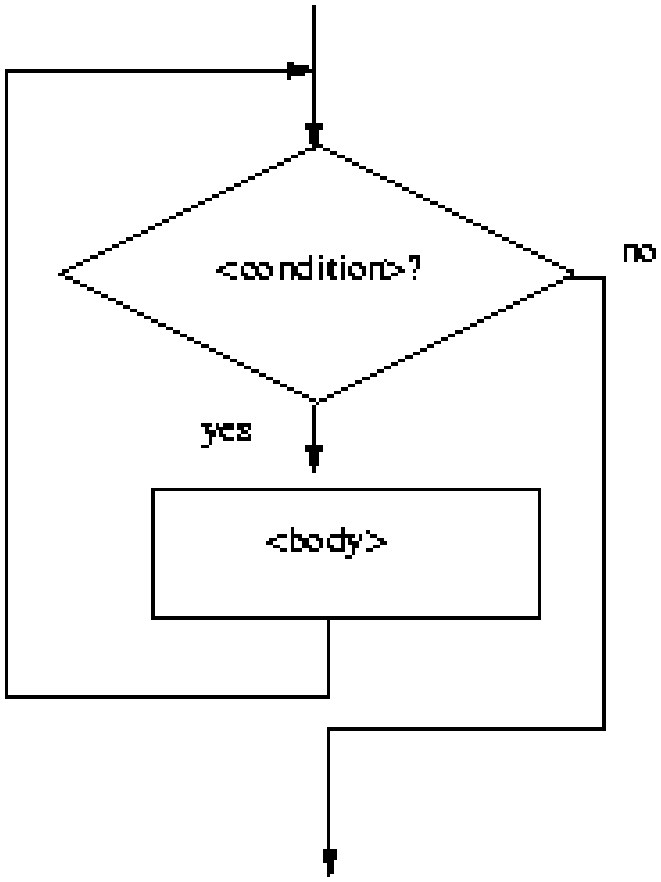
PYTHON: CONTROL STRUCTURES II

Objectives

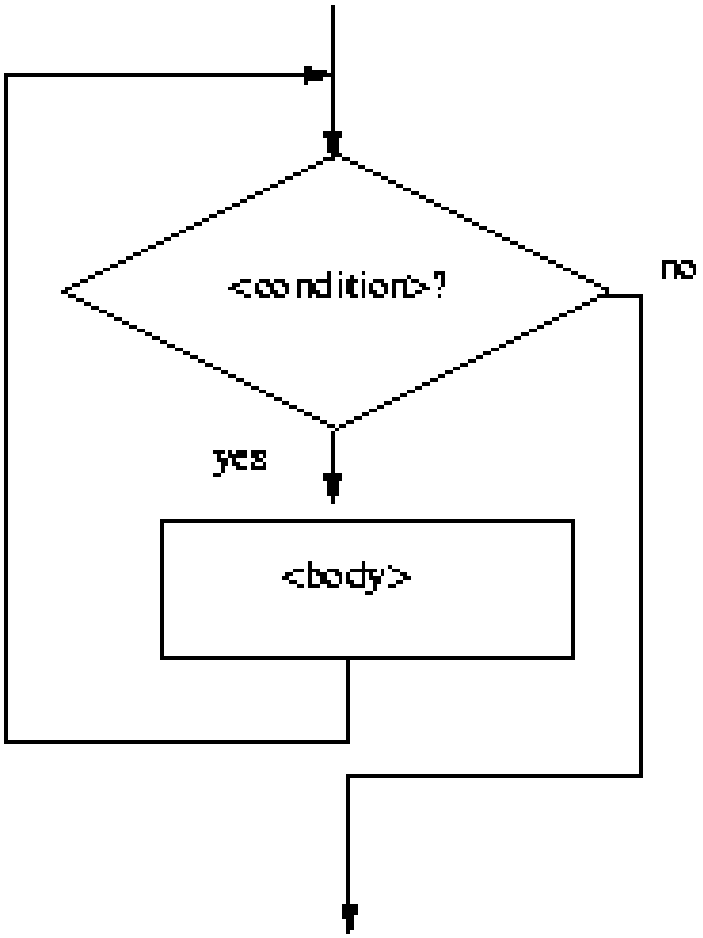
- To understand the applications of `while` loop
- To understand the applications of infinite loop and the role of the `break` and `continue` statements
- To have more practices on definite and indefinite loops

while statement

- `while <condition>:`
  `<body>`
- `condition` is a Boolean expression, just like the one in `if` statements. The body is a sequence of one or more statements.
- Semantically, the body of the loop executes repeatedly as long as the condition remains True. When the condition is False, the loop terminates.



while statement



$$\sum_{current=1}^{n} current = 1 + 2 + .. + n$$

EXERCISE 4.1

Try the code below and figure out what it does. Print out the value of `sum`.

```
sum = 0
current = 1

n = int(input("Enter a value: "))

while current <= n:
    sum = sum + current
    current = current + 1
```

$$\sum_{current=1}^{n} current = 1 + 2 + .. + n$$

EXERCISE 4.2

Modify the code in Exercise 4.1, so that `while current <= n:` now becomes `while current < n:`.

How do you modify other parts of the program to make it work?

Input error checking

The **`while`** statement is well suited for **input error checking** in a program.

```
# Display program welcome
print("This program converts temperatures (Fahrenheit/Celsius)")
print("Enter (F) to convert Fahrenheit to Celsius")
print("Enter (C) to convert Celsius to Fahrenheit")

# Get temperature to convert
f_or_c = input("Please enter selection: ")

while f_or_c != "F" and f_or_c != "C":
    f_or_c = input("Please enter selection again: ")

temp = int(input("Enter temperature to convert: "))
```

EXERCISE 4.3

Integrate the code of the temperature conversion program in Lab 1. Finish the remaining part of the temperature conversion program in the previous slide.

