# `while` !

Loops

# Introducing: **`while`** Loops

- General form of a **`while`** loop statement:

```
while [boolean expression "test"]:
    <repeat block – statements run when test is true>
```

- *Like* an **`if-then`** statement:
  - The test must be a boolean expression
  - if the test evaluates to **`True`**, the computer will move to the first line of code in the repeat block
  - If the test evaluates to **`False`**, the computer will *jump* over the repeat block

- ***Important! Unlike*** an if-then, **after the last statement in the repeat block** completes, the computer will next ***jump backwards*** **to the test** and start anew.

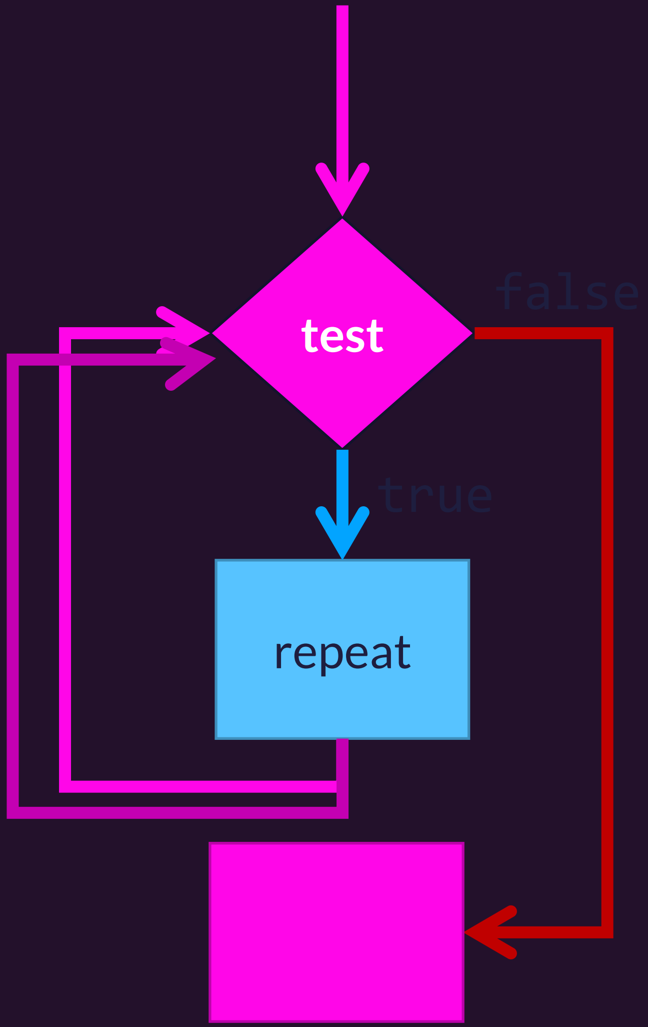
- A **`while`** loop statement can be used *anywhere* you can write a statement.

# `while` loop Flow of Control

1. When a **`while`** statement is encountered, its **`boolean` test** expression is evaluated

2. If the **test** is **`True`**,
   a) then the processor will **proceed into the repeat block.**
   b) **At the end of the repeat block,** the processor jumps back to **step 1**.

3. If the **test** is **`False`**, the processor will jump over the repeat block and continue on.



# Example Setup

In VSCode:

1. Open your COMP110 Workspace
   - File > Open Recent > comp110-workspace

2. Open the File Explorer Pane

3. Create a new Python module in lessons directory
   - Right click lessons
   - Select new file
   - Name it "ls11_while_loop.py"

4. Copy over the program to the right

```python
"""A while loop demo."""

iterations: int = int(input("Loop how many times? "))
i: int = 0
while i < iterations:
    print("In repeat block!")
    print("i is " + str(i))
    i = i + 1

print("After repeat block!")
print("i's terminal value is " + str(i))
```

# Writing a **`while`** loop that iterates a specific number of times.

- Repeating a task a specific number of times is a common task in computing.
  - *Iteration* is the *repetition* of a process
  - You will see this pattern, and variations of it, frequently!
- Three keys:
  1) Initialize a counter variable to 0.
  2) Test will that the counter variable is less than the # of times you want to repeat
  3) **Don't forget!** Incrementing your counter variable.
- **`i`** is an exception to variable name rules
  - Reminder: choose variable names descriptive of their purpose!
  - Why **`i`**? Looong history of being used as a counter variable in computing.

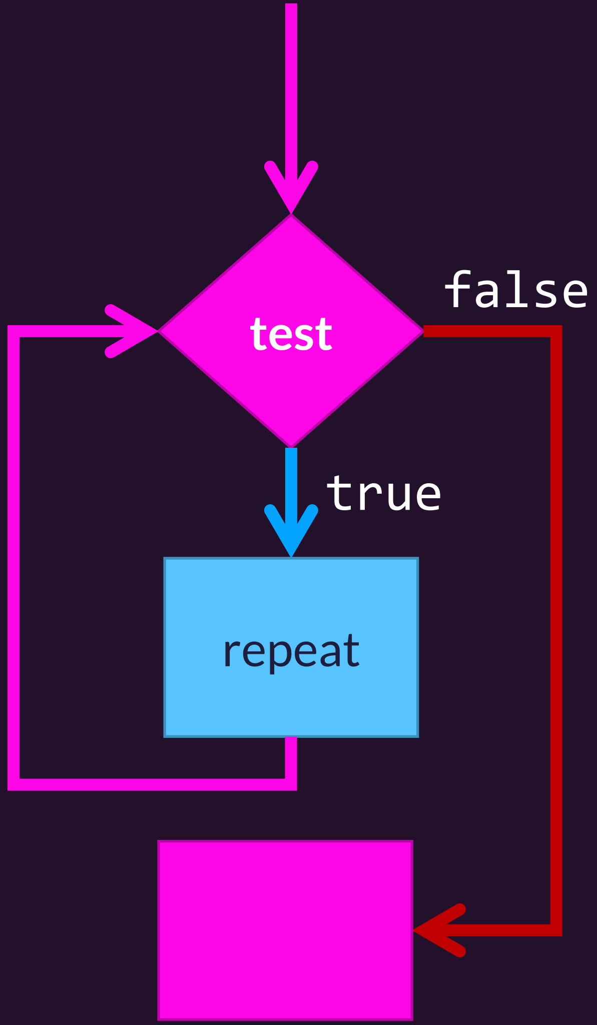
```
i: int = 0  (1)

while i < _(2)_:

    // Do Something Useful

    i = i + 1  (3)
```

# while loop Statement Notes

- If the test is ***not True*** the first time the while loop is encountered,
  then the computer will jump past the repeat block.

- If the test ***never evaluates to False***,
  then the loop is called an ***infinite loop***.

- The only way to *stop* an *infinite* loop is to end your program's process.
  - Press Control+C to send a special "interrupt" signal to your program which should cause it to exit.



## How do you **avoid infinite loops?**

Your **test** condition must eventually evaluate to **`False`**, therefore

a value in the test must be changing inside the repeat block, such that

progress is made toward the test expression evaluating to `False`.

```
i = 0
while i < n:
    print("Loop!")
```

**Bad!** Nothing is changing inside of the repeat block.

```
i = 0
while i < n:
    print("Loop!")
    i = i - 1
```

**Bad!** Subtracting 1 from `i` is not making progress toward `i >=  n`.

```
i = 0
while i < n:
    print("Loop!")
    i = i + 1
```

**Good!** Adding 1 to `i` *is* making progress toward `i >=  n`.