IT Department

Flow Chart and Control Statements

Soma Solaiman Zadeh

Programming I (IT 117)

Fall 2024 - 2025

Week 4

January 8, 2025

Outline

- What is **Algorithm?**
- What is **Flowchart**?
- **Control Structure**
- **Boolean Expressions**
- **Relational Operators** and **Membership Operators**

Algorithms and Flowcharts

- **Algorithms** and **flowcharts** are two different tools that are helpful for creating new programs.
- Both **Algorithms** and **Flowcharts** are used to explain the process of a program step-by-step.
- What is the difference between **Algorithm** and **Flowchart**?

Algorithms

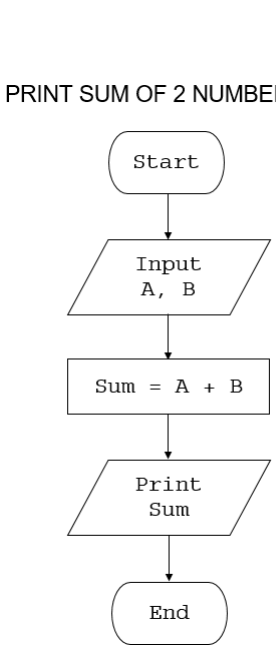
- An **algorithm** is a set of step-by-step instructions to **solve a specific problem** or **perform particular task**.
- Typical stages of an algorithm:
  1. **Input**
  2. **Processing**
  3. **Output**
  4. **Termination**

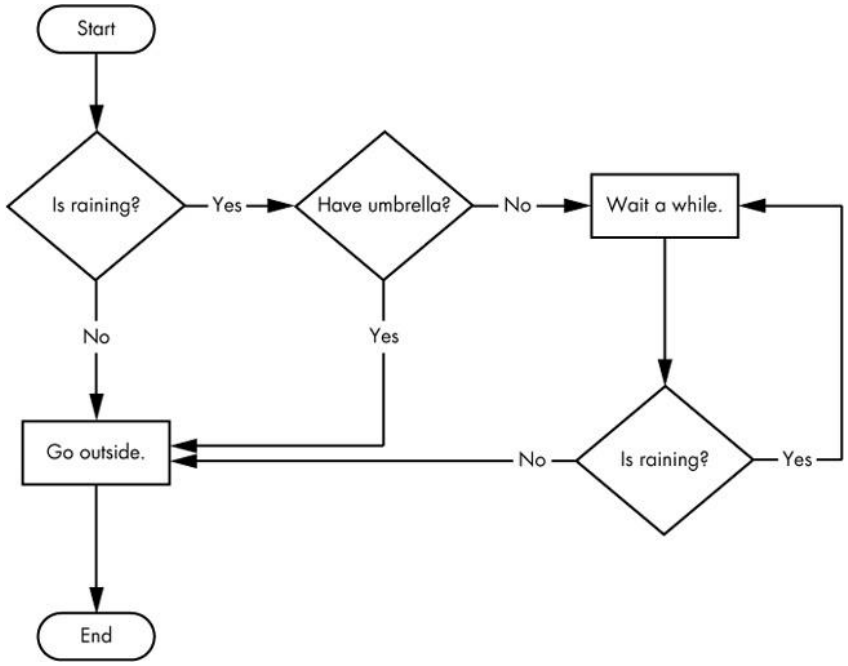
Stages of an Algorithm

1. **Input** → defining the inputs that the algorithm will use.
2. **Processing** → Performing logical and arithmetic calculations on the Input.
3. **Output** → The result of algorithm's processing on input is called output. Output is used to solve the problem or perform the task.
4. **Termination** → Once all the steps are executed successfully and the output is produced, the algorithm reaches its termination point.

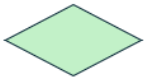
Flowchart

- A **Flowchart** is the **graphical representation of an algorithm** that shows the sequence of steps and decisions needed to perform a process, by using different **symbols**, **shapes** and **arrows**.
- Using flowcharts can show the sequence and logic of each step before writing a computer program.