Example Program: Temperature Converter

```
# convert.py
# A program to convert Celsius temps to Fahrenheit
# by: Susan Computewell

def main():
    celsius = eval(input("What is the Celsius temperature? "))
    fahrenheit = 9/5 * celsius + 32
    print("The temperature is", fahrenheit, "degrees Fahrenheit.")

main()
```

EXERCISE 4.4

Extend your code in Exercise 4.3 by asking user for a list of temperatures to convert. Your program will print each temperature's conversion in a new line.

A list can be input simply through

```
list = eval(input(" … "))
```

as comma separated values.

Infinite loop

- An **infinite loop** is an iterative control structure that never terminates (or eventually terminates with a system error).
- Create an infinite loop from the code in Exercise 4.1.
- Implement a mechanism that will restrict a maximum number of loops to a threshold using `break`.
- Executing `break` causes Python to immediately exit the enclosing loop.
- Avoid using `break` too often within loops, because the logic of a loop is hard to follow when there are multiple exits.

`continue` statement

- Run the following program:

```
def main():
    counter = 0
    while counter < 10:
        counter = counter + 1
        if counter == 5:
            continue
        print(counter)

main()
```

- Guess the function of the `continue` statement.

```
# Display program welcome
print("This program converts temperatures
(Fahrenheit/Celsius)")
print("Enter (F) to convert Fahrenheit to Celsius")
print("Enter (C) to convert Celsius to Fahrenheit")

# Get temperature to convert
f_or_c = input("Please enter selection: ")

while f_or_c != "F" and f_or_c != "C":
    f_or_c = input("Please enter selection again: ")

temp = int(input("Enter temperature to convert: "))
```

EXERCISE 4.5

Rewrite the code on slide 7 on Input Error Checking as an infinite loop as follows:

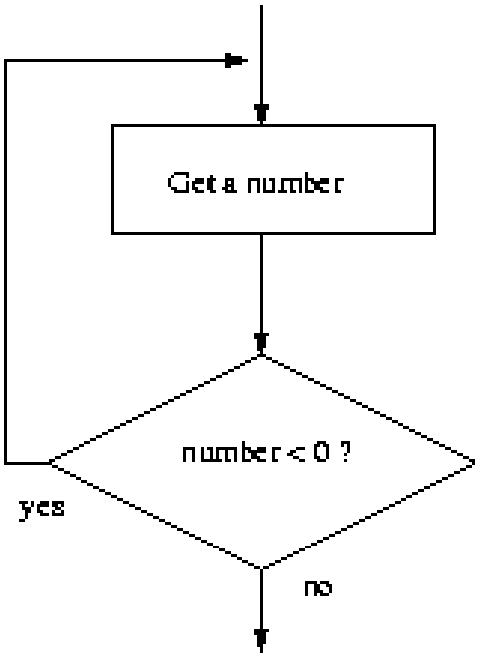
```
while True:
    f_or_c = input("Please enter selection again: ")
    …
```

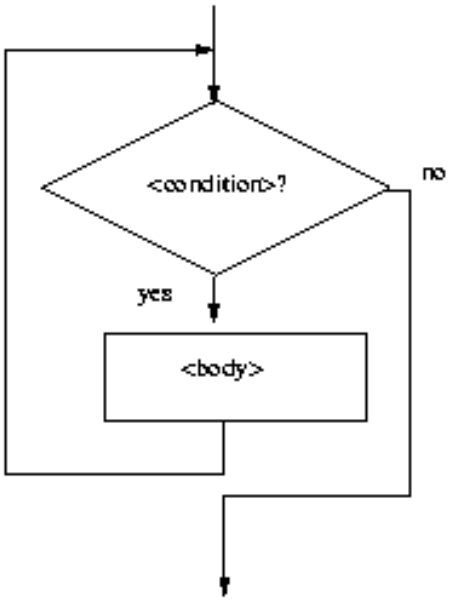
Finish the remaining code and `break` as appropriate.

Post-Test Loop

- Still remember in the lecture the following structure was introduced?

```
repeat
    get a number from the user
until number is >= 0
```





Post-Test Loop

- If the user types an incorrect input, the program asks for another value.
- This process continues until a valid value has been entered.
  - This process is called input validation.
- Python doesn't have a built-in statement to do this, but we can do it with a slightly modified `while` loop.
- We could simulate a post-test loop by

```
number = -1
while number < 1:
    number = eval(input("Enter a positive number:
"))
```

Nested Loops

- Consider the following code:

```
def main():
    for i in range(1, 6):
        for j in range(1, 6):
            print("*", end="");
        print()

main()
```

- What is the aim of it?

Designing Nested Loops

- Break the problem into two parts.
- The first is to process line by line (outer loop).
- The second is to process the items on each line (inner loop).
- Note that the two parts are not dependent.
  - How about the next exercise?

```
def main():
    for i in range(1, 6):
        for j in range(1, 6):
            print("*", end="");
        print()

main()
```

EXERCISE 4.6

Write a program that prints out a symmetric tree. A user will be prompt for a symbol and a positive position for the tip of the leaves. Starting from the tip, the tree will fan out on both sides symmetrically until left side of the leaves reaches the left edge, assuming that the right edge has no limit. The sample input and output are shown in the next slide:

```
Please enter a symbol: *
Please enter a positive position
of the symbol on the first line: 10
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
 *******************
*********************
>>>
```

```
Please enter a symbol: *
Please enter a positive position
of the symbol on the first line: 10
          *
         ***
        *****
       *******
      *********
     ***********
    *************
   ***************
  *****************
 *******************
>>>
```

END