Pythonプログラミングをする時に絶対に注意したほうがいい20のこと

齋藤弦空 著

目次

Pythonプログラミングをする時に絶対に注意したほうがいいこと20

読者限定無料プレゼント

【注意事項】

・本書の内容は、その正確性や完全性を保証するものではありません。

・本書の内容を実践する場合は、自己責任で行ってください。

・著者や出版社は、いかなる損害についても責任を負いかねます。

・小説やフィクションというわけではありませんが、ある種のフィクションのような感覚も常に持ちつつ、どこか俯瞰的な視点を持って、学習したり、参考程度にお楽しみいただければ幸いです。

・本書は、完全にではございませんが、AIを使用しながら書いていますので、過度なクオリティや読みやすい文章の期待はしないでください。ただ、そういう意味では、AIライティングにおける1つの参考例としてお読みいただくのも1つの楽しみ方かなと個人的には思います。

以上のことを踏まえた上で、もし宜しければお読みいただければ幸いです。

少しでも参考になる部分がございましたら幸いです。

また、Kindleを読む際は、読む本について、学びたいことを明確にし、それに集中して読むことによって学習効果が高まることが期待できます。

加えて、Kindleでは、ハイライトとメモを使うことができますので、大事だと思ったところや自分なりの理解をメモすることによって、あとで見返すことができて、より深い理解に繋がりますので、もし宜しければお試しください。

Pythonプログラミングをする時に絶対に注意したほうがいいこと1

スペルミスに注意する。

Pythonプログラミングをする際には、スペルミスに注意することが重要です。プログラミング言語はコンピュータが理解するための言語であり、正確さが求められます。そのため、スペルミスやタイプミスがあると、意図しない結果が生じる可能性があります。

例えば、変数名を誤ってスペルミスをしてしまった場合、その変数が参照されないため、エラーが発生します。また、関数名やモジュール名をスペルミスした場合、Pythonインタプリタはその関数やモジュールを見つけることができず、同様にエラーが発生します。

スペルミスを避けるためには、以下のような方法があります。

・コードを入力する前に、スペルチェックソフトウェアを使用する。

・変数名や関数名を分かりやすく、簡潔にする。

・エディタやIDEを使用して、スペルミスを検出する。

・コードのレビューを行い、スペルミスを修正する。

Pythonプログラミングにおいては、スペルミスがエラーを引き起こす可能性が高いため、スペルミスに注意することが重要です。

Pythonプログラミングをする時に絶対に注意したほうがいいこと2

インデントを正しく設定する。

Pythonプログラミングにおいて、インデント（字下げ）は非常に重要な役割を果たします。インデントは、プログラム内のコードブロックの開始と終了を示すために使用されます。Pythonは、インデントのスペース数によってブロックを区別するため、正しいインデントが必要です。したがって、インデントを正しく設定することが重要です。

例えば、以下のようなコードを考えます。

bash

```
if x > 0:
print("x is positive")
else:
print("x is zero or negative")
```

このコードには、インデントのエラーがあります。Pythonでは、if文のコードブロックはインデントされる必要がありますが、このコードではインデントがされていません。正しいコードは以下のようになります。

bash

```
if x > 0:
    print("x is positive")
else:
    print("x is zero or negative")
```

このコードでは、if文のコードブロックとelse文のコードブロックが正しいインデントで示されています。インデントが適切でない場合、エラーが発生する可能性があります。

インデントを正しく設定するためには、以下のような方法があります。

・スペースとタブを混在させない。

・インデントにはスペースを使用する。

・一貫性のあるインデントスタイルを使用する。

・エディタやIDEのインデント設定を確認する。

・コードレビューを行い、インデントが正しいかどうかを確認する。

Pythonプログラミングにおいて、インデントのエラーは非常に一般的であり、プログラムの動作を誤らせる可能性があります。したがって、インデントを正しく設定することは、Pythonプログラミングにおいて非常に重要なことです。

Pythonプログラミングをする時に絶対に注意したほうがいいこと3

コードをコメントで説明する。

Pythonプログラミングにおいて、コードをコメントで説明することは非常に重要です。コードがどのように動作するかを説明することにより、プログラムをより理解し、保守しやすくします。以下に、コードをコメントで説明することの重要性と方法を示します。