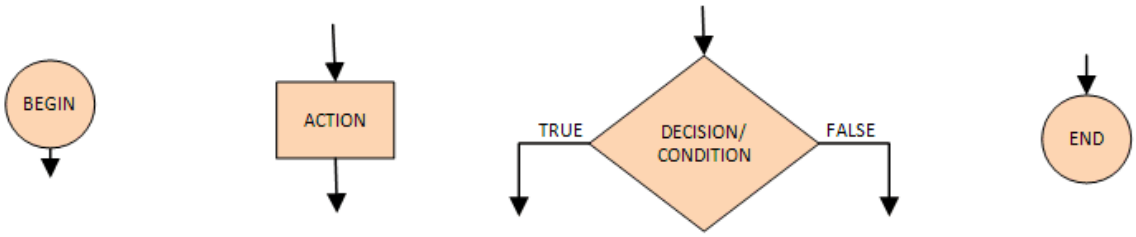




Flowchart Shapes

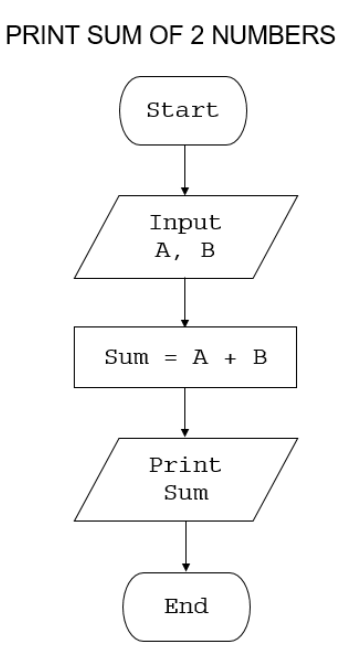
Symbol	Name	Function
	Oval	Represents the start or end of a process
	Rectangle	Denotes a process or operation step
	Arrow	Indicates the flow between steps
	Diamond	Signifies a point requiring a yes/no
	Parallelogram	Used for input or output operations

Flowchart Shapes



Flowchart Example 1

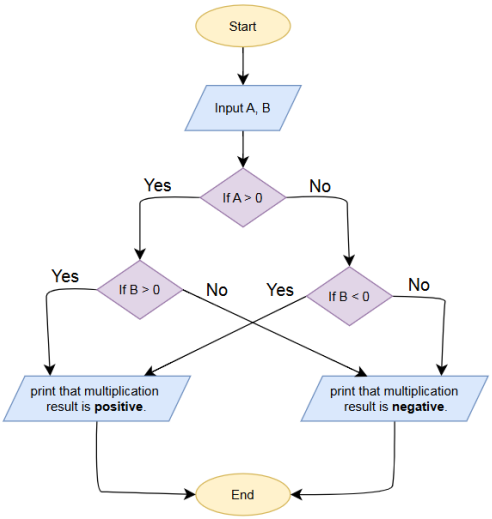
- Draw a flowchart for the process of calculating the summation of two input numbers and print the result.



Flowchart Example 2

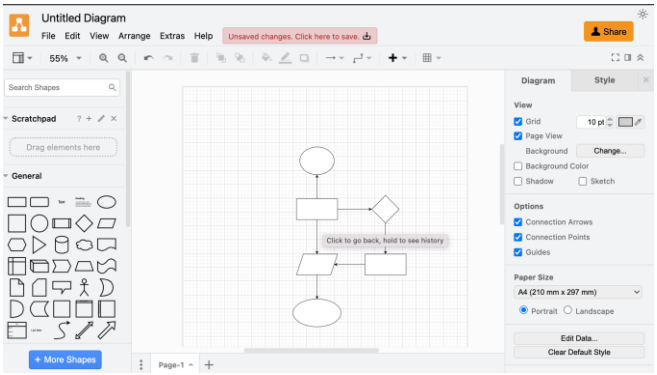
- Draw a flowchart for the process of entering two numbers, and print the **sign** of their multiplication result, either it is positive or negative.

(**Hint –** If both numbers are positive or both are negative, their multiplication result is positive. If one of the two numbers is positive and the other one is negative, their multiplication result is negative.)

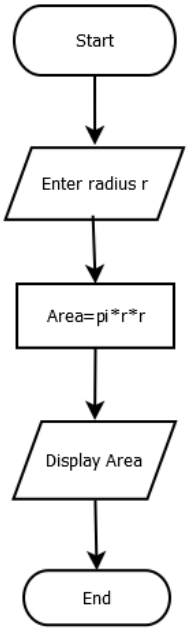


Online Tool for Drawing Flowchart

- "**draw.io**" is a popular online diagramming tool that allowed users to create various types of diagrams, including **flowcharts**.
- Through **draw.io** or **app.diagrams.net,** you have access to the app to create flowcharts.



