Readme!

Feel free to edit this template by adding / editing and removing slides. I've tried to cover most of what you might want to include but there are many ways of providing evidence so don't be afraid to delete slides that you don't need.

Slides with a green background should be used to show planning evidence of the individual components and the testing procedures you made during the development process. Orange slides can be used to show evidence of the refinements and final testing of the completed program.

Program name goes here: Shapes_program_v2.py

Link to github Repository: Link

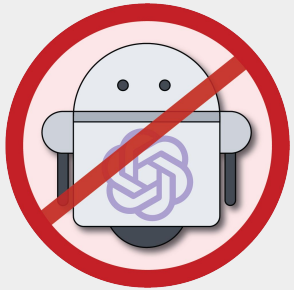
Links to trello board / project management tools: Link #2

Link to a short (2-3 mins) program testing video:
https://drive.google.com/file/d/1VqRi2kNlApwC7rPlZeMU7puIc-OTkAEC/view?usp=drive_link

Name of the final completed program: Perimeter/Area Calculator

You MUST provide evidence showing how the problem has been decomposed, how the components have been developed and trialled, and of how they have been assembled and tested to create a final, working outcome.

Authenticity!!



Using generative AI to 'help' is not allowed. NZQA is interested in your knowledge and skills. If you use external resources (like Stack OverFlow, W3C please cite / acknowledge the source).

By submitting this assessment, you declare that the work is your own. This means that you have generated the content yourself and have not looked at any other student's work. If you have used external sources, those have been clearly cited.

Relevant Implication - Functionality

Explain a relevant implication and how you addressed it. Please watch this video to learn how to do this.

Functionality is how well it works and making sure that it works properly and as intended. It should not crash if the wrong input is entered but it should correct the user or put them in the right direction. Functionality is important because it is there to keep the work running and stop it from crashing.

Relevant Implication - Usability

Explain a relevant implication and how you addressed it.

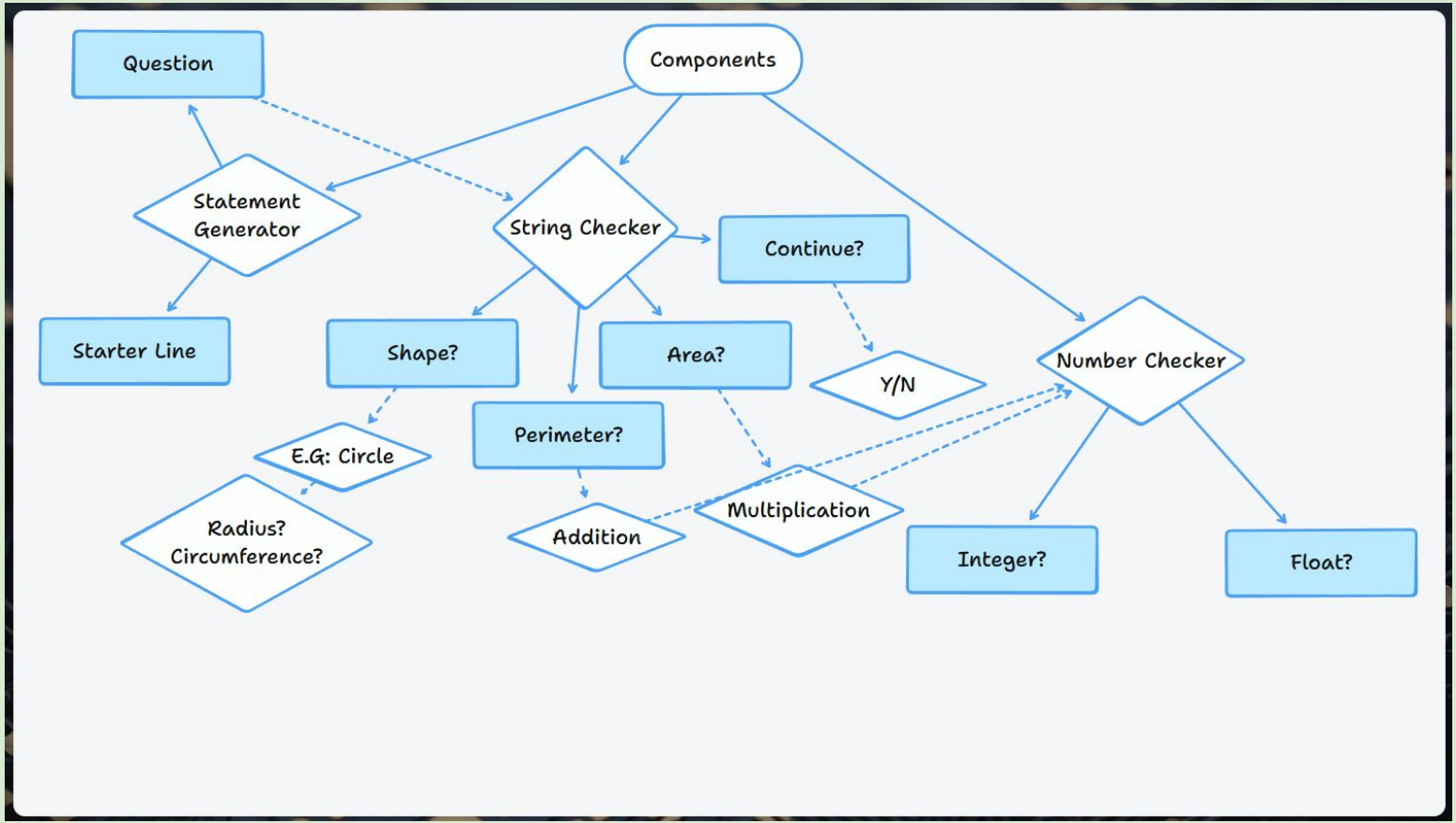
Usability is when you make sure that your code or creation is easy to use without too much hassle or learning and it is pretty straightforward on what they need to do. Usability is important because it makes sure the user doesn't have to learn new skills to be able to use the work or code.

