

SELECTION WITH IF (MAY14)

Overview

- so far we've learned how to write basic simple programs (SIPO)
- it is often the case the processing of input depends on the situation (properties of input),
- we should be able to select what code to execute
- hence today we will learn if statement enabling selection in our codes

if: a simple example

Check if a number is negative or positive, and accordingly output a message.

- SIPO statements are insufficient: processing depends on input
- Hence we need a selection statement: **if**

if: a simple example

Check if a number is negative or positive, and accordingly output a message.

- SIPO statements are insufficient: processing depends on input
- Hence we need a selection statement: **if**

```
value = eval(input("Enter an integer (e.g, 23 or -118): "))
if value < 0:
    print(value, "is negative")
else:
    print(value, " is positive")
```

- evaluate **value < 0**
- if **True** execute first **print**
- if **False** skip first **print**, execute second **print**
- Either case, only one of **print** statements executed!
- Hence **if** selects what to execute based on a test

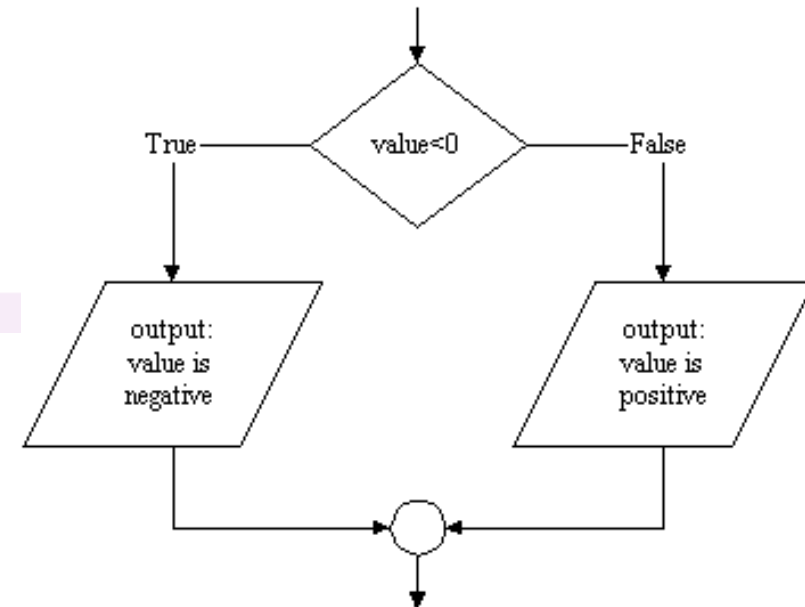
if: a simple example

Check if a number is negative or positive, and accordingly output a message.

- SIPO statements are insufficient: processing depends on input
- Hence we need a selection statement: **if**

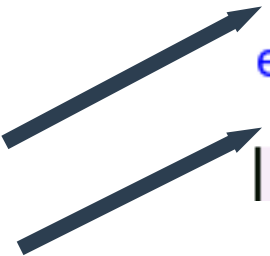
```
value = eval(input("Enter an integer (e.g, 23 or -118): "))  
if value < 0:  
    print(value, "is negative")  
else:  
    print(value, " is positive")
```

- evaluate **value < 0**
- if **True** execute first **print**
- if **False** skip first **print**, execute second **print**
- Either case, only one of **print** statements executed!
- Hence **if** selects what to execute based on a test



if: a simple example

```
value = eval(input("Enter an integer (e.g, 23 or -118): "))  
if value < 0:  
    print(value, "is negative")  
else:  
    print(value, " is positive")
```



Note that the statements after the if and else statements are indented four spaces.

An improvement

What is the output for
input '0'?

```
value = eval(input("Enter an integer (e.g, 23 or -118): "))  
if value < 0:  
    print(value, "is negative")  
else:  
    print(value, " is positive")
```