Flowchart to Python Code



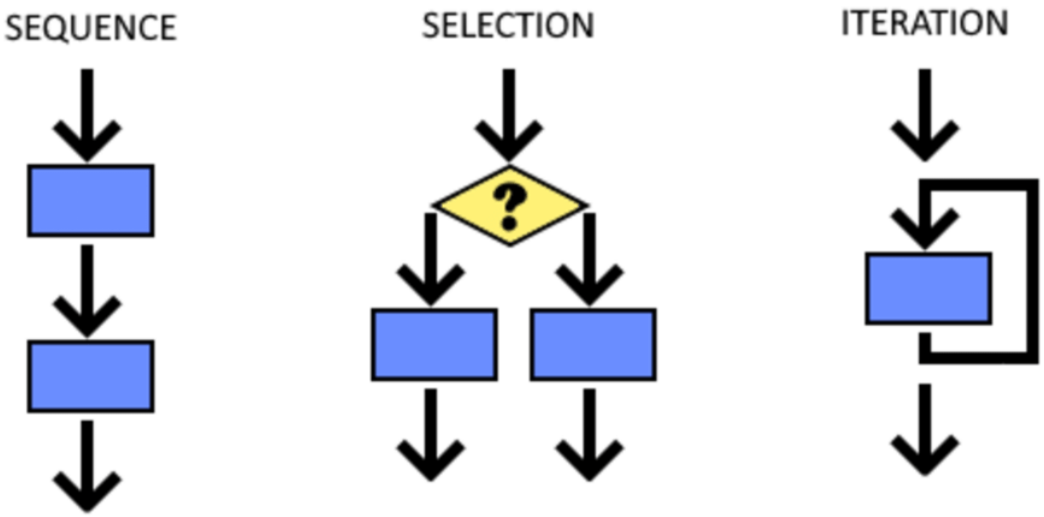

```
r = input ("Please enter radius value of the circle: ")

area = 3.14 * r **2

print (area)
```

- This example shows how we can write a Python code according to a flowchart.
- This flowchart is for calculating the area of a circle according to the radius value entered by the user.

Control Structure

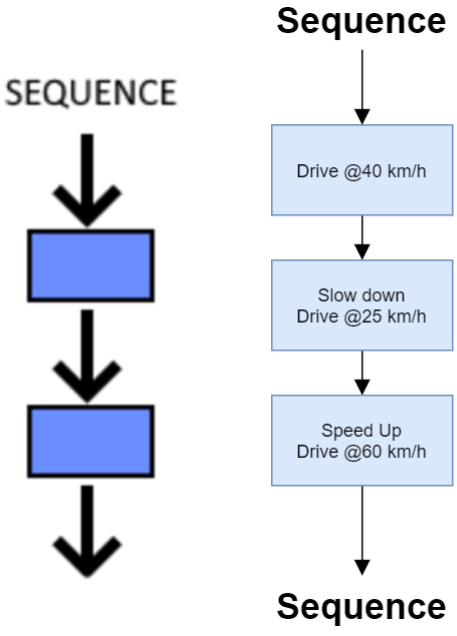
- **Control** Flow is the order that instructions are executed in a program.
- A **control statement** is a statement that determines control flow of a set of instructions.
- There are **three forms of control** that programming languages provide:
  - **Sequential** control
  - **Selection** control
  - **Iterative** control

Tree Forms of Control in Programming



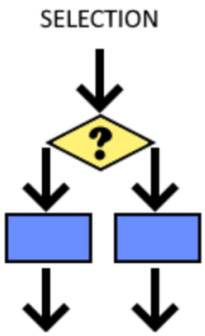
Sequential Control

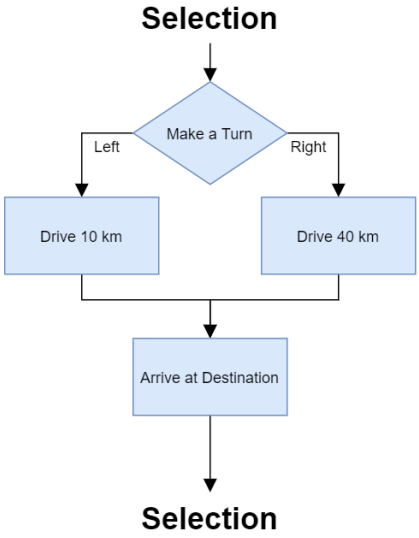
- **Sequential** means “in sequence” or “one-after-the-other”.
- All statements are in the order that we want them to be executed, and the program executes them in sequence from the **Start** statement to the **End** statement.