Relevant Implication - End User

Explain a relevant implication and how you addressed it.

End user implications just means that you need to make sure that it can meet the needs of your target audience and what they will need it for, making sure that it does the right thing for them.

Decomposition



Updated Planning

Components	**Statement Generator**	**Number Checker**	**String Checker**
Statement Generator	Start Line Simple (emoji's, instructions)	Integer or Float? (whole or decimal)	Yes_no checker
Number Checker	User History	Between {high} & {low} variables	List Names
String Checker		Variable numbers	

This update shows my plan in a easie visuality that makes it easier to understand what is in each component while the flow chart showed what happened when something else happened. It also shows the changes I made in the statement generator section taking away one part and adding and changing different parts in the other components

Planning - Component 1

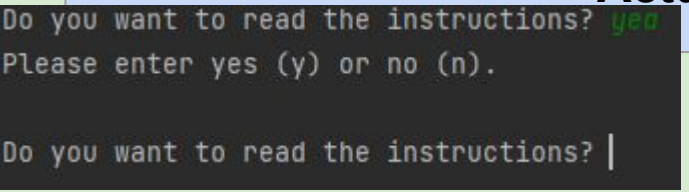
A String checkers makes sure that the user enters the right word with the right spelling and if the user doesn’t then it gives the user the right options that they can choose from to make sure that they enter the right answer with the right spelling

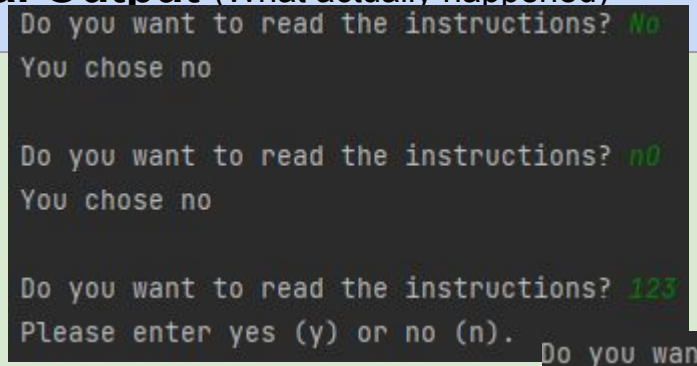
Component 1 - Test Plan (Trello and testing screenshot)