コードを説明することにより、他の人がプログラムを理解しやすくなります。
コードを説明することにより、自分自身が後でプログラムを理解しやすくなります。
コードをコメントで説明することにより、プログラムの意図が明確になります。
コメントを追加する方法は、以下のような方法があります。

行末コメント：コードの横にコメントを追加する。
ブロックコメント：コードブロックの開始と終了の間にコメントを追加する。
ドキュメンテーション文字列：コードの関数やモジュールの先頭に文字列を追加する。
以下は、例として、行末コメントとブロックコメントの追加方法を示します。

python

```
# 以下は、数値を比較して、結果を出力するコードです。
if x > y: # xがyより大きい場合に実行される
    print("x is greater than y")
else: # xがy以下の場合に実行される
```

```
    print("x is less than or equal to y")


"""
以下は、数値を比較するための関数です。
引数として2つの数値を受け取り、大きい方を返します。
"""
def compare_numbers(x, y):
    if x > y:
        return x
    else:
        return y
```

以上のように、コードにコメントを追加することで、プログラムの理解を容易にすることができます。しかし、コメントが多すぎると、逆にプログラムの理解を妨げる可能性があるため、適度なコメントの追加が重要です。

Pythonプログラミングをする時に絶対に注意したほうがいいこと4

変数や関数名を分かりやすくする。

Pythonプログラミングにおいて、変数や関数名を分かりやすくすることは非常に重要です。以下に、変数や関数名を分かりやすくすることの重要性と方法を示します。

・変数や関数名を分かりやすくすることにより、プログラムの意図が明確になります。

・変数や関数名を分かりやすくすることにより、他の人がプログラムを理解しやすくなります。

・変数や関数名を分かりやすくすることにより、自分自身が後でプログラムを理解しやすくなります。

以下は、変数や関数名を分かりやすくする方法の例です。

・変数名：変数が何を表すのかを明確にし、命名規則に従う（snake_caseなど）。

・関数名：関数が何を行うのかを明確にし、命名規則に従う（snake_caseなど）。関数名には動詞を含めることが望ましい。

・定数名：定数を表す変数名は、すべて大文字で記述し、単語間をアンダースコアで区切ることが望ましい。

・クラス名：クラス名は、大文字で始まり、単語の先頭文字も大文字で表記することが望ましい。

以下は、具体的な例です。

ruby

```
# 変数名の例
num_students = 30
max_score = 100
student_names = ["Alice", "Bob", "Charlie"]

# 関数名の例
def calculate_average(scores):
    total = sum(scores)
    num_scores = len(scores)
    return total / num_scores

# 定数名の例
```

```
GRAVITY = 9.8
PI = 3.141592

# クラス名の例
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def calculate_area(self):
        return self.width * self.height
```

以上のように、変数や関数名を分かりやすくすることで、プログラムの理解を容易にすることができます。ただし、名前が長すぎると、逆にプログラムの可読性が低下する可能性があるため、適切な長さを保つことが重要です。

Pythonプログラミングをする時に絶対に注意したほうがいいこと5

不必要なループや処理を避ける。

Pythonプログラムを効率的に実行するには、不必要なループや処理を避けることが重要です。以下に、不必要なループや処理を避ける方法の例を示します。

・ループの回数を最小限に抑える

処理が必要なデータセットのサイズによっては、処理を実行する前にデータをフィルタリングすることで、処理対象を減らすことができます。ループ内で重複する計算を避けることができます。

・内包表記を使用する

内包表記は、1行でリストや辞書を生成する方法です。ループを使用する場合に比べて、簡潔かつ高速に処理できます。

・ジェネレータ式を使用する

ジェネレータ式は、イテレータオブジェクトを生成するための方法です。イテレータを使用する場合に比べて、簡潔かつ高速に処理できます。

・関数を適切に使用する

関数を使用することで、重複する処理をまとめることができます。

・メモ化を使用する

メモ化は、計算結果をキャッシュしておくことで、同じ入力値に対して再計算を行わなくて済むようにする方法です。

以下は、具体的な例です。

python

```
# ループの回数を最小限に抑える
for i in range(len(data)):
    if data[i] > 0:
        process_data(data[i])

# 内包表記を使用する
filtered_data = [x for x in data if x > 0]

# ジェネレータ式を使用する
gen_data = (x for x in data if x > 0)

# 関数を適切に使用する
def process_positive_data(data):
    for x in data:
```

```
        if x > 0:
            process_data(x)

# メモ化を使用する
def fibonacci(n, memo={}):
    if n in memo:
        return memo[n]
    if n <= 2:
        return 1
    memo[n] = fibonacci(n-1, memo) + fibonacci(n-2, memo)
    return memo[n]
```