An improvement

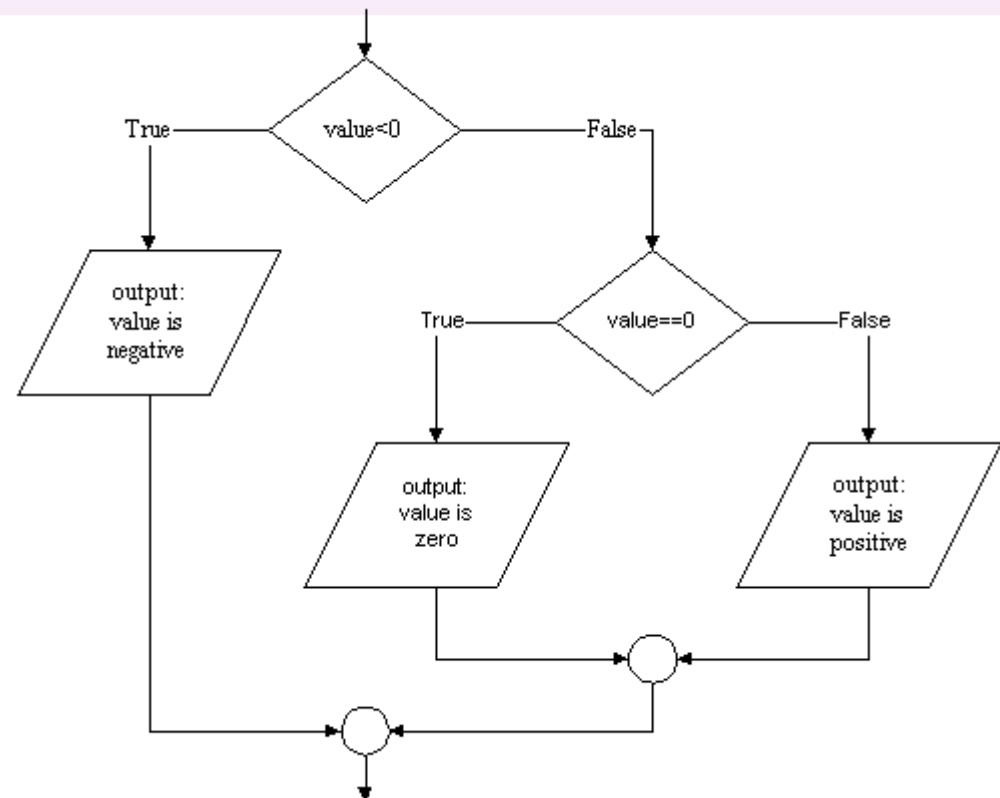
0 is not positive!

hence consider 3 cases:

1. value is negative
2. value is 0
3. value is positive

Consider the 3rd possibility via ->

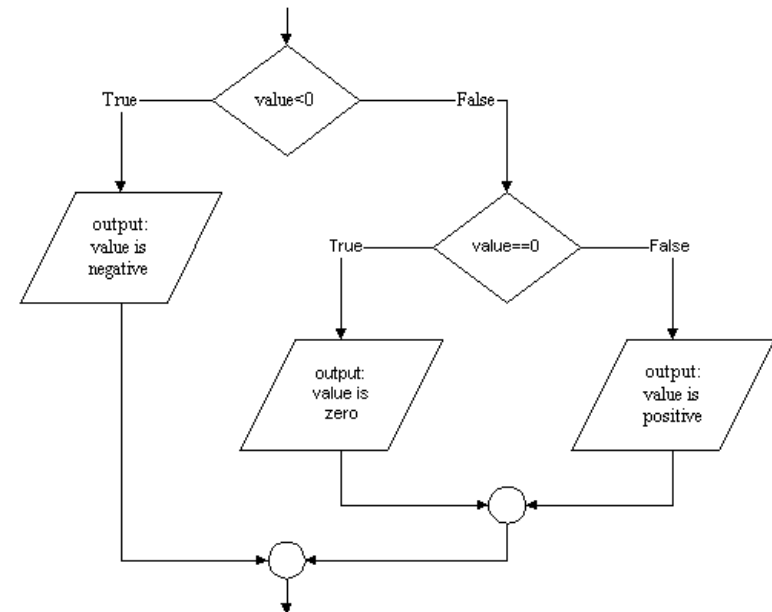
```
value = eval(input("Enter an integer (e.g, 23 or -118): "))
if value < 0:
    print(value, "is negative")
else:
    print(value, " is positive")
```



An improvement: nested if statements

```
value = eval(input( "Enter an integer (e.g. 23 or -118): " ))
if value < 0:
    print(value, "is negative")
else:
    if value == 0:
        print(value, "is neither positive nor negative")
    else:
        print(value, "is positive")
```

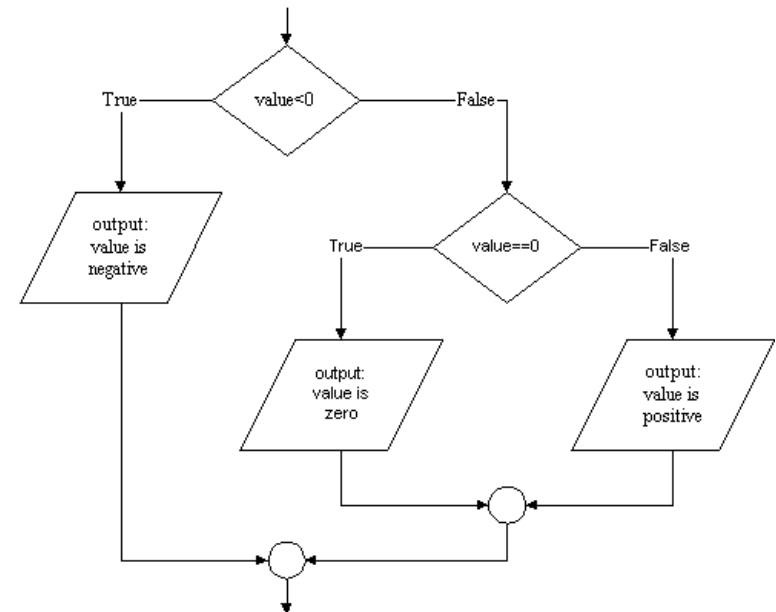
- Hence, we can nest 2 **if** statements
- Note the '==' operator
- Note the second level of indentation