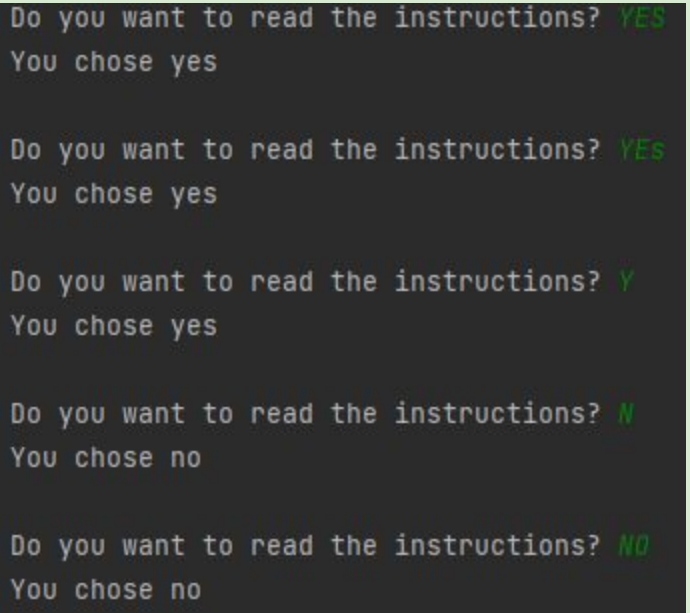
Data Input	**Expected Output** (What should happen)
Yes	Shows instructions / more info
Nope	Re-ask the question/rewrite the statement and gives correct options to write
No	Write the next part / program continues
Yea	Re-ask the question/rewrite the statement and gives correct options to write
123	Re-ask the question/rewrite the statement and gives correct options to write
Y	Shows instructions / more info

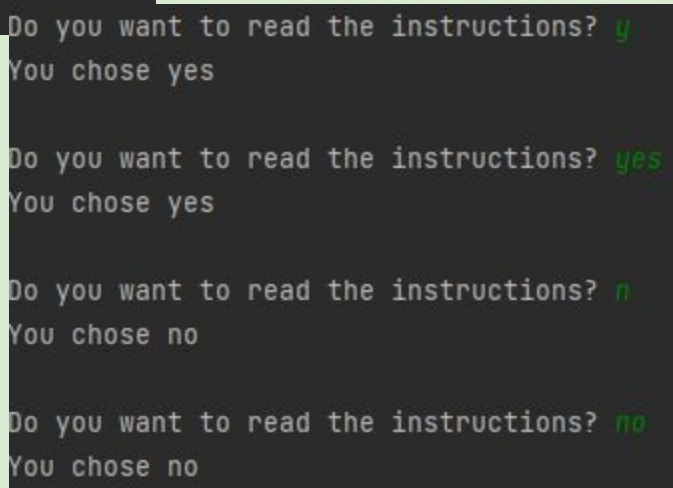
Component 1 - Test Plan (Trello and testing screenshot)

Actual Output (What actually happened)

