**## 独学プログラマー**

**## 13章 オブジェクト指向プログラミングの4大要素 P.169**

#▼▼ チャレンジ1 ▼▼

class Rectangle:

def \_\_init\_\_(self, width, length):

self.width = width

self.length = length

self.shape = "Rectangle"

# 外周の長さを返す

def calculate\_perimeter(self):

return (self.width \* 2 + self.length \* 2)

class Square:

def \_\_init\_\_(self, width):

self.width = width

self.shape = "Square"

# 外周の長さを返す

def calculate\_perimeter(self):

return (self.width \* 4)

# エントリポイント

rec = Rectangle(10,30)

squ = Square(15)

shapes = [rec, squ]

for a\_shape in shapes:

print("{}の外周の長さは...".format(a\_shape.shape))

print(a\_shape.calculate\_perimeter())

#▲▲ チャレンジ1 ▲▲

#▼▼ チャレンジ2 ▼▼

class Square:

def \_\_init\_\_(self, width):

self.width = width

self.shape = "Square"

# 外周の長さを返す

def calculate\_perimeter(self):

return (self.width \* 4)

# 横幅を変更する

def change\_size(self, delta):

self.width += delta

# エントリポイント

squ = Square(15)

print("初期状態の{}の外周の長さは...".format(squ.shape))

print(squ.calculate\_perimeter())

try:

delta = int(input("幅変更\n"))

squ.change\_size(delta)

except ValueError as e:

print(e.args)

print("入力した値が不正のため幅変更は行いません。\n")

print("変更後の{}の外周の長さは...".format(squ.shape))

print(squ.calculate\_perimeter())

#▲▲ チャレンジ2 ▲▲

#▼▼ チャレンジ3 ▼▼

class Shape:

def \_\_init\_\_(self):

pass

def what\_am\_i(self):

print("I am a shape.")

class Rectangle(Shape):

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

self.shape = "Rectangle"

# 外周の長さを返す

def calculate\_perimeter(self):

return (self.width \* 2 + self.height \* 2)

class Square(Shape):

def \_\_init\_\_(self, width):

self.width = width

self.shape = "Square"

# 外周の長さを返す

def calculate\_perimeter(self):

return (self.width \* 4)

# 横幅を変更する

def change\_size(self, delta):

self.width += delta

# エントリポイント

rec = Rectangle(10,30)

squ = Square(15)

rec.what\_am\_i()

squ.what\_am\_i()

#▲▲ チャレンジ3 ▲▲

#▼▼ チャレンジ4 ▼▼

class Horse:

def \_\_init\_\_(self, name, rider):

self.name = name

self.rider = rider

class Rider:

def \_\_init\_\_(self, name):

self.name = name

# エントリポイント

the\_rider = Rider("Satoru Suzuki")

hor = Horse("Serturu Naliya", the\_rider)

print("{0}(馬)の騎手は{1}です。".format(hor.name, hor.rider.name))

#▲▲ チャレンジ4 ▲▲

# 解答URL：<http://tinyurl.com/hz9qdh3>

**## 独学プログラマー**

**## 10章 知識を1つにまとめる P.134～139**

def main():

isPlayGame = True

print("「Hangmanゲーム」 START")

print("-----------------------")

while True:

strGameWord = input("[Player1]\n1～10文字以内の英単語を入力してください。\n 例)cat\n")

iGameWordLen = len(strGameWord)

if (iGameWordLen < 1) or (iGameWordLen > 10):

print(msg("文字数が不正です。"))

continue

hangman(strGameWord)

msg = "もう一度やる？？ y/n\n"

isYes = input(msg)

if isYes == "y":

continue

else:

print("また遊んでね～～")

break

def hangman(strPlayer1Word):

isWin = False

iMissCount = 0

iSuccCount = 0

hangList = [

" A ",

" A AA ",

" A AA ",

" AAAAAAAA ",

" A AA"

]

strTempWord = list("\_" \* len(strPlayer1Word))

player1WordList = list(strPlayer1Word)

while iMissCount < len(hangList):

msg = "\n[Player2]\n1文字予想しよう。\n 例)c\n"

strPlayer2Word = input(msg)

print("----------")

if strPlayer2Word in player1WordList:

print("\n入力文字が当たった！")

iSuccCount = iSuccCount + 1

iIndexCha = player1WordList.index(strPlayer2Word)

strTempWord[iIndexCha] = strPlayer2Word

player1WordList[iIndexCha] = "@"

else:

iMissCount = iMissCount + 1

print("予想英単語：" + "".join(strTempWord))

print("\n絵：")

print("\n".join(hangList[0:iMissCount]))

if "\_" not in strTempWord:

isWin = True

break

if not isWin:

print("絵(A)が完成しました!")