以上のように、不必要なループや処理を避けることで、Pythonプログラムの実行効率を向上させることができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと6

例外処理を考慮する。

Pythonプログラミングをする際に、例外処理を適切に行うことが非常に重要です。以下は例外処理に関する注意点です。

・エラーメッセージを理解する：エラーが発生した場合、Pythonはエラーメッセージを出力します。エラーメッセージを理解し、問題を修正するために役立てることが大切です。

・try-except文を使用する：例外処理を行うには、try-except文を使用します。tryブロックでコードを実行し、エラーが発生した場合にexceptブロックで処理します。

・全般的な例外を処理しない：例外を処理する場合、全般的な例外を捕捉するためにExceptionを使用するのは避けてください。必要な例外のみを捕捉するようにしましょう。

・finallyブロックを使用する：finallyブロックを使用することで、例外が発生しても必ず実行される処理を指定することができます。ファイルを閉じるなどの必要な後処理を行うために使用することができます。

・raise文を使用する：raise文を使用することで、意図的に例外を発生させることができます。例外を発生させることで、予期せぬエラーが発生することを防ぎ、コードの安全性を高めることができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと7

セキュリティに配慮する。

Pythonプログラミングをする際に、セキュリティに配慮することは非常に重要です。以下はセキュリティに関する注意点です。

・パスワードなどの機密情報を含むデータを安全に保存する：パスワードなどの機密情報を含むデータをファイルに保存する場合は、暗号化することが推奨されます。

・入力の検証を行う：ユーザーからの入力値を使用する場合は、必ず入力の検証を行うようにしましょう。入力値の正当性を確認することで、悪意のある攻撃からプログラムを保護することができます。

・OSコマンドのインジェクションを避ける：プログラム内でOSコマンドを実行する場合は、ユーザーからの入力を直接使用しないようにしま

しょう。ユーザー入力を正しくエスケープすることで、OSコマンドのインジェクション攻撃からプログラムを保護することができます。

・SQLインジェクションを避ける：プログラム内でSQLを実行する場合は、ユーザーからの入力を直接使用しないようにしましょう。ユーザー入力を正しくエスケープすることで、SQLインジェクション攻撃からプログラムを保護することができます。

・ライブラリやフレームワークの脆弱性に注意する：Pythonのライブラリやフレームワークには、脆弱性が存在する場合があります。定期的にライブラリやフレームワークをアップデートし、セキュリティ上の問題が解決されていることを確認するようにしましょう。

Pythonプログラミングをする時に絶対に注意したほうがいいこと8

メモリ使用量を考慮する。

・メモリ使用量が大きい場合は、プログラムの実行速度が低下するだけでなく、システム全体のパフォーマンスにも影響を与える可能性があるため、メモリ使用量を最小限に抑えることが重要です。

・大量のデータを扱う場合は、ジェネレーターやイテレーターを使用することで、一度にすべてのデータをメモリに読み込むことなく、必要な時に必要な分だけ読み込むことができます。

・リストや辞書などの大きなデータ構造を多数作成する場合は、不要なオブジェクトを削除することでメモリを解放し、メモリ使用量を最小限に抑えることができます。

・メモリリークを防ぐために、必要なオブジェクトだけを作成し、不要になったら適切に解放するように注意する必要があります。

・メモリ使用量を測定するために、Pythonの標準ライブラリにはsys.getsizeof()という関数があります。この関数を使用して、オブジェクトが使用するメモリ量を調べることができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと9

関数を小さく保つ。

・関数を小さく保つことで、コードの再利用性や保守性が向上します。また、デバッグやテストも容易になります。

・関数は、一つのタスクや目的に焦点を当てるように設計することが重要です。一つの関数で複数のタスクを処理しようとすると、コードが複雑化し、バグが生じる可能性が高くなります。