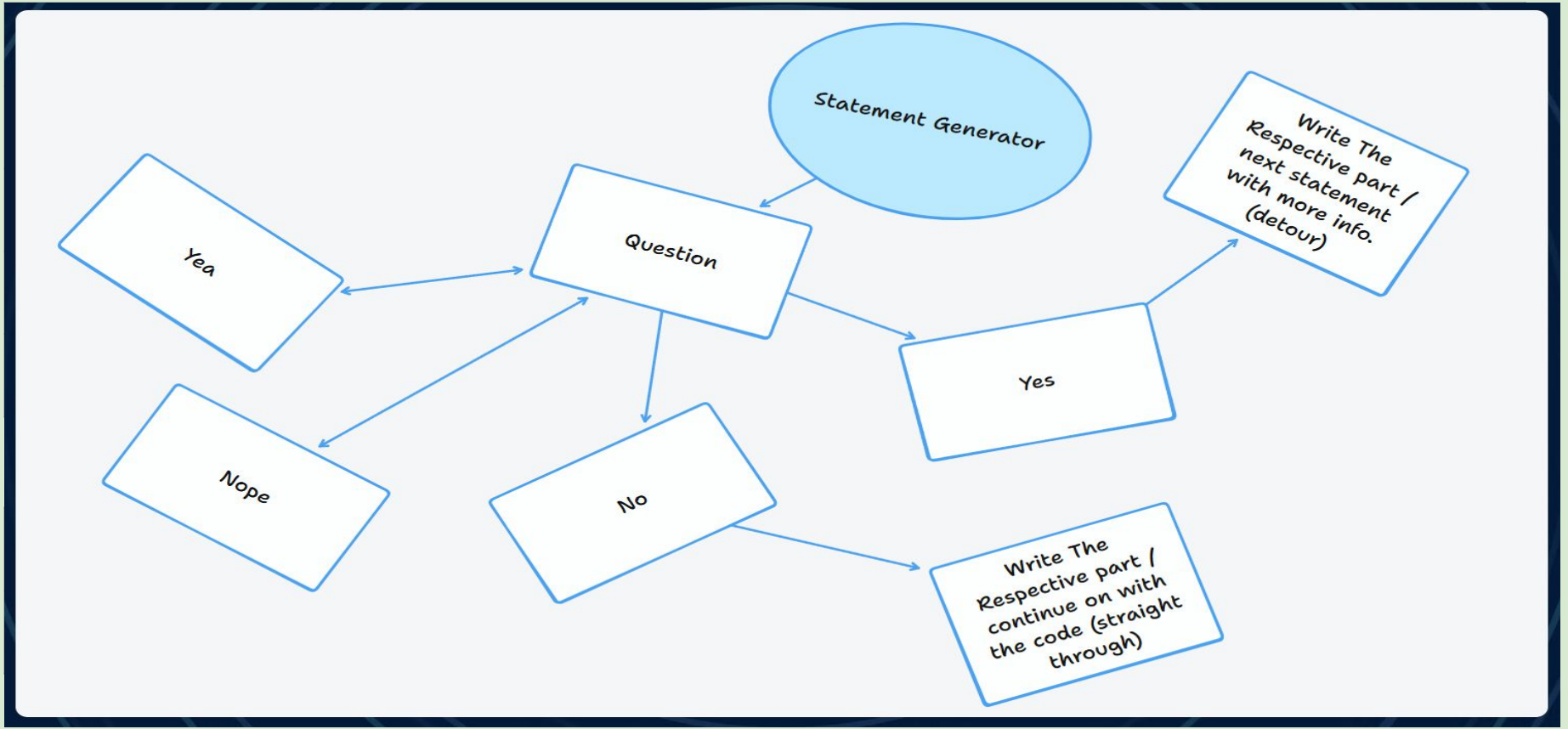








Component 1 - Test Plan (Trello and testing screenshot)



Component Trialing

I have 2 versions of a string checker. Version 1 makes it so the the user has to enter the whole word from the list while version 2 makes it so that the user only has to enter 1 letter from the words unless there are two words with the same starting letter in which case I will change the code so that the user needs to enter at least 2 or 3 etc… I chose version two as it is more user friendly for users that can’t spell very well.

Component Trialing

shapes_program_v1.py | C_04_string_check_v2.py

```python
# Functions go here
def string_check(question, valid_ans_list, num_letters):
    """Checks that the users enter the full word
    or the 'n' letter/s of a word from a list of valid responses"""

    while True:

        response = input(question).lower()

        for item in valid_ans_list:

            # check if the response is the word
            if response == item:
                return item

            # check if it's the 'n' letter
            elif response == item[:num_letters]:
                return item

        print(f"Please choose an option from {valid_ans_list}")


# Variables
yes_no = ["yes", "no"]


# Main routine goes here
like_coffee = string_check("Do you like coffee? ", yes_no, 1)
```

```python
# Functions go here
def string_check(question, valid_ans_list):
    """Checks that the users enter the full word
    or the first letter of a word from a list of valid responses"""

    while True:

        response = input(question).lower()

        for item in valid_ans_list:

            # check if the resopnse is the word
            if response == item:
                return item

            # check if it's the first letter
            elif response == item[0]:
                return item

        print(f"Please choose an option from {valid_ans_list}")


# Main routine goes here
levels = ['easy', 'medium', 'hard']

like_coffee = string_check("Do you like coffee?", ['yes', 'no'])
choose_level = string_check("Choose a level: ", levels)
```

Planning - Component 2

Integer Checker:
An integer checker is all about making sure that the user enters a suitable whole number and if the user has not entered the right type of number or number in the range then the program will tell them they need to enter a whole number between {low} & {high} values

Component 2 - Test Plan (Trello and testing screenshot)

Data Input	**Expected Output** (What should happen)
3	Continue with the code
49	Continue with the code
51	Tell user they need to enter a “whole number between 1 & 50
0	Tell user they need to enter a “whole number between 1 & 50
50	Continue with the code
1	Continue with the code
2.1	Tell user they need to enter a “whole number between 1 & 50