print("Player1の勝利です! ")

print("正解は...{}でした".format(strPlayer1Word))

else:

print("英単語を当てました!")

print("Player2の勝利です!")

print("...")

print("「Hangmanゲーム」 FINISH")

def msg(msgNumber):

msgList = {

"1":"「Hangmanゲーム」 START",

"2":"1～10文字以内の英単語を入力してください。\n 例)cat\n",

"3":"文字数が不正です。",

"4":"Do you want to quit ?? y/n (y:yes, n:no)",

"wait":"Please wait...\n"

}

return msgList[msgNumber]

# エントリポイント

main()

# 解答URL：<http://tinyurl.com/j7rb8or>

**## 独学プログラマー**

**## 15章 知識を1つにまとめる P.178～187**

from random import shuffle # カードシャッフル用

from operator import attrgetter # 順位ソート用

class WarGame:

def \_\_init\_\_(self):

self.roundCount = 0 # ラウンドカウンタ初期化

self.c\_deck = Deck() # デッククラスのインスタンス作成

self.c\_print =Print() # 出力クラスのインスタンス作成

self.c\_deck.mk\_random\_deck() # ランダムデック作成

# ルール説明

def rule\_description(self):

print("""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ルール説明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 各プレイヤーは、トランプの山札から一枚カードを引く。

2. 引いたカードを場に出す。

3. 一番大きい数字を出したプレイヤーの勝利。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

""")

# エントリープレイヤー決定

def entry\_player(self):

playerCount = 0 # プレイヤーカウンタ

self.playerList = [] # プレイヤーリスト

while True:

playerCount += 1

msg = "プレイヤー{0}の名前を入力してください\n"

name = input(msg.format(playerCount)) # プレイヤー名を入力

self.playerList.append(Player(name)) # プレイヤーをリストに追加

self.playerList[playerCount - 1].num = playerCount # プレイヤー番号を格納

print("")

if playerCount > 1:

print("※ 他にエントリーする人はいますか？")

if quit() == "q":

break

print("以下プレイヤーがエントリーされています")

self.c\_print.\_break()

for i, a\_player in enumerate(self.playerList):

print("プレイヤー{}：".format(i + 1) + a\_player.name)

self.c\_print.\_break()

# カードをデックから引く

def draw\_card(self):

self.roundCount += 1

msg = "\*\*\*\*\*\* ラウンド{} \*\*\*\*\*\*"

input(msg.format(self.roundCount))

for a\_player in self.playerList:

msg = "\n{}さんが、カードを引く番です\n[Enter...]"

input(msg.format(a\_player.name))

a\_player.card = self.c\_deck.draw\_card()

# カードオープン

def open\_card(self):

msg = "\n\*\* カードをオープン \*\*\n[Enter...]"

input(msg)

self.c\_print.\_break()

for a\_player in self.playerList:

msg = "{}さん：{}"

print(msg.format(

a\_player.name, a\_player.card))

self.c\_print.\_break()

msg = "[Enter...]"

input(msg)

# 勝負

def match\_card(self):

tmpCard = None

tmpName = ""

playerNum = 0 # プレイヤーリストのインデックスを格納

# 一番強いカードを持つプレイヤーを見つける

for i, a\_player in enumerate(self.playerList):

if tmpCard is None:

tmpCard = a\_player.card

playerNum = i

else:

if tmpCard < a\_player.card:

tmpCard = a\_player.card

playerNum = i

msg = "\nラウンド{}の勝者は「{}」を引いた\nプレイヤー{}の「{}」さんです！"

print(msg.format(

self.roundCount, self.playerList[playerNum].card,

self.playerList[playerNum].num, self.playerList[playerNum].name))

self.playerList[playerNum].wins += 1 # 勝利数をカウント

# 順位表示 (wins, num, name)

def disp\_rank(self):

rank = 1 # 表示順位を格納

tempWins = 0 # 比較する勝利数を格納

# 勝利数を基にプレイヤーを降順でソート

rankList = sorted(

self.playerList, key = attrgetter('wins'), reverse = True)

print("\n順位は...")

