The *while* Loop

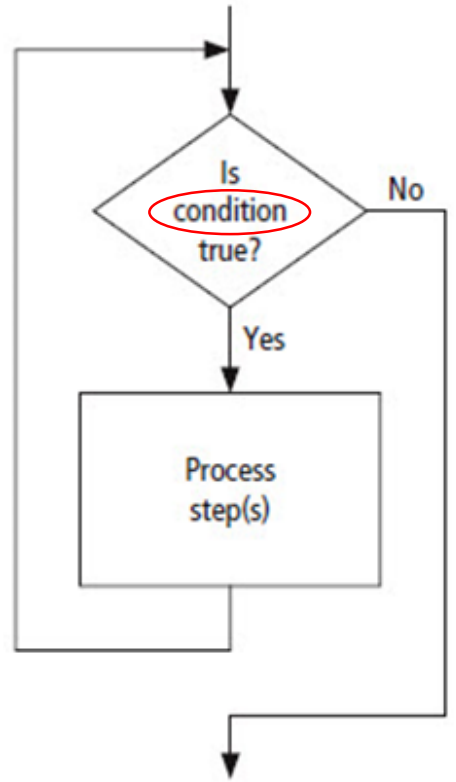
Section 3

Chapter 3

Quiz 6

Repetition Structure

while condition is true
Process step(s)



Recall: conditions

- A *condition* is an expression that evaluates to either `True` or `False`.

- Conditions are used to make decisions
  - Choose between options
  - Control loops

- Conditions typically involve
  - Relational operators (e.g., <, >=)
  - Logical operators (e.g., and, or, not)

Recall: Relational Operators

Python Notation	Numeric Meaning	String Meaning
==	equal to	identical to
!=	not equal to	different from
<	less than	precedes lexicographically
>	greater than	follows lexicographically
<=	less than or equal to	precedes lexicographically or is identical to
>=	greater than or equal to	follows lexicographically or is identical to
in		substring of

```
>>> 5 in [3, 5, 9]
True
```

The *while* Loop

- A repetition structure executes a block of code repeatedly

- A *while* loop repeatedly executes an indented block of statements
  - As long as a certain **condition** is `True`

```
while condition:
    indented block of statements
```

Example 1: Display numbers 1-5

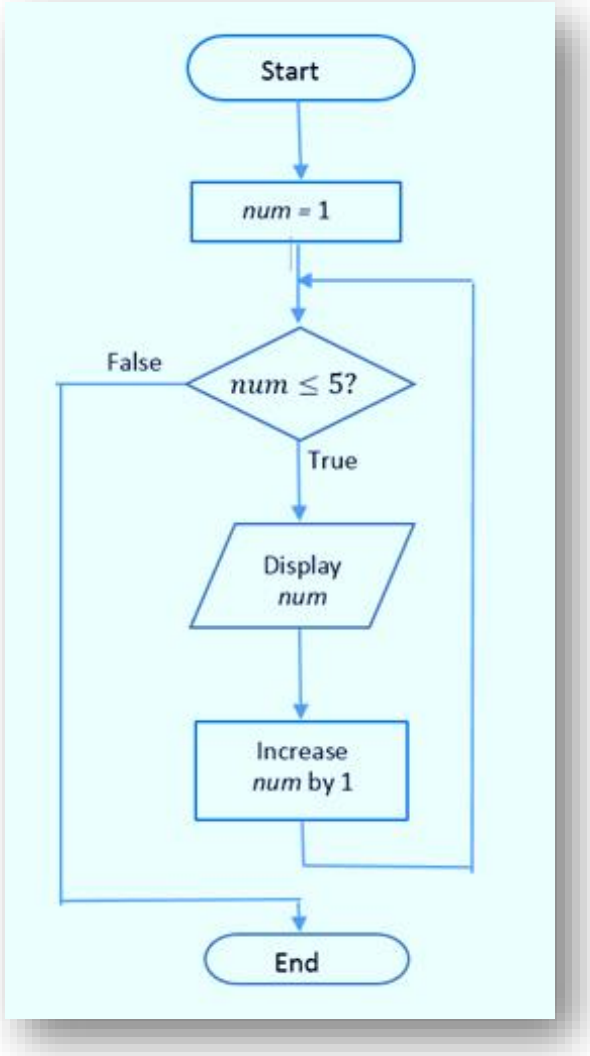
- Displays numbers from 1 to 5.
  - It would be tedious to use print(1), print(2), print(3), …

```
# Display numbers from 1 to 5.

print(1)
print(2)
print(3)
print(4)
print(5)

#It would be super tendious if one wants to display 10,000 numbers...
```

```
1
2
3
4
5
```

  - We may use a repetition structure instead.

Example 1:Display numbers 1-5



Example 2: Display odd numbers <=100

- Display all odd numbers between 1 and 100.
  - Clearly, printing them one by one would not be wise.
  - Use a *while* loop.

```
1    45
3    47
5    49
7    51
9    53
11   55
13   57
15   59
17   61
19   63
21   65
23   67
25   69
27   71
29   73
31   75
33   77
35   79
37   81
39   83
41   85
43   87
     89
     91
     93
     95
     97
     99
```

Example 3: Input validation

- Recall in HW5, we asked user to enter their marriage status using numbers 1 or 2.