Component 2 - Test Plan (Trello and testing screenshot)

Actual Output (What actually happened)

```
Please enter one of the sides of your rhombus: 51
Oops - please enter a whole number between 1 and 50
Please enter one of the sides of your rhombus:
```

```
Please enter one of the sides of your rhombus: 0
Oops - please enter a whole number between 1 and 50
Please enter one of the sides of your rhombus:
```

```
Please enter one of the sides of your rhombus: 2.1
Oops - please enter a whole number between 1 and 50
Please enter one of the sides of your rhombus:
```

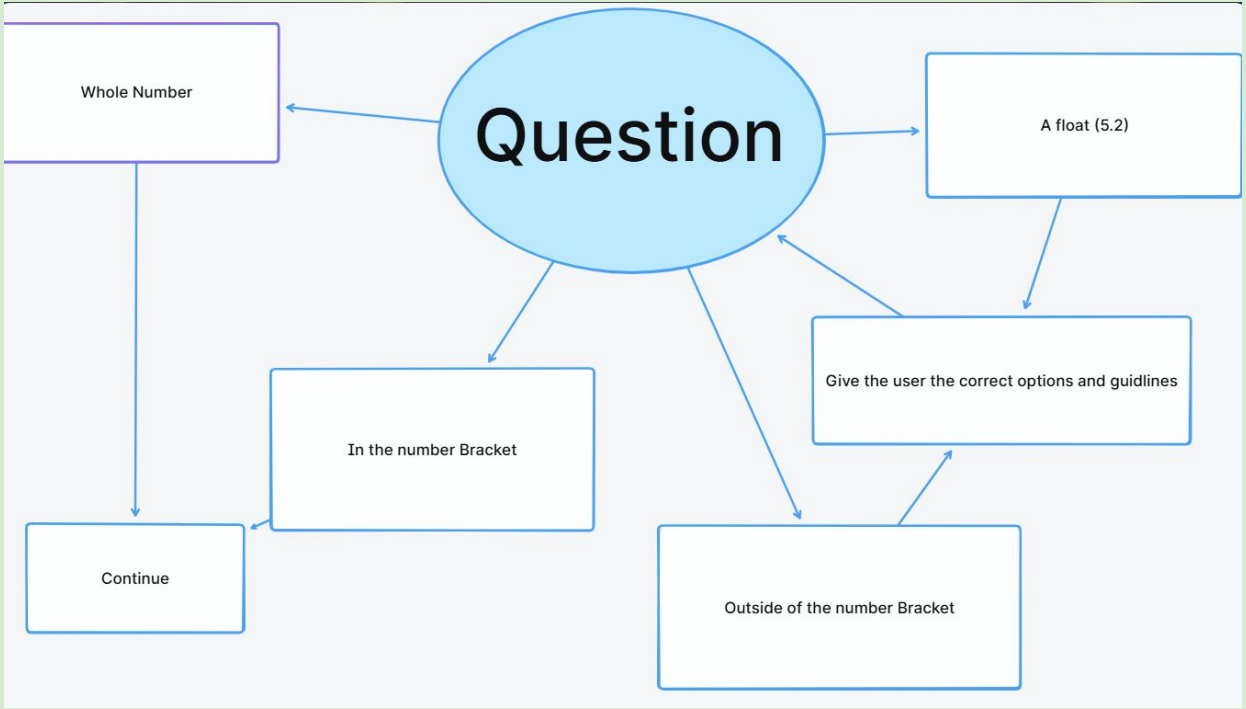
```
Please enter one of the sides of your rhombus: 3

Please enter the length of your rhombus:
```

```
Please enter the length of your rhombus: 49

Please enter the height of your rhombus:
```

```
Please enter the length of side Nº 1: 1

Please enter the length of side Nº 2: 50
```

Component 2 - Test Plan (Trello and testing screenshot)

Actual Output (What actually happened)



Component Trialing

I had two versions of a number checker the first one was to specify and make the user only enter a specific type of number (e.g: Float, Integer). The second version was made to allow the user to enter either type of number while giving them a high and low value to choose from. I chose the second version because for a calculator the user needs to be able to enter either type of number so that they can get the right answer to their question.