for a\_player in rankList:

# 同順位を考慮して表示順位を算出

if tempWins == 0:

tempWins = a\_player.wins

if tempWins > a\_player.wins:

tempWins = a\_player.wins

rank += 1

msg = "{}位：{}勝 プレイヤー{} {}さん"

print(msg.format(

rank, a\_player.wins, a\_player.num, a\_player.name))

# プレイゲーム

def play\_game(self):

print("War Game を始めます。")

self.rule\_description()

self.entry\_player()

while len(self.c\_deck.deck) >= len(self.playerList):

if quit() == "q":

break

self.draw\_card()

self.open\_card()

self.match\_card()

if len(self.c\_deck.deck) < len(self.playerList):

print("\nデックにカードが残っていないため、優勝者を決定します。")

self.who\_is\_winner()

self.disp\_rank()

if quit() != "q":

self.roundCount = 0 # ラウンドカウンタ初期化

self.c\_deck.mk\_random\_deck() # ランダムデック作成

self.play\_game()

else:

print("\nまた遊んでね～～")

# 優勝者決定

def who\_is\_winner(self):

tempWins = 0 # ラウンド勝利数を格納

winnnersList = [] # 同率優勝者を格納

tmpName = "" # 仮優勝者名を格納

for a\_player in self.playerList:

if tempWins < a\_player.wins:

tempWins = a\_player.wins

winnnersList = [] # 優勝者リストを初期化

winnnersList.append(a\_player) # 優勝者リストにプレイヤーを追加

elif tempWins == a\_player.wins:

winnnersList.append(a\_player) # 優勝者リストに同率プレイヤーを追加

print("\n今回の優勝者は...")

for a\_winner in winnnersList:

msg = " {}勝した プレイヤー{}の「{}」さん"

print(msg.format(

a\_winner.wins, a\_winner.num, a\_winner.name))

print("です！ おめでとうございます！")

class Player:

def \_\_init\_\_(self, name):

self.name = name # プレイヤー名を格納

self.wins = 0 # 勝利数を格納

self.card = None # Cardオブジェクトのため、Noneで初期化

self.num = 0 # プレイヤー番号を格納

class Deck:

def \_\_init\_\_(self):

self.deck = [] # カードリストを宣言

def mk\_random\_deck(self):

for i in range(4):

for j in range(2, 15):

self.deck.append(Card(i, j))

shuffle(self.deck)

def draw\_card(self):

if len(self.deck) == 0:

return

else:

return self.deck.pop()

class Card:

markList = ["clubs", "diamonds", "hearts", "spades"]

# Noneを0,1に使うことで、引数値に対応

valueList = [None, None,

"2", "3", "4", "5", "6", "7", "8", "9", "10",

"Jack", "Queen", "King", "Ace"]

def \_\_init\_\_(self, mark, value):

self.mark = mark # トランプのマークインデックスを格納

self.value = value # トランプの値インデックスを格納

# 特殊メソッド <演算子

def \_\_lt\_\_(self, otherCard):

if self.value < otherCard.value:

return True

elif self.value == otherCard.value:

return self.mark < otherCard.mark

else:

return False

# 特殊メソッド >演算子

def \_\_gt\_\_(self, otherCard):

if self.value > otherCard.value:

return True

elif self.value == otherCard.value:

return self.mark > otherCard.mark

else:

return False

# 特殊メソッド print

def \_\_repr\_\_(self):

newPrint = self.valueList[self.value] + \

" of " + self.markList[self.mark]

return newPrint

class Print:

def \_\_init\_\_(self):

self.symbolNum = 25

def \_break(self):

print("\*" \* self.symbolNum)

def quit():

msg = "'q'で終了、それ以外のキーで続行..."

return input(msg)

# エントリポイント

game = WarGame()

game.play\_game()

"""

【補足説明】

変数名について

mk：make

rm：remove

▼▼出力結果▼▼

======== RESTART: C:/workspace/Python-3.8.2-Shell/Book\_01/15\_War\_Game.py =======

War Game を始めます。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ルール説明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 各プレイヤーは、トランプの山札から一枚カードを引く。

2. 引いたカードを場に出す。

3. 一番大きい数字を出したプレイヤーの勝利。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

プレイヤー1の名前を入力してください

suzuki

プレイヤー2の名前を入力してください

satoru

※ 他にエントリーする人はいますか？

'q'で終了、それ以外のキーで続行...

