

# Computer Science Programming Fundamentals

## Readability



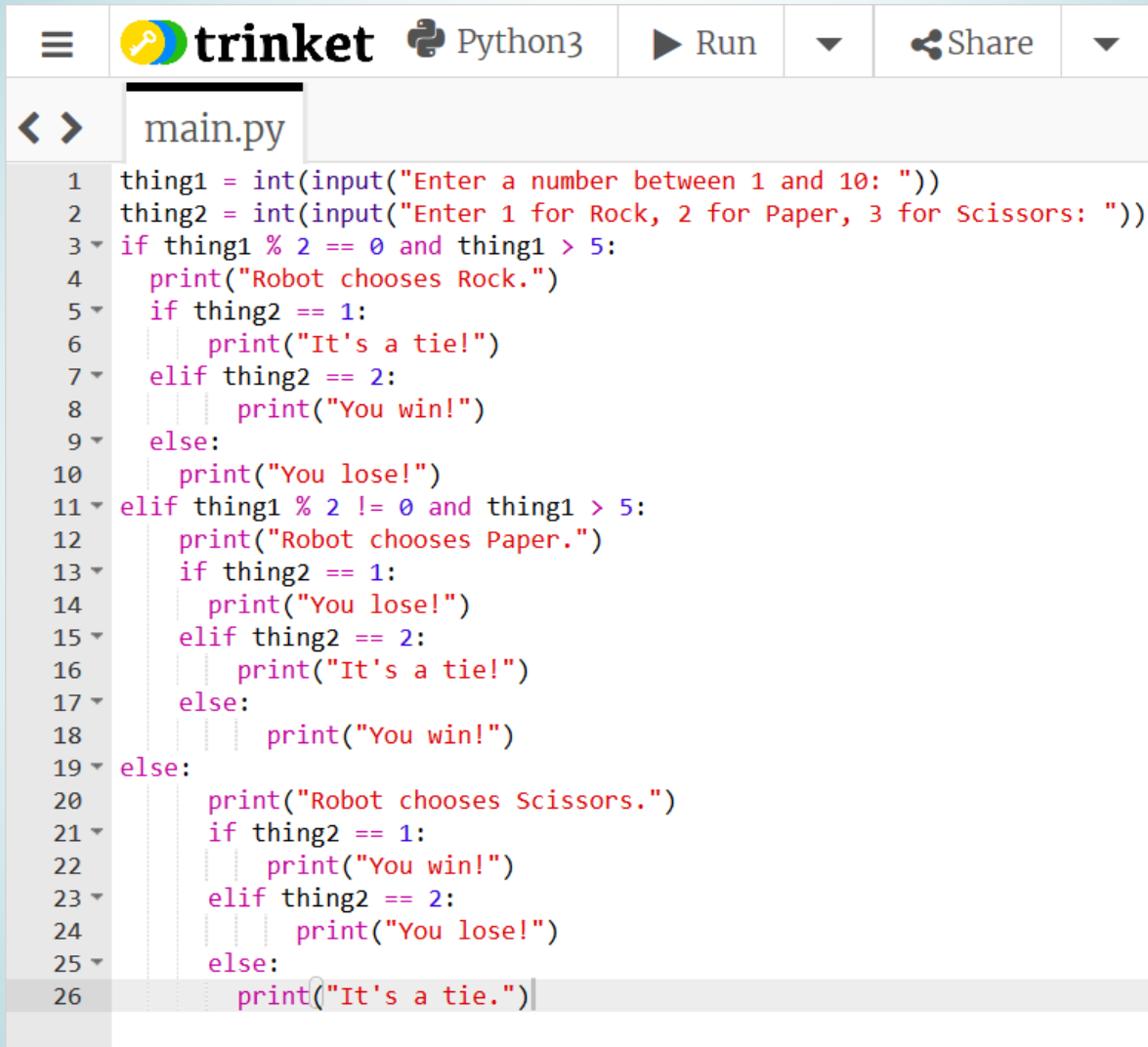
# 💡 Readability Intro

Is your handwriting hard to read? Have you ever tried to read something you wrote, but couldn't because it was so messy? Although we write code by typing, something similar can happen if we aren't careful.

This lesson will explore the basics of how we can make sure our code is easy to read and understand for ourselves and for others.