Component Trialing

```python
# Functions go here
def num_check(question, low, high):
    """Checks users enter an integer between two values"""

    error = f"Oops - please enter an integer between {low} and {high}"

    while True:

        try:
            # Change the response to an integer and check that it's more than zero
            response = int(input(question))

            if low <= response <= high:
                return response
            else:
                print(error)

        except ValueError:
            print(error)

# Main Routine goes here

# loop for testing purposes...
while True:

    print()

    # ask user for an integer
    my_num = num_check("Choose a number: ", 1, 10)
    print(f"You chose {my_num}")
```

```python
# Functions go here
def num_check(question, num_type, exit_code=None):
    """Checks users enter an integer / float that is more than
    zero (or the optional exit code)"""

    error = "Oops - please enter an integer more than zero"

    while True:
        response = input(question).lower()

        # check for exit code
        if response == exit_code:
            return response

        try:
            # Change the response to an integer and check that it's more than zero
            response = num_type(response)

            if response > 0:
                return response
            else:
                print(error)

        except ValueError:
            print(error)

...
while True:
    print()

    my_float = num_check("You can enter a decimal or whole number more than 0: ", float)
    print(f"Thanks. You chose {my_float}")
    print()
    my_int = num_check("You can only enter a whole number more than 0: ",
                       int, "")
    if my_int == "":
        print("You have chosen infinite mode")
    else:
        print(f"Thanks. You chose {my_int}")
```

Work Log

7/08/2026:
I worked on getting the while loop created and working correctly
5/08/2026:
I added in the code for adding together the numbers that the user enters and putting them into a formula to work out the answer for the perimeter and area.
3/08/2026:
I added in all of the different shapes and started working on what to write for them
27/07/2026:
I started working on what shapes I was going to add
23/07/2026:
I started to work on the instructions and what they were going to say
21/07/2026:
I added in the functions that I was going to be using in my code
20/07/2026:
I created the new file and finished of working on the videos to get what I needed for my code

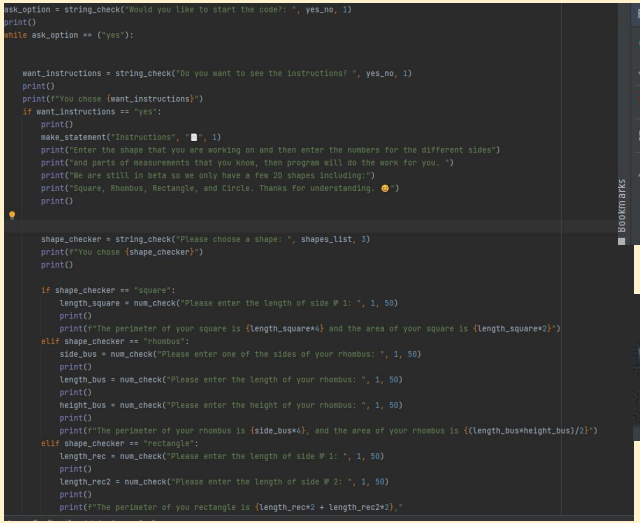
Version Control

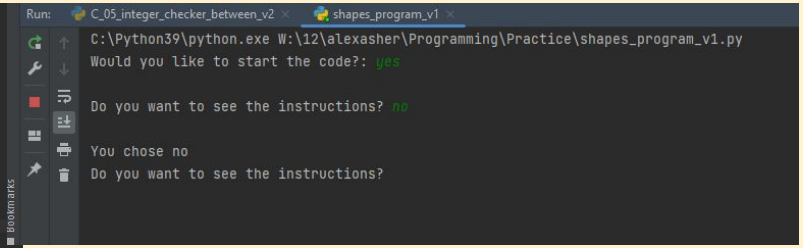
The version one of my code is done in a way so that it can take only small numbers and that I just wrote the code instead of making small fixtures to make some bits of the code a bit smaller. The version 2 of my code makes it so that there is one more function or an import to make the code easier as well as more uses of my while loops to make sure that the code works as expected as well as bypassing things on the rerun through to make it so that the code can run quicker. I also made the number high values higher so that the user can do bigger shapes.