・関数の長さは、一般的に10行から50行程度にすることが望ましいとされています。これ以上長くなる場合は、別の関数に分割することを検討しましょう。

・関数名は、その関数が何をするかを簡潔に表すようにしましょう。また、Pythonの慣習に従い、関数名は小文字で、単語間はアンダースコアで区切ることが推奨されています。

・関数には、ドキュメンテーション文字列（docstring）を記述することが重要です。docstringは、関数の説明や引数、戻り値などを記述することができ、コードの可読性を向上させることができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと10

テストコードを書く。

Pythonプログラミングをする際に、以下の点に注意し、テストコードを書くことでプログラムの品質を向上させることができます。

・テスト対象となるコードを明確にする

・正常系だけでなく、異常系のテストケースも用意する

・テストする範囲を狭め、小さな単位でテストする

・テストコードを自動化し、実行しやすい形式にする

・テスト結果を記録し、異常があった場合は原因を特定しやすくする

・テストコードを定期的に実行し、プログラムの品質を維持する

Pythonプログラミングをする時に絶対に注意したほうがいいこと11

コードの再利用性を高める。

Pythonプログラミングをする際に、以下の点に注意し、コードの再利用性を高めることができます。

・処理を関数やクラスにまとめる

・ライブラリや外部モジュールを積極的に活用する

・変数や関数名を汎用的なものにする

・設定値などを外部ファイルに分離する

・ドキュメントやコメントを充実させ、他の人が理解しやすくする

・テストコードを作成し、再利用性を確認する

Pythonプログラミングをする時に絶対に注意したほうがいいこと12

オブジェクト指向プログラミングの原則を守る。

Pythonはオブジェクト指向プログラミングに対応しており、クラスやオブジェクトを利用した開発が可能です。オブジェクト指向プログラミングをする際には、以下の原則に従うことが重要です。

・カプセル化：オブジェクト内部の状態を保護するために、属性やメソッドのアクセスを制限する。

・継承：既存のクラスを拡張した新しいクラスを作成することができ、コードの再利用性を高めることができる。

・ポリモーフィズム：同じインターフェースを持つ異なるクラスやオブジェクトを使い分けることができる。

・抽象化：重要な部分を抽出して共通化することで、コードの複雑性を下げることができる。

これらの原則を守ることで、コードの保守性や再利用性を高めることができます。また、オブジェクト指向プログラミングには独自の用語や概念がありますので、これらにも注意して理解し、適切に使いこなすようにしましょう。

Pythonプログラミングをする時に絶対に注意したほうがいいこと13

コードを定期的に整理する。

Pythonプログラミングにおいて、コードを定期的に整理することは非常に重要です。以下に、その重要性と具体的な内容を示します。

・コードを定期的に整理することにより、コードの可読性が向上し、理解しやすくなります。

・長期間にわたって保守されるコードにおいて、コードが整理されていないと、コードベースが複雑になり、問題を特定し修正するのが困難になることがあります。

・コード整理の一例として、冗長なコードを削除したり、重複したコードをまとめたりすることが挙げられます。

・また、関数やクラスなどのコードの塊を適切に整理し、それらをモジュールやパッケージに分割することも重要です。これにより、コードの再利用性が高まります。

・コードの整理には、コードレビューを行うことも有効です。他の人が見たときに、どのように見えるかを確認することで、コードを改善することができます。

・最後に、コード整理はプロジェクトの品質を向上させるための重要なステップであり、定期的に実施することが望ましいです。

Pythonプログラミングをする時に絶対に注意したほうがいいこと14

エラーのデバッグに時間をかける。

プログラムを書く際に、エラーは避けられないものであり、コードにバグがあるときにはエラーメッセージが表示されます。

しかし、エラーメッセージだけでは原因がわからない場合があります。そのため、エラーのデバッグに時間をかけることが重要です。具体的には、以下のような対策を取ることが必要です。

・エラーメッセージを確認する

・ログを取る

・デバッガーを使用する

・コードを小さくする

・コードを分割する

・コメントを付ける

これらの対策を取ることで、エラーを早期に発見し、修正することができます。エラーの修正に時間をかけることで、プログラムの品質を高めることができ、より安定したプログラムを作成することができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと15

