

[読者になる](#)

gotutiyans blog

2020-04-14

【python3.x】 練習問題55本ノック 【問題のみ】

[python](#)

はじめに

ここではpythonの練習問題を掲載しています。

データを処理する際には、必ず「データの格納」と「データの取り出し」を行うことになるので、その方法を知ることを目的としています。問題は暗算で解けるものや、頭の中で答えが分かるものが多数ありますが、あくまでも、「行いたい処理をpython3という言語ではどのように表現するのか」を確認するものです。

解答の一例は、以下のリンクで公開しています。

gotutiyans.hatenablog.com

一応筆者としては、問題と解答のタブを2つとも開きながら学習することを想定しています。また、解答のページでは、本記事に解答のコードが追加されるだけで、解答のページだけを見ながら取り組むこともできます。

全ての問題には、必ず出力が存在します。出力は `print()` を用いて、画面に表示させることを想定しています。（改行区切り？空白区切り？などの細かな体裁は気にしません。）

(2020/12/4 追記) 演習問題との関係で、45番目の降順ソートの項目を削除し、ラムダ式の項目に書き換えました。

- [はじめに](#)
- [問題](#)
 - [1. 変数](#)
 - [2. swap](#)
 - [3. 四則演算+](#)
 - [4. 余り](#)
 - [5. べき乗](#)
 - [6. if, 比較演算子<, >](#)
 - [7. 比較演算子==, bool](#)

プロフィール



[ごつちゃん \(id:gotutiyans\)](#)

processingや競技プログラミング、自然言語処理など

[読者になる](#) 23

[このブログについて](#)

検索

注目記事

2020-04-14

[【python3.x】 練習問題55本ノック【問題と解答】](#)

2020-09-08

[【sklearn】 LabelEncoderの使い方を丁寧に](#)

2020-09-02

[【Pytorch】 nn.Embeddingの使い方を丁寧に](#)

2020-04-21

[【Pytorch】 ミニバッチ学習に便利なDataSet・DataLoaderの使い方](#)

2020-08-07

[Processingで「時間」を扱う一般的なテクニック](#)

月別アーカイブ

- ▶ [2023 \(2\)](#)
- ▶ [2022 \(5\)](#)
- ▶ [2021 \(16\)](#)
- ▼ [2020 \(33\)](#)
 - [2020 / 12 \(2\)](#)
 - [2020 / 11 \(3\)](#)
 - [2020 / 10 \(6\)](#)
 - [2020 / 9 \(8\)](#)
 - [2020 / 8 \(1\)](#)

- [8. 文字列（ランダムアクセス）](#) [2020 / 7 \(2\)](#)
- [9. 文字列（結合）](#) [2020 / 6 \(1\)](#)
- [10. 文字列（format）](#) [2020 / 5 \(1\)](#)
- [11. 文字列（replace）](#) [2020 / 4 \(5\)](#)
- [12. 文字列（lower）](#) [2020 / 1 \(4\)](#)
- [13. 文字列（upper）](#) [▶ 2019 \(9\)](#)
- [14. 文字列（文字数）](#) [▶ 2018 \(61\)](#)
- [15. 文字列 → 数値](#)
- [16. リスト](#)
- [17. リスト（結合）](#)
- [18. リスト（append）](#)
- [19. リスト（insert）](#)
- [20. リスト（forによる検索）](#)
- [21. リスト（forによる検索, enumerate）](#)
- [22. リスト（len）](#)
- [23. リスト, if（存在確認）](#)
- [24. タプル, リストの負のindex](#)
- [25. 辞書](#)
- [26. 辞書（keys）](#)
- [27. 辞書（values）](#)
- [28. 辞書（items）](#)
- [29. 辞書（キーの存在確認）](#)
- [30. スライス1](#)
- [31. スライス2](#)
- [32. スライス3](#)
- [33. 集合](#)
- [34. 積集合](#)
- [35. 和集合](#)
- [36. 差集合](#)
- [37. 型の確認](#)
- [38. strip](#)
- [39. split](#)
- [40. join](#)
- [41. max](#)
- [42. min](#)
- [43. sum](#)
- [44. 昇順ソート](#)
- [45. ラムダ式](#)
- [46. range\(\)](#)
- [47. 内包表記](#)
- [48. 例外処理](#)
- [49. ビット演算](#)
- [50. モジュールのインポート](#)
- [演習1（forのネスト）](#)
- [演習2（辞書の値ソート）](#)
- [演習3（num2freq）](#)
- [演習4（word2freq）](#)
- [演習5（Jaccard係数）](#)