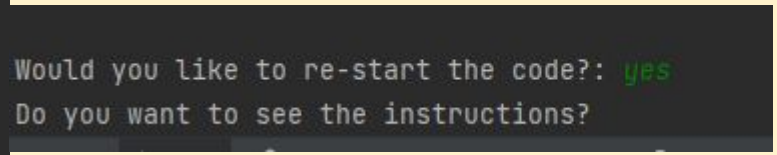
Improvements & Refinements

Before

Before this improvement if I didn't want to see the instructions I was not able to continue with the code until I said yes and when I restarted the code it asked me if I wanted to see the instructions again and I needed to bypass this



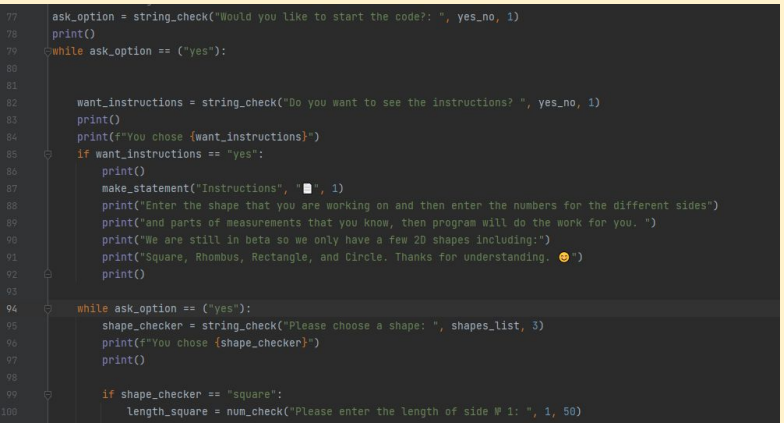


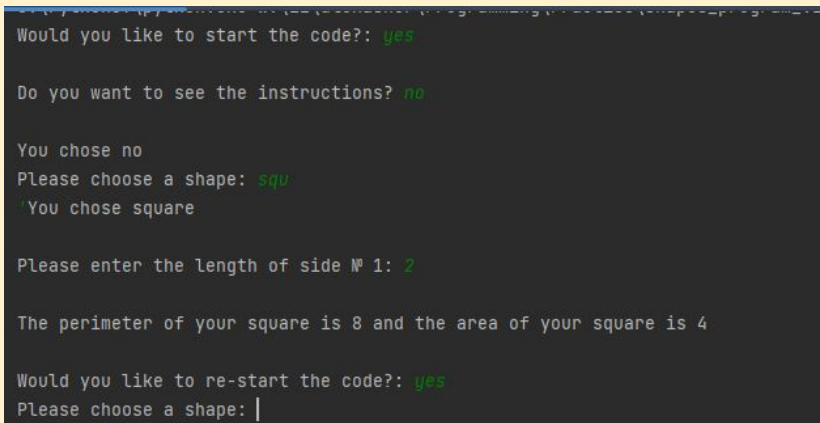


Improvements & Refinements

After

After I added a bit of code in line 94 I rechecked my code and reran all the bits that I needed to and the bit of code that I added in made it so the if I said no to reading the instructions it didn't ask me again and also when I said yes to restarting the code it didn't ask me if I needed the instructions again which is what I wanted it to do.





Improvements & Refinements

Before	After
Before I changed my code and made it better I had to manually write out 3.14 into my code for when I needed to multiply by PI which for me was tedious and made the code look a bit messy,	So I added in an import for PI and changed the code around to PI instead of 3.14 which made the code look more fancy and less messy for me.