プレイヤー3の名前を入力してください

Strong

※ 他にエントリーする人はいますか？

'q'で終了、それ以外のキーで続行...

プレイヤー4の名前を入力してください

Weak

※ 他にエントリーする人はいますか？

'q'で終了、それ以外のキーで続行...q

以下プレイヤーがエントリーされています

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

プレイヤー1：suzuki

プレイヤー2：satoru

プレイヤー3：Strong

プレイヤー4：Weak

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'q'で終了、それ以外のキーで続行...

\*\*\*\*\*\* ラウンド1 \*\*\*\*\*\*

suzukiさんが、カードを引く番です

[Enter...]

satoruさんが、カードを引く番です

[Enter...]

Strongさんが、カードを引く番です

[Enter...]

Weakさんが、カードを引く番です

[Enter...]

\*\* カードをオープン \*\*

[Enter...]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

suzukiさん：7 of spades

satoruさん：2 of diamonds

Strongさん：10 of diamonds

Weakさん：4 of diamonds

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Enter...]

ラウンド1の勝者は「10 of diamonds」を引いた

プレイヤー3の「Strong」さんです！

'q'で終了、それ以外のキーで続行...

～～～～

～～～～

ラウンド4の勝者は「10 of spades」を引いた

プレイヤー2の「satoru」さんです！

'q'で終了、それ以外のキーで続行...q

今回の優勝者は...

2勝した プレイヤー2の「satoru」さん

です！ おめでとうございます！

順位は...

1位：2勝 プレイヤー2 satoruさん

2位：1勝 プレイヤー1 suzukiさん

2位：1勝 プレイヤー3 Strongさん

3位：0勝 プレイヤー4 Weakさん

'q'で終了、それ以外のキーで続行...

また遊んでね～～

▲▲出力結果▲▲

"""

# 解答：本書 P.178～187 参照

**## 独学プログラマー**

**## 20章 知識を1つにまとめる P.238～245**

# Thanks so much for reading my book. Feel free to contact me at cory[at]theselftaughtprogrammer.io.

# ▼▼ チャレンジ 1 ▼▼

import urllib.request

from bs4 import BeautifulSoup

import os

from datetime import datetime

import csv

class Scraper:

def \_\_init\_\_(self, site, filename):

self.site = site

self.wd = os.getcwd() # 本実行fileのディレクトリを取得

self.filename = self.wd + "/" + filename + ".csv"

def scrape(self):

r = urllib.request.urlopen(self.site)

html = r.read()

parser = "html.parser"

sp = BeautifulSoup(html, parser)

for tag in sp.find\_all("a"):

url = tag.get("href")

if url is None:

continue

if "https://www" in url:

print("\n" + url)

self.write\_file(url)

def write\_file(self, writeMsg):

with open(self.filename, 'a', newline ="") as f: # 'a'：追記

self.w = csv.writer(f)

self.w.writerow([writeMsg]) # []で囲むことで、1オブジェクトとしてみなす

# エントリポイント

news = "https://news.google.com/" # GoogleニュースのURL

date = datetime.now() # 現在時刻を取得

strDate = str(date.year) + str(date.month) + str(date.day) + \

str(date.hour) + str(date.minute)

Scraper(news, strDate).scrape()

# ▲▲ チャレンジ 1 ▲▲

"""

出力ファイル名：2020542246.csv

▼▼ 出力結果 ▼▼

https://www.google.com/intl/en/about/products

https://www.google.com/webhp

https://www.youtube.com/?gl=US

https://www.google.com/calendar

https://www.google.com/chrome/?brand=CHZO&utm\_source=google.com&utm\_medium=desktop-app-launcher&utm\_campaign=desktop-app-launcher&utm\_content=chrome-logo&utm\_keyword=CHZO

https://www.google.com/shopping?hl=en&source=og

https://www.google.com/finance

https://www.blogger.com/

https://www.google.com/save

https://www.google.com/intl/en/about/products

https://www.google.com/producer/app/privacy?hl=en-US

https://www.google.com/producer/app/tos?hl=en-US

https://www.google.com/about

▲▲ 出力結果 ▲▲ """