# Readability pt 1

- Run the example:



The screenshot shows a Trinket Python3 editor interface. The file is named 'main.py'. The code is a Rock-Paper-Scissors game where a robot chooses a move based on a random number (thing1) and the user chooses a move (thing2). The code is poorly formatted, with inconsistent indentation and lack of comments, making it difficult to read.

```
1 thing1 = int(input("Enter a number between 1 and 10: "))
2 thing2 = int(input("Enter 1 for Rock, 2 for Paper, 3 for Scissors: "))
3 if thing1 % 2 == 0 and thing1 > 5:
4     print("Robot chooses Rock.")
5     if thing2 == 1:
6         print("It's a tie!")
7     elif thing2 == 2:
8         print("You win!")
9     else:
10        print("You lose!")
11 elif thing1 % 2 != 0 and thing1 > 5:
12     print("Robot chooses Paper.")
13     if thing2 == 1:
14         print("You lose!")
15     elif thing2 == 2:
16         print("It's a tie!")
17     else:
18         print("You win!")
19 else:
20     print("Robot chooses Scissors.")
21     if thing2 == 1:
22         print("You win!")
23     elif thing2 == 2:
24         print("You lose!")
25     else:
26         print("It's a tie.")
```

This code works, but it's hard to understand for several reasons. Let's improve it!

# Readability pt 2

There are two variables defined at the top: `thing1` and `thing2`. You can name a variable whatever you want, but it's best to name it something that tells us what it does.

- Change:

```
thing1 = ...
```

```
thing2 = ...
```

to

```
user_number = ...
```

```
user_choice = ...
```

# Readability pt 3

Adding comments to our code can make it easier to understand.

To add a comment in Python, use the `#` symbol. Python will ignore everything on the line after it, so it won't mess up your code.

- Add comments to the three main if statements in the Rock, Paper, Scissors code to explain what each one does like so: notice the space!

```
# If user's number is even and bigger  
than 5, the robot will choose rock.
```

```
if ... :
```

```
    ...
```

```
# If user's number is odd and bigger  
than 5, the robot will choose paper.
```

```
elif ... :
```

```
    ...
```

```
# Otherwise, the robot will choose  
scissors.
```

```
else ... :
```

```
    ...
```

# Readability pt 4

Consistent indentation makes code easier to read and can protect it from an **IndentationError**.

- Correct the bad example code following these rules:
  - Anything not in an if statement should start at the "wall."

```
1 variable = "I start at the wall!"
```

- Anything inside **ONE** if statement should start **four spaces** from the wall.

```
1 location = "inside one if statement"
2
3 if location == "inside one if statement":
4     print("I am four spaces from the wall!")
5     print("Me too!")
```

- Anything inside **two** if statements should start **eight spaces** from the wall.

```
1 location = "inside one if statement"
2 depth = 2
3
4 if location == "inside one if statement":
5     if depth == 2:
6         print("We are")
7         print("Eight far")
8         print("Har har")
```



# Readability pt 5

Look back at the code examples on the previous page. Notice that when we define a variable, the pattern is `name = value`. There is one empty space on each side of the equal sign. Also, the last two examples both have an empty line.

These empty spaces and lines are called **whitespace**.

Adding whitespace breaks up your code visually and makes it much easier to read.

- Look back at the example code in previous lessons to get a feel for how whitespace can be used for better readability.

# Readability pt 6



## Activity:

- Take the Rock, Paper, Scissors robot code you made in the previous lesson and make it ugly. Break the rules about indentation, variable names, and whitespace that you learned in this lesson. Make sure the code still runs without errors, though!
- Trade your ugly code for someone else's ugly code and fix the ugliness! Add comments to make it easy to understand.

## Ugly Example:

See the code example at the beginning of this lesson.

Good example:



```
num = int(input("Enter a number between 1 and 10: "))

user_choice = int(input("Enter 1 for Rock, 2 for Paper,
3 for Scissors: "))

# If user's number is less than or equal to 3, Robot
will choose rock.

if num <= 3:
    print("Robot chooses Rock.")
# Tell the user the outcome given the robot's choice.
    if user_choice == 1:
        print("It's a tie!")
    elif user_choice == 2:
        print("You win!")
    else:
        print("You lose!")
# If user's choice is less than or equal to six, Robot
will choose paper.
elif num <= 6:
    print("Robot chooses Paper.")
# Tell the user the outcome given the robot's choice.
    if user_choice == 1:
        print("You lose!")
    elif user_choice == 2:
        print("It's a tie!")
    else:
        print("You win!")
```

Continued on next page...

## ... Continued from previous page

```
# If user's choice is greater than 6, just choose
scissors.
else:
    print("Robot chooses Scissors.")
# Tell the user the outcome given the robot's choice.
    if user_choice == 1:
        print("You win!")
    elif user_choice == 2:
        print("You lose!")
    else:
        print("It's a tie.")
```

# Readability pt 8



## What to look for:

- The variable names are reasonably specific, giving an idea of what they do.
- The comments explain key parts of the code.
- The indentation makes it clear which lines are part of if statements and which aren't.
- The white space makes it reasonably easy for the eye to track down different parts of the code on the page.