Improvements & Refinements

Before	After

```
    if shape_checker == "square":
        length_square = num_check("Please enter the length of side № 1: ", 1, 50)
        print()
        print(f"The perimeter of your square is {length_square*4} and the area of your square is {length_square*2}")
    elif shape_checker == "rhombus":
        side_bus = num_check("Please enter one of the sides of your rhombus: ", 1, 50)
        print()
        length_bus = num_check("Please enter the length of your rhombus: ", 1, 50)
        print()
        height_bus = num_check("Please enter the height of your rhombus: ", 1, 50)
        print()
        print(f"The perimeter of your rhombus is {side_bus*4}, and the area of your rhombus is {(length_bus*height_bus)/2}")
    elif shape_checker == "rectangle":
        length_rec = num_check("Please enter the length of side № 1: ", 1, 50)
        print()
        length_rec2 = num_check("Please enter the length of side № 2: ", 1, 50)
        print()
        print(f"The perimeter of you rectangle is {length_rec*2 + length_rec2*2},"
              f" and the area of your rectangle is {length_rec*length_rec2}")
    else:
        circumference = num_check("please enter the radius of your circle: ", 1, 50)
        print()
        print(f"The circumference of your circle is {2*3.15*circumference}, and the area of your circle is "
              f"{3.15*circumference**2}")
    print()
    ask_option = string_check("Would you like to re-start the code?: ", yes_no, 1)

else:
    print()
    print("Thanks for using this calculator, have a good day. 😊")
    print("The program has ended")
```

```
# Functions go here

def num_check(question, low, high):
    """Checks users enter an integer between two values"""

    error = f"Oops - please enter a whole number between {low} and {high}"

    while True:

        try:
            # Change the response to an integer and check that it's more than zero
            response = int(input(question))

            if low <= response <= high:
                return response
            else:
                print(error)

        except ValueError:
            print(error)


def make_statement(statement, decoration, lines=1):
    """Creates headings (3lines), subheadings (2 lines) and
    emphasized text / mini-headings (1 line).
    Only use emoji for single line statements"""

    middle = f"{decoration * 3} {statement} {decoration * 3}"
    top_bottom = decoration * len(middle)

    if lines == 1:
        print(middle)
    elif lines == 2:
        print(middle)
```

```
            length_rec2 = num_check("Please enter the length of side № 2: ", 1, 50)
            print()
            print(f"The perimeter of you rectangle is {length_rec*2 + length_rec2*2},"
                  f" and the area of your rectangle is {length_rec*length_rec2}")
        else:
            circumference = num_check("please enter the radius of your circle: ", 1, 50)
            print()
            print(f"The circumference of your circle is {2*pi*circumference}, and the area of your circle is "
                  f"{pi*circumference**2}")
        print()
        ask_option = string_check("Would you like to re-start the code?: ", yes_no, 1)

else:
    print()
    print("Thanks for using this calculator, have a good day. 😊")
    print("The program has ended")
```

```
# Imports go here
from math import pi

# Functions go here

def num_check(question, low, hig
    """Checks users enter an int
```

Comprehensive Testing Evidence - Final Program Version

Data Input	**Expected Output** (What should happen)
y	Show the user their instructions
yes	Show the user their instructions
n	Skip Instructions step
no	Skip Instructions step
nah	Tell user to enter the correct word from the list [“yes”, “no”]
yea	Tell user to enter the correct word from the list [“yes”, “no”]
34	Tell user to enter the correct word from the list [“yes”, “no”]

Comprehensive Testing Evidence - Final Program Version

Data Input	**Expected Output** (What should happen)
51	Tell user to enter a whole number between 1 & 50
0	Tell user to enter a whole number between 1 & 50
5.6	Tell user to enter a whole number between {high} & {low}
6	Continue and either ask next question or give answer to perimeter and area
3	Continue and either ask next question or give answer to perimeter and area
hte	Tell user to enter a whole number between {high} & {low}

Discussion

*** NOTE ***

Where you can link the information to a relevant implication e.g. usability, accessibility, functionality

The information that I got while I was planning how my digital outcome would work was good because when I started my planning the planning sheet that I used was not a trello board it was a flowchart that didn't properly show the different components it showed what happened after one thing happened. For me it was quite a good way for me to see how the different components worked together but I had to change it because it was not a proper planning resource. The result of having to do this was negative on my work because it wasted time that I could've used to start writing my code sooner. This was because my flow chart didn't properly show me where I needed to start my code and it didn't have an end point to it meaning that it was just a continuous loop. For example I didn't know what funcion I was going to start with, either the Statement generator, number checker, or string checker and that meant I spent time developing the string checker and number checker at the same time. This was not an efficient use of my time as it meant I had less time to further develop the final program.

Discussion

*** NOTE ***

Where you can link the information to a relevant implication e.g. usability, accessibility, functionality

The information that I got from testing how my digital outcome works was very useful because it helped me to figure out what was wrong with my code and where the different bugs were and what was missing to make my code more advanced and meet the criteria. Testing my program was a good way for me to see how my code actually worked so that I knew what I needed to do to help make it better. The result of the testing my program was an improved final result because I was able to add new functions into my code to make the overall writing smaller and simpler to understand and use.