# 解答URL：<http://tinyurl.com/gkv6fuh>

Print済

**## 独学プログラマー**

**## 21章 データ構造 P.251～252**

# ▼▼ スタックについて ▼▼

class Stack:

def \_\_init\_\_(self):

self.items = []

# スタックが空かの確認

def is\_empty(self):

return not self.items

# スタック最に要素を追加

def push(self, item):

self.items.append(item)

# スタック最後の要素を削除し、返す

def pop(self):

return self.items.pop()

# スタック最後の要素を取得

def peek(self):

return self.items[-1]

# スタックサイズを取得

def size(self):

return len(self.items)

# エントリポイント

stack = Stack()

for c in "Hello":

stack.push(c)

reversed\_string = ""

print(" \*\* Before \*\* ")

print("stack.items：" + str(stack.items))

print("reversed\_string：" + reversed\_string)

print(stack.peek())

while stack.size():

reversed\_string += stack.pop()

print("\n \*\* After \*\* ")

print("stack.items：" + str(stack.items))

print("reversed\_string：" + reversed\_string)

# ▲▲ スタックについて ▲▲

""" ▼▼ 出力結果 ▼▼

\*\* Before \*\*

stack.items：['H', 'e', 'l', 'l', 'o']

reversed\_string：

\*\* After \*\*

stack.items：[]

reversed\_string：olleH

▲▲ 出力結果 ▲▲ """

# 解答URL：<https://github.com/calthoff/self_taught/blob/master/python_ex299.py/>

## 独学プログラマー

## 21章 データ構造 P.257

# ▼▼ チャレンジ 1 ▼▼

class Stack:

def \_\_init\_\_(self):

self.items = []

# スタックが空かの確認

def is\_empty(self):

return not self.items

# スタック最に要素を追加

def push(self, item):

self.items.append(item)

# スタック最後の要素を削除し、返す

def pop(self):

return self.items.pop()

# スタック最後の要素を取得

def peek(self):

return self.items[-1]

# スタックサイズを取得

def size(self):

return len(self.items)

# エントリポイント

stack = Stack()

for c in "yesterday":

stack.push(c)

reversed\_string = ""

print(" \*\* Before \*\* ")

print("stack.items：" + str(stack.items))

print("reversed\_string：" + reversed\_string)

while stack.size():

reversed\_string += stack.pop()

print("\n \*\* After \*\* ")

print("stack.items：" + str(stack.items))

print("reversed\_string：" + reversed\_string)

# ▲▲ チャレンジ 1 ▲▲

""" ▼▼ 出力結果 ▼▼

\*\* Before \*\*

stack.items：['y', 'e', 's', 't', 'e', 'r', 'd', 'a', 'y']

reversed\_string：

\*\* After \*\*

stack.items：[]

reversed\_string：yadretsey

▲▲ 出力結果 ▲▲ """

# ▼▼ チャレンジ 2 ▼▼

class Stack:

def \_\_init\_\_(self):

self.items = []

# スタックが空かの確認

def is\_empty(self):

return not self.items

# スタック最に要素を追加

def push(self, item):

self.items.append(item)

# スタック最後の要素を削除し、返す

def pop(self):

return self.items.pop()

# スタック最後の要素を取得

def peek(self):

return self.items[-1]

# スタックサイズを取得

def size(self):

return len(self.items)

# エントリポイント

stack = Stack()

cha2List = [1, 2, 3, 4, 5]

revcha2List = []

for c in cha2List:

stack.push(c)

print(" \*\* Before \*\* ")

print("stack.items：" + str(stack.items))

print("cha2List：" + str(cha2List))

print("revcha2List：" + str(revcha2List))

while stack.size():

revcha2List.append(stack.pop())

print("\n \*\* After \*\* ")

print("stack.items：" + str(stack.items))

print("cha2List：" + str(cha2List)) # リスト

print("revcha2List：" + str(revcha2List)) # 要素逆順の新リスト

# ▲▲ チャレンジ 2 ▲▲

""" ▼▼ 出力結果 ▼▼

\*\* Before \*\*

stack.items：[1, 2, 3, 4, 5]

cha2List：[1, 2, 3, 4, 5]

revcha2List：[]

\*\* After \*\*

stack.items：[]

cha2List：[1, 2, 3, 4, 5]

revcha2List：[5, 4, 3, 2, 1]

▲▲ 出力結果 ▲▲ """

# 解答URL：https://github.com/calthoff/tstp/tree/master/part\_IV/data\_structures/challenges