- If users entered something other than 1 or 2, we can use a while loop to keep asking users to enter the correct number.

```
Enter your marriage status (1 for single, 2 for married): 3
Enter your marriage status (1 for single, 2 for married): 5
Enter your marriage status (1 for single, 2 for married): 99
Enter your marriage status (1 for single, 2 for married): -1
Enter your marriage status (1 for single, 2 for married): what?!!
Enter your marriage status (1 for single, 2 for married): 2
>>>
```

String methods that return Boolean values

- Given: strings **`str1`** and **`str2`**
  - **`str1.startswith(str2)`**
  - **`str1.endswith(str2)`**

```
>>> "CityU".startswith("City")
True
>>> "CityU".endswith("yu")
False
```

- For determining the type of an item
  - **`isinstance(item, dataType)`**

```
>>> isinstance("CityU", str)
True
>>> isinstance(3.0, int)
False
>>> isinstance([3, 2], list)
True
>>> isinstance((3,2,[5,4]), tuple)
True
```

String methods that return Boolean values

Method	Returns True when
str1.isdigit()	all of *str1*'s characters are digits
str1.isalpha()	all of *str1*'s characters are letters of the alphabet
str1.isalnum()	all of *str1*'s characters are letters of the alphabet or digits
str1.islower()	*str1* has at least 1 alphabetic character and all of its alphabetic characters are lowercase
str1.isupper()	*str1* has at least 1 alphabetic character and all of its alphabetic characters are uppercase
str1.isspace()	*str1* contains only whitespace characters

```
>>> "666".isdigit()
True
>>> "CityU".isalpha()
True
>>> "CityU 2020".isalnum()
False
>>> "CityU2020".isalnum()
True
>>> "99r".islower()
True
>>> "99R".isupper()
True
>>> "     ".isspace()
True
```

Example 4: Input validation for nonnegative integer

- One can use .isdigit() to verify whether the user has entered a nonnegative integer.

```
Enter a nonnegative integer: 5.6
Please enter again. Please enter a nonnegative integer: 0.23
Please enter again. Please enter a nonnegative integer: ajlk4jt
Please enter again. Please enter a nonnegative integer: -3
Please enter again. Please enter a nonnegative integer: -5.6
Please enter again. Please enter a nonnegative integer: 31654.58
Please enter again. Please enter a nonnegative integer: 71
>>>
```

Example 5: Find max, min, average of a sequence of numbers

```
Enter capital 'S' to stop entering numbers:
Enter a number: 56
Enter a number: -12
Enter a number: 9.889
Enter a number: 45.2
Enter a number: -23
Enter a number: S

Max: 56
Min: -23
Average: 15.2178
```

- Strategy:
  - Let the user enter numbers until s/he indicates there is no more number.
  - Put all user-generated numbers in a list.
  - Use built-in functions to find the max, min, and average.

Example 6: Compound interest

- Given an initial balance and a 3% annual interest rate, calculate the number of years for the balance to reach 1 million.

```
Enter the initial deposit: 200000
In 55 years, you will have a million dollars.
```

The *break* Statement

- When **`break`** is executed
  - Loop immediately terminates

- Break statements usually occur under *if* statements

Example 7: display numbers

- In displaying numbers 24-100, suppose you want to **stop** the program as soon as a number is divisible by 11.

```
24
25
26
27
28
29
30
31
32
33
```

The *continue* Statement

- When **`continue`** is executed in a while loop
  - Current iteration of the loop terminates
  - Execution returns to the loop's header

- **`continue`** is usually under an *if* statement.

Example 8: display numbers

- In displaying numbers 1-20, suppose you want to skip all numbers divisible by 3.

```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
```

Infinite Loops

- Infinite loops never ends.

- Can use Ctrl + c to force stop a program while running.

Example 9: infinite loop

```
#Infinite loop

while 1==1:     # The condition is always True.
    print("This ain't gonna end, is it?")

#Use Ctrl + c to terminate programs that are running.
```

```
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
This ain't gonna end, is it?
Traceback (most recent call last):
  File "C:\Users\tzhan7\Desktop\eg.py", line 4, in <module>
    print("This ain't gonna end, is it?")
KeyboardInterrupt
>>>
```

else statement

- An else statement can also be added for while loops.

- The else part is executed once the condition in the while statement is no longer True.

```
1
2
3
4
5
That's all the numbers from 1 to 5.
```

```
# Display numbers from 1 to 5.
num = 1
while num <= 5:
    print(num)
    num += 1
else:    #executed when num>5.
    print("That's all the numbers from 1 to 5.")
```

Classwork 6. Land Utilization

- Hong Kong currently utilizes 7% of its 1114 square kilometres of land for residential purposes. Its breakdown is as follows:

Land Utilization	Area (km^2)	%
Private Residential	27	2.4
Public Residential	17	1.5
Rural Settlement	35	3.1
Other	1035	93
Total	1114	100

- Suppose private residential land will grow by 3% annually, public residential land will increase by a fixed amount annually, and rural settlement land remains constant. Write a Python program to calculate the number of years for the total residential lands to double.
- Let the user enter the annual addition of public residential land (in km^2).
- **Use a *while* loop.**
- The output should resemble the following. Upload the .py file and the output screenshot to Canvas.

```
Enter the annual addition of public residential land (in sq km): 1
It will take at least 34 year(s) for the residential land in Hong Kong to double in size.

Press ENTER to exit.
```