カテゴリー

- [論文メモ \(8\)](#)
- [GEC \(6\)](#)
- [文法誤り訂正 \(6\)](#)
- [pytorch \(3\)](#)
- [機械学習・深層学習 \(11\)](#)
- [論文紹介 \(1\)](#)
- [python \(13\)](#)
- [ICPC \(21\)](#)
- [ゲーム制作 \(6\)](#)
- [processing \(18\)](#)
- [振り返り \(2\)](#)
- [競技プログラミング全般 \(2\)](#)
- [NAIST \(1\)](#)
- [spaCy \(1\)](#)
- [nltk \(1\)](#)
- [sklearn \(2\)](#)
- [Web開発 \(1\)](#)
- [書評 \(1\)](#)
- [yukicoder \(8\)](#)
- [AtCoder \(10\)](#)
- [AOJ \(17\)](#)
- [CodinGame \(1\)](#)
- [ゲーム攻略 \(1\)](#)
- [openframeworks \(19\)](#)
- [CodeForces \(2\)](#)

問題

1. 変数

x という変数に 2 を代入し、それを3倍した数を出力してください。

期待する出力 : 6

2. swap

変数 a に 100 を代入し、変数 b に 200 を代入します。その後、両者の値を入れ替えて、a に 200 , b に 100 が代入されているようにしてください。出力としては a と b を出力してください。出力形式は問いませんが、print(a,b) とすると、空白区切で出力できます。

期待する出力の一例 : 200 100

3. 四則演算+α

変数 a に 10 を代入、変数 b に 2 を代入し、a と b の和、差、積、商を出力してください。出力形式は問いません。また、商の値が小数か整数かは問いません。

期待する出力の一例 : 12 8 20 5.0

4. 余り

変数 a に 5 を代入、変数 b に 3 を代入し、a を b で割った時の余りを求めてください。

期待する出力 : 2

5. べき乗

変数 a に 5 を代入、変数 b に 10 を代入し、a の b 乗、つまり a^b を出力してください。

期待する出力 : 9765625

6. if, 比較演算子 <, >

変数 a に 5 を代入、変数 b に 10 を代入し、a と b のうち大きい方を出力してください。

期待する出力 : 10

7. 比較演算子 ==, bool

変数 a に 5 を代入し、a が偶数ならTrue、そうでなければFalseを出力してください。(条件式をそのままprintすれば良いです。)

期待する出力 : False

8. 文字列（ランダムアクセス）

変数に、'python' という文字列を代入し、先頭の文字を0番目として2番目の文字を出力してください。

期待する出力 : t

9. 文字列（結合）

'py' と 'thon' という2つの文字列をそれぞれ変数に代入し、結合したものを出力してください。

期待する出力 : python

10. 文字列（format）

変数 a に 5 を、変数 b に 3 を代入し、これらの変数を用いて '5%3=2' という文字列を出力してください。

期待する出力 : 5%2=2

11. 文字列（replace）

変数に文字列 'some1' を代入し、この文字列中の 1 を one に変換してください。

期待する出力 : someone

12. 文字列（lower）

変数に文字列 'This Is A Sentence .' を代入し、この文字列を全て小文字に変換してください。

期待する出力 : this is a sentence .

13. 文字列（upper）

変数に文字列 'This Is A Sentence .' を代入し、この文字列を全て大文字に変換してください。

期待する出力 : THIS IS A SENTENCE .