計算量を考慮する。

以下は、計算量を考慮する上で注意すべきポイントの例です。

・ループの数を減らす：Pythonは柔軟な言語であり、ループの使用が多いことがありますが、必要最小限にとどめることが望ましいです。特に、多数の要素を処理する場合は、ループ内で高コストな操作を繰り返すのではなく、一度のループで複数の操作を行うようにします。

・適切なデータ構造を選択する：データの操作は、適切なデータ構造を使用することによって最適化できます。リスト、タプル、セット、辞書、配列など、Pythonにはさまざまなデータ構造があります。各データ構造には、特定の操作に最適なものがあります。

・大規模データに対応する：Pythonは大量のデータを処理できるように設計されていますが、適切な工夫を行わない場合、処理が遅くなることがあります。大量のデータを処理する場合は、適切な分割やバッチ処理を行うことが必要です。

・キャッシュを使用する：キャッシュを使用することで、同じ処理を繰り返さなくて済みます。特に、計算の結果を再利用する必要がある場合、キャッシュを使用することで計算時間を大幅に短縮できます。

・ライブラリを使用する：Pythonには、科学計算やデータ処理など、特定のタスクに最適化されたライブラリがたくさんあります。これらのライブラリを活用することで、より高速で効率的なコードを書くことができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと16

クロスプラットフォームにする。

具体的には、以下のようなポイントに注意することが重要です。

・プラットフォームごとにファイルパスの区切り文字が異なるため、ファイルパスを処理する場合は、osモジュールの関数を使用して、プラットフォームごとに適切な区切り文字を使用するようにします。

・同様に、改行コードもプラットフォームによって異なるため、osモジュールの関数を使用して、プラットフォームごとに適切な改行コードを使用するようにします。

・ファイルのエンコーディングもプラットフォームごとに異なる場合があるため、ファイルを開くときには、エンコーディングを指定するようにします。

・プラットフォームによっては、Pythonが標準でインストールされていない場合があります。その場合は、プログラムを実行する前にPythonがインストールされているかどうかを確認する必要があります。

・また、外部ライブラリを使用する場合は、ライブラリが各プラットフォームで利用可能かどうかを確認する必要があります。

これらの注意点を踏まえて、Pythonプログラムをクロスプラットフォームにすることで、ユーザーが使用するプラットフォームに依存しない、柔軟性の高いプログラムを開発することができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと17

リファクタリングを頻繁に行う。

リファクタリングとは、コードの品質を向上させるために、コードの構造や設計を改善する作業のことです。Pythonプログラミングにおいては、リファクタリングを頻繁に行うことで、コードの保守性や拡張性を高め、バグの発生を減らすことができます。

リファクタリングを行うためには、まずコードを慎重に分析し、改善の余地がある箇所を見つける必要があります。これには、コードレビューやコード解析ツールの使用などが役立ちます。

次に、改善が必要な箇所を修正する必要があります。この際には、コードの意図が変わらないように注意することが重要です。また、変更の影響が他の部分に及ばないようにテストを行い、問題がないことを確認する必要があります。

最後に、改善したコードをドキュメント化し、チーム全体で共有することで、今後の開発作業をスムーズに進めることができます。

リファクタリングは、一度だけ行うものではありません。常にコードの品質を向上させるために、定期的に行うことが必要です。

Pythonプログラミングをする時に絶対に注意したほうがいいこと18

コードの可読性を高める。

Pythonプログラミングをする際に、コードの可読性を高めることは非常に重要です。以下に、可読性を高めるために注意すべきポイントをいくつか挙げます。

1.変数や関数名を意味のある名前にする

変数名や関数名は、その役割がわかるように意味のある名前をつけるようにしましょう。例えば、aやbといった単語では何を表しているのかわかりにくいため、countやtotalといった名前を使うと、コードの意図が明確になります。

2.適切なコメントをつける

コード内には、処理の内容や目的が分かりにくい部分があることがあります。そのような場合は、適切なコメントをつけるようにしましょう。ただし、あまりにも冗長なコメントや、コードの内容と全く関係のないコメントは避けるようにしましょう。