Selection Control

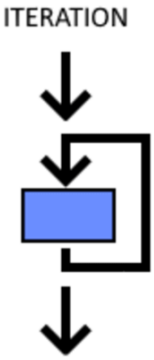
- A **selection control** allows you to make decisions in your code about the current state of your program, and then to choose one of two choices leading to the next statement.

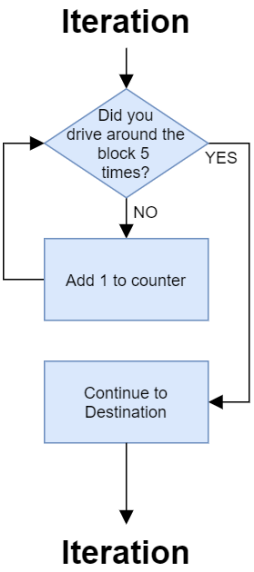




Iterative Control

- An **iterative control** statement executes a sequence of statements multiple times, based on a condition or set of conditions.





Boolean Expressions

- We already mentioned **Boolean** data type as one of the basic data types. **Boolean** data type **contains two Boolean values**, denoted as **True** and **False** in Python.

- A **Boolean expression** is an expression **that has a Boolean value**.
- **Boolean expressions** are used to represent the conditions for **selection** and **iterative** control statements.

Relational Operators in Python

- The **relational operators (comparison operators)** in Python perform the usual **comparison operations**.

- Relational expressions are a type of **Boolean expression** since they have a **Boolean** result.

Relational Operators (Comparison Operators)

Operator	Description	Example	Example's Result
<	Less than	6 < 4	False
>	Greater than	9 > 5	True
<=	Less than or equal to	8 <= 12	True
>=	Greater than or equal	10 >= 15	False
= =	Equal to	7 == 9	False
!=	Not equal to	7 != 9	True

Let's Try it!

- What is the result value of each relational expression?

10 == 20 → **False**

10 != 20 → **True**

10 <= 20 → **True**

Relational Operators on String Values

- Can we compare two strings by using relational operators? **YES.**
- We can compare two strings using their **ASCII code**.
- The **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange, or **ASCII**, is a character encoding standard.
  - Using **ASCII code** of characters helps representing text in digital form.

ASCII Code of Characters

Character	ASCII Code
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Character	ASCII Code
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Character	ASCII Code
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Relational Operators on String Values - Examples

- We compare two strings based on the ASCII code of their first character. If their first character is the same, the ASCII code of the second character is compared, and so on.

'Football' <= 'Swimming'	→	True
'Basketball' < 'Boxing'	→	True
'Badminton' >= 'Basketball'	→	False
'Cycling' != 'Handball'	→	True
'Swimming' == 'Boxing'	→	False

Finding ASCII Code of a Character in Python

- You can use **ord( )** function to know the ASCII code of any given character.
- The **ord( )** function requires the parameter in the form of a **string** that has a length of one (only one character).
- The function returns an integer, which is the ASCII code for the string that is passed in the function.
- **Example –**

```
ch = 'q'

code = ord(ch)

print("The ASCII code of" , ch , "is" , code)
```

Output → The ASCII code of q is 113

Membership Operators

- **Membership operators** are used to check whether a value or variable exists in a sequence or not.
- There are two membership operators: **in** , **not in**

Membership Operators	Examples	Result
in	10 **in** (10, 20, 30)	True
	'red' **in** ('red', 'green', 'blue')	True
not in	10 **not in** (10, 20, 30)	False

Membership Operators

- The membership operators can also be used to check if a given character or string occurs within another string,

'Good' **in** 'Good Morning' → **True**
'M' **in** 'Good Morning' → **True**
'm' in 'Good Morning' → **False**

- The **membership operators**, like relational operators, can be used to construct Boolean expressions.

Let's Try it!

- What is the result value of each expression?

10 **in** (40 , 20 , 10) → **True**

10 **not in** (40 , 20 , 10) → **False**

grade = 'A'

grade **in** ('A' , 'B' , 'C' , 'D') → **True**

city = 'Zaxo'

city **not in** ('Erbil' , 'Sulaymaniah' , 'Duhok') → **True**