14. 文字列（文字数）

変数に文字列 'How many characters?' を代入し、この文字列の文字数を出力してください。空白も含めるものとします。

期待する出力 : 20

15. 文字列 → 数値

変数 a に文字列 '34' を代入し、変数 b に文字列 '43' を代入し、これらを数字と見なした時の和を出力してください。

期待する出力 : 77

16. リスト

変数にリスト [1,2,3,4,5] を代入し、(先頭を0番目として)3番目の要素を出力してください。

期待する出力 : 4

17. リスト（結合）

変数 li1 に [1,2,3] のリストを代入、変数 li2 に [4,5] のリストを代入し、2つのリストを結合して出力してください。

期待する出力 : [1,2,3,4,5]

18. リスト (append)

変数にリスト [1,2,3,4,5] を代入し、このリストの末尾に 6, 7 を1つずつ順番に追加してください。その後、最終的なリストを出力してください。

期待する出力 : [1,2,3,4,5,6,7]

19. リスト (insert)

変数にリスト [1,2,3,4,5] を作成し、先頭を0番目としたときに、0番目と1番目の間に 100 を挿入してください。

期待する出力 : [1,100,2,3,4,5]

20. リスト (forによる検索)

変数に [1,2,3,4,5] の5つの要素を持つリストを格納して、偶数の要素だけ出力してください。出力形式は問いません。

期待する出力 : 2 4

21. リスト (forによる検索, enumerate)

変数に [1,2,3,4,5] の5つの要素を持つリストを格納して、添え字が偶数番目の要素だけ出力してください。出力形式は問いません。ただし、先頭の添え字を0番目とします。

期待する出力 : 1 3 5

22. リスト (len)

変数にリスト [11,22,33,44,55,66] を代入し、リストの要素数を出力してください。

期待する出力 : 6

23. リスト, if (存在確認)

リスト [11,22,33,44,55] に、値 44 が存在するかどうかをifを用いて判定してください。存在すれば True, そうでなければ False を出力します。

期待する出力 : True

24. タプル, リストの負のindex

変数に [1,2,3,4,5] の5つの要素を持つリストを格納して、リストの先頭の要素と末尾の要素をタプルとして格納し、出力してください。

期待する出力 : (1, 5)

25. 辞書

変数に、値の組みとして {'A':1, 'B':2, 'C':3, 'D':4, 'E':5} を持つ辞書を格納して、出力してください。

期待する出力 : {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5}

26. 辞書 (keys)

変数に、値の組みとして `{'A':1, 'B':2, 'C':3, 'D':4, 'E':5}` を持つ辞書を格納して、キーを要素としたリストを作成し、出力してください。

期待する出力：`['A', 'B', 'C', 'D', 'E']`

27. 辞書 (values)

変数に、値の組みとして `{'A':1, 'B':2, 'C':3, 'D':4, 'E':5}` を持つ辞書を格納して、値（バリュー）を要素としたリストを作成し、出力してください。

期待する出力：`[1,2,3,4,5]`

28. 辞書 (items)

変数に、値の組みとして `{'A':1, 'B':2, 'C':3, 'D':4, 'E':5}` を持つ辞書を格納して、キーと値（バリュー）の組みのタプルを要素としたリストを作成し、出力してください。

期待する出力：`[('A', 1), ('B', 2), ('C', 3), ('D', 4), ('E', 5)]`

29. 辞書（キーの存在確認）

今、以下のように辞書が作成済みです。

```
d = {'apple':10, 'grape':20, 'orange':30}
```

この辞書に対して、`'apple'` というキーが存在するかを確認し、存在しなければ、`'apple'` というキーに対して-1という値を追加してください。また、同様のことを`'pineapple'` でも行なってください。その後、最終的な辞書を出力してください。

