジュール

ここではrandomモジュールに用意されている関数randintを用いて、コンピュータの出す手がランダムになるようにしよう。
random.randint(x,y)と書くことで、xからyまでの整数をランダムに取得することができる。

```
import utils
import random
random モジュールの読み込み
    ⋮
if utils.validate(player_hand):
    computer_hand = random.randint(0, 2)
                    randint 関数を用いて、
                    0～2 の数値をランダムに取得
        ⋮
```

練習

【問題：script.py 】

```
import utils
# randomモジュールを読み込んでください

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if utils.validate(player_hand):
    # randintを用いて0から2までの数値を取得し、変数computer_handに代入してください

    if player_name == '':
        utils.print_hand(player_hand)
    else:
        utils.print_hand(player_hand, player_name)

    utils.print_hand(computer_hand, 'コンピューター')

    result = utils.judge(player_hand, computer_hand)
    print('結果は' + result + 'でした')
else:
    print('正しい数値を入力してください')
```

練習

【問題：utils.py 】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

def judge(player, computer):
    if player == computer:
        return '引き分け'
    elif player == 0 and computer == 1:
        return '勝ち'
    elif player == 1 and computer == 2:
        return '勝ち'
    elif player == 2 and computer == 0:
        return '勝ち'
    else:
        return '負け'
```

練習

【答え：script.py 】

```
import utils
# randomモジュールを読み込んでください
import random

print('じゃんけんをはじめます')
player_name = input('名前を入力してください：')
print('何を出しますか？（0: グー, 1: チョキ, 2: パー）')
player_hand = int(input('数字で入力してください：'))

if utils.validate(player_hand):
    # randintを用いて0から2までの数値を取得し、変数computer_handに代入してください
    computer_hand = random.randint(0, 2)

    if player_name == '':
        utils.print_hand(player_hand)
    else:
        utils.print_hand(player_hand, player_name)

    utils.print_hand(computer_hand, 'コンピューター')

    result = utils.judge(player_hand, computer_hand)
    print('結果は' + result + 'でした')
else:
    print('正しい数値を入力してください')
```

練習

【答え：utils.py 】

```
def validate(hand):
    if hand < 0 or hand > 2:
        return False
    return True

def print_hand(hand, name='ゲスト'):
    hands = ['グー', 'チョキ', 'パー']
    print(name + 'は' + hands[hand] + 'を出しました')

def judge(player, computer):
    if player == computer:
        return '引き分け'
    elif player == 0 and computer == 1:
        return '勝ち'
    elif player == 1 and computer == 2:
        return '勝ち'
    elif player == 2 and computer == 0:
        return '勝ち'
    else:
        return '負け'
```

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。
例えば、データの型という概念はscratchにはあっただろうか？

メモ

プログラミング教室のテクノロ

なまえ：