3.適切なインデントを使う

Pythonの場合、コードブロックはインデントで表現されます。適切なインデントを使うことで、コードの構造がわかりやすくなります。また、Pythonではインデントによってプログラムの意味が変わるため、注意が必要です。

4.関数の引数や戻り値を明確にする

関数の引数や戻り値には、意味がある名前をつけるようにしましょう。また、関数のドキュメント化を行うことで、その関数が何をするものかを明確にすることができます。

5.複雑な処理は分割する

複雑な処理は、複数の関数に分割することで、コードの可読性を向上させることができます。また、分割することで、再利用性も高まります。

6.コードのフォーマットを整える

コードのフォーマットは、読みやすさに大きく影響します。コードを整形するためのツールやライブラリが多数存在するため、利用してみることをおすすめします。

以上のように、コードの可読性を高めるためには、意味のある名前をつけたり、適切なコメントをつけたりすることが大切です。また、コードを整形することで、読みやすさを向上させることができます。

Pythonプログラミングをする時に絶対に注意したほうがいいこと19

コード規約を守る。

PythonにはPEP 8と呼ばれる公式のコーディング規約があり、この規約に従うことでコードの読みやすさや保守性が向上します。

具体的には、以下のような点に気をつけることが重要です。

・インデントを揃える：スペース4つ分を1段階とし、全体的に揃えます。

・変数や関数名を小文字でスネークケース（アンダースコアで単語をつなげた形）にする：例えば、my_variable、my_functionなど。

・クラス名を大文字でキャメルケース（単語の先頭を大文字にしてつなげた形）にする：例えば、MyClassなど。

・80文字を超える行は避ける：長くなる場合は改行する。

・文字列はシングルクォート（'）で囲む：ダブルクォート（"）でも構いませんが、一貫性を持たせるためにシングルクォートを推奨します。

・コメントを入れる：コードの意図や処理内容を説明するコメントを入れることで、コードの読みやすさが向上します。

このように、コード規約を守ることで、コードの品質を向上させることができます。特に、複数人での開発やメンテナンスを行う場合には、一貫性のあるコードが重要となります。

Pythonプログラミングをする時に絶対に注意したほうがいいこと20

バージョン管理システムを使用する。

バージョン管理システムを使用することは、Pythonプログラマーにとって非常に重要なことです。バージョン管理システムを使用することで、プログラムの履歴を追跡することができます。これにより、問題が発生した場合に以前のバージョンに戻すことができます。また、複数の

人々が同じプログラムに作業する場合にも、コードの競合を解決するために非常に役立ちます。

Gitは、Pythonプログラミングにおいて最も一般的に使用されるバージョン管理システムの1つです。Gitを使用すると、ソースコードの変更履歴を追跡し、異なるバージョンを比較して差分を確認することができます。また、Gitを使用して、コードの異なるバージョンをブランチと呼ばれる異なる分岐で管理することもできます。これにより、コードの変更を追跡し、プロジェクトの進行状況を管理することができます。

バージョン管理システムを使用することで、プログラムの品質を向上させ、チームワークを容易にし、より信頼性の高いコードを開発することができます。したがって、Pythonプログラミングをする際には、バージョン管理システムを使用することが非常に重要です。

読者限定無料プレゼント

現在、出版グループ「シンザンパレット」のLINE公式アカウントを友達追加していただいた方に、

・アクセスを上げる方法に関する音声（52分29秒）

・売れるアイディアを発想する方法に関する音声（34分16秒）

・仕組み化に関しての音声（21分47秒）

の3つの音声を無料でプレゼントしています。

※プレゼント内容は、その時々で異なる可能性があります。ご了承の上、友達追加してください。

「シンザンパレット」のLINE公式アカウントでは、人生を豊かにするためのヒントや「ここだけは絶対に注意したほうがいい！」という注意点をご紹介予定です。

無料で友達追加できますので、もし宜しければ友達追加してください。

今後、さらに豪華な驚くようなプレゼントをする可能性もありますの

で、ぜひご期待ください。

シンザンパレットのLINE公式アカウント

https://lin.ee/P8iTcqp

Pythonプログラミングをする時に絶対に注意したほうがいい20のこと

著者　齋藤弦空

作成日　令和5年4月16日