期待する出力：`{'apple': 10, 'grape': 20, 'orange': 30, 'pineapple': -1}`

30. スライス1

変数に文字列 `'training'` を代入し、先頭を0番目として、1番目から4番目までを取り出して出力してください。

期待する出力：`rain`

31. スライス2

変数に文字列 `'understand'` を代入し、先頭を0番目として、奇数番目の文字列だけを取り出した文字列を出力してください。

期待する出力：`nesad`

32. スライス3

変数にリスト `[1,2,3,4,5]` を格納して、これを逆順に出力してください。

期待する出力：`[5,4,3,2,1]`

33. 集合

変数にリスト `[1,1,2,3,3,4,5]` を代入し、このリストを集合に変換して出力してください。

期待する出力：`{1,2,3,4,5}`

34. 積集合

変数 `set1` に集合 `{1,2,3,4,5}` , 変数 `set2` に集合 `{3,4,5,6,7}` を代入し, 積集合を出力してください。

期待する出力 : `{3,4,5}`

35. 和集合

変数 `set1` に集合 `{1,2,3,4,5}` , 変数 `set2` に集合 `{3,4,5,6,7}` を代入し, 和集合を出力してください。

期待する出力 : `{1,2,3,4,5,6,7}`

36. 差集合

変数 `set1` に集合 `{1,2,3,4,5}` , 変数 `set2` に集合 `{3,4,5,6,7}` を代入し, 差集合を出力してください。

期待する出力 : `{1,2}`

37. 型の確認

3つの変数に以下のように代入したコードがあります。各変数の型を出力してください。出力形式は問いません。

```
data1 = {'A':1, 'B':2}
data2 = "hoge"
data3 = {1,2,3,4,5}
```

期待する出力 : `<class 'dict'> <class 'str'> <class 'set'>`

38. strip

文字列 `'This is sentence .\n'` の改行記号を消したものが出力してください。

期待する出力 : `This is sentence .`

39. split

変数に文字列 `'c C++ // python java'` を代入し, この文字列を空白で区切った後のリストを出力してください, また, 同じことを `'/'` でも行い, 出力してください。

期待する出力 : `['c', 'C++', '//', 'python', 'java']`, `['c C++ ', '', 'python java']`

40. join

変数にリスト `['This', 'is', 'a', 'sentence']` を代入し, これらを空白区切で結合した文字列を出力してください。

期待する出力 : `This is a sentence`

41. max

変数にリスト `[11,2,7,13,5]` を代入し, 最大値を出力してください。

期待する出力 : `13`

42. min

変数にリスト [11,2,7,13,5] を代入し、最小値を出力してください。

期待する出力 : 2

43. sum

変数にリスト [11,2,7,13,5] を代入し、総和を出力してください。

期待する出力 : 38

44. 昇順ソート

変数にリスト [5,3,1,4,2] を格納し、昇順に並び替えたリストを出力してください。

期待する出力 : [1,2,3,4,5]

45. ラムダ式

いま、辞書を要素とするリストが次のように与えられています。

```
li = [{ 'a': 6, 'b': 7, 'c': 6},
       { 'a': 4, 'b': 2, 'c': 3},
       { 'a': 1, 'b': 5, 'c': 8}]
```

このとき、辞書のキーが 'b' の値に関して降順になるようにソートしてください。

期待する出力 : [{ 'a': 6, 'b': 7, 'c': 6}, { 'a': 1, 'b': 5, 'c': 8}, { 'a': 4, 'b': 2, 'c': 3}]

46. range()

0 から 99 の100の要素からなるリストを出力してください。

期待する出力 : [0,1,2,...,99] (一部省略)

47. 内包表記

変数にリスト [5,4,3,2,1] を代入し、要素と添字の値（先頭を0とする）を足し合わせた数値を要素に持つリストを新たに作成してください。

内包表記を使うと便利です。

期待する出力 : [5,5,5,5,5]

48. 例外処理

下記のように変数が作られています。

```
a = 0
b = 5
```

この時、 $\frac{a}{b}$ と $\frac{b}{a}$ を出力してください。ただし、ゼロ割りが発生した時には zero division を出力します。

例外処理を用いて書いてみると良いと思います。

期待する出力 : 0, zero division

49. ビット演算

変数 `a` に `10` を、変数 `b` に `5` を代入します。この時、`a` と `b` の論理和、論理積、排他的論理和を出力してください。出力形式は問いません。

期待する出力：`15 0 15`

50. モジュールのインポート

`math` モジュールをインポートして、 $\theta = \frac{\pi}{2}$ の下で、 $\sin\theta^2 + \cos\theta^2$ の値を計算してください。

期待する出力：`1.0`

演習1 (forのネスト)

`1` から `31` までの整数を要素とするリスト1と、`1` から `12` までの整数を要素とするリスト2を作成します。リスト1とリスト2から値を1つずつ取ってきた時、1の位が一致する値が何組あるかを求めてください。例えば、`1` と `11` は、1の位が両方 `1` なので、カウントに入れます。`1` と `1`、`31` と `1` なども同様です。

期待する出力：`38`

演習2 (辞書の値ソート)

辞書が以下のように定義されています。

```
dic = {'two':324, 'four':830, 'three':493, 'one':172, 'five':1024}
```

この時、値で昇順ソートしたと考えた場合のキーの並びを出力してください。

期待する出力：`['one', 'two', 'three', 'four', 'five']`

演習3 (num2freq)

リストが以下のように定義されています。

```
nums = [1,2,4,3,2,1,5,1]
```

このリストに対して、要素の出現数を格納した辞書 `num2freq` を作成し、出力してください。これは `num2freq[要素] = 出現数` となるような辞書で、例えば

`num2freq[1] = 3` です。

期待する出力：`{1: 3, 2: 2, 4: 1, 3: 1, 5: 1}`

演習4 (word2freq)

今、文字列が以下のように与えられています。

```
doc = 'i bought an apple .\ni ate it .\nit is delicious .'
```

`\n` は改行記号なので、3つの文が3行に渡って書かれていることになります。

この文章中の単語を用いて、キーとして単語、値として出現数を持つような辞書 `word2freq` を作成し、出力してください。ただし、改行記号は単語に含めないでください。

ヒント：改行記号で `split` してから空白で `split` すれば、単語に分割できます。

期待する出力：`{'i': 2, 'bought': 1, 'an': 1, 'apple': 1, '.': 3, 'ate': 1}`

```
1, 'it': 2, 'is': 1, 'delicious': 1}
```

演習5 (Jaccard係数)

2つの集合A,Bの類似度を計算する方法として、**Jaccard係数**というものがあります。例えば、文章の類似度の指標として使われています。

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{A \text{と } B \text{の積集合の要素数}}{A \text{と } B \text{の和集合の要素数}}$$

2つのリストが以下のように与えられている時、リストを集合と見なした時のJaccard係数の値を求めてください。

```
list1 = [12, 23, 34, 45, 56, 67, 78, 89]
list2 = [21, 32, 43, 45, 65, 67, 78, 98]
```

期待する出力 : 0.23076923076923078

ごつちやん (id:gotutiyan) 3年前



関連記事

M 2020-10-09
[ICPC 2019 国内予選A 「期末試験の成績」 解説](#)
 問題: <https://onlinejudge.u-aizu.ac.jp/challenges/sources/i...>

n = ; 2020-09-09
[【sklearn】Classification_reportの使い方を丁寧に](#)
 はじめに 本記事ではsklearn.metrics.classification_reportに...

a 2020-04-14
[【python3.x】練習問題55本ノック【問題と解答】](#)
 はじめに ここではpythonの練習問題を掲載しています。データ...

一筆書き

2020-01-14
[【入門】processingで最小限の一筆書きゲームを作る](#)

最小限ゲーム制作 本記事は、processingで一筆書きのゲームを作る記事です。早速...

2019-05-12

シユーティング [【入門】processingで最小限のシユーティングゲームを作る](#)
最小限ゲーム制作 はじめに 座標系 setup()とdraw() 初期設定 自機を作る マウス...

コメントを書く

« [【python3.x】 練習問題55本ノック](#)
【問題...

[書評・感想：リーダブルコード »](#)

はてなブログをはじめよう！

gotutiyansさんは、はてなブログを使っています。あなたもはてなブログをはじめてみませんか？

[はてなブログをはじめる（無料）](#)

[はてなブログとは](#)



gotutiyans blog
Powered by [Hatena Blog](#) | [ブログを報告する](#)