An improvement: nested if statements

```
value = eval(input( "Enter an integer (e.g. 23 or -118): " ))
if value < 0:
    print(value, "is negative")
else:
    if value == 0:
        print(value, "is neither positive nor negative")
    else:
        print(value, "is positive")
```

- Hence, we can nest 2 **if** statements
- Note the '==' operator
- Note the second level of indentation



Multiway decision: elif

- Multiway decision (as opposed to two-way) is very common, so Python provides a special syntax: **elif**

```
value = eval(input( "Enter an integer (e.g, 23 or -118): " ))
if value < 0:
    print(value, "is negative")
elif value == 0:
    print(value, "is neither positive nor negative")
else:
    print(value, "is positive")
```

elif combines else & if

if syntax

- Evaluates expressions one by one until one is found to be **True**
- Then executes corresponding statement
- Rest of **if** is skipped!!!
- if all expressions are false, executes else section & false statement

```
if test-expression1:  
    true-statement1  
[elif test-expression2:  
    true-statement2]  
  
...  
[else:  
    false-statement]
```

if syntax

- Evaluates expressions one by one until one is found to be **True**
- Then executes corresponding statement
- Rest of **if** is skipped!!!
- if all expressions are false, executes else section & false statement

```
value = eval(input( "Enter an integer (e.g. 23 or -118): " ))  
  
if value < -100:  
    print(value, "is very negative")  
elif value < 0:  
    print(value, "is negative")  
elif value == 0:  
    print(value, "is neither positive nor negative")  
else:  
    print(value, "is positive")
```

What is the output
for inputs
“-20” and “-101”

Relational Expressions

operator	description
<	less than
<=	less than or equal to
==	equal
!=	not equal
>=	greater than or equal to
>	greater than

try in Python shell!

Relational Expressions

Python also provides logical operators that can be combined with relational operators:

```
age >= 20 and age < 30
```

```
age >= 20 or income < 18.45|
```

```
In [54]: not True
```

```
Out[54]: False
```

```
In [55]: not False
```

```
Out[55]: True
```

x	not x
0	1
1	0

x	y	x and y	x or y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

Example: A single program to convert F to C or C to F

```
print("This program converts temperatures from Fahrenheit to Celsius,")
print("or from Celsius to Fahrenheit.")
print("Choose")
print("1 to convert Fahrenheit to Celsius")
print("2 to convert Celsius to Fahrenheit")

choice = eval(input("Your choice? "))

if choice == 1:
    print("This program converts temperatures from Fahrenheit to Celsius.")
    temp_in_f = eval(input("Enter a temperature in Fahrenheit (e.g. 10) and press Enter: "))
    temp_in_c = (temp_in_f - 32.0)*5.0/9.0
    print(temp_in_f, " degrees Fahrenheit = ", temp_in_c, " degrees Celsius.")

elif choice == 2:
    print("This program converts temperatures from Celsius to Fahrenheit .")
    temp_in_c = input("Enter a temperature in Celsius (e.g. 10) and press Enter: ")
    temp_in_f = temp_in_c *9.0/5.0 + 32.0
    print(temp_in_c, " degrees Celsius = ", temp_in_f, " degrees Fahrenheit.")

else:
    print("Error: Your choice not recognized!")
```


Case Study: Converting numerical grades to letter grades

Write a program that accepts numerical grade and outputs letter grade

Numerical Grade	Letter Grade
95-100	A+
86-94	A
80-85	A-
75-79	B+
70-74	B
65-69	B-
62-64	C+
58-61	C
55-57	C-
50-54	D
0-50	F

Case Study: Converting numerical grades to letter grades

```
# Version 1.
print " ... " # Instructions to user
grade = eval(input("..."))
if grade >= 95 and grade <= 100:
    print("A+")
if grade >= 86 and grade <= 94:
    print("A")
if grade >= 80 and grade <= 85:
    print("A-")
# ...
# and so on down to
# ...
if grade >= 50 and grade <= 54:
    print("D")
if grade >= 0 and grade <= 50:
    print("F")
if grade < 0 or grade > 100:
    print("Error: The grade must be between 0 and 100.")
```

Case Study: Converting numerical grades to letter grades (version 2)

```
# Version 2.
print(" ... ") # Instructions to user
grade = eval(input(" ... "))
if grade >= 95 and grade <= 100:
    print("A+")
elif grade >= 86 and grade <= 94:
    print("A")
elif grade >= 80 and grade <= 85:
    print("A-")
# ...
# and so on down to
# ...
elif grade >= 50 and grade <= 54:
    print("D")
elif grade >= 0 and grade <= 50:
    print("F")
else:
    print("Error: The grade must be between 0 and 100.")
```

Case Study: Converting numerical grades to letter grades (version 3)

```
# Version 3.  
print(" ... ") # Instructions to user  
grade = eval(input("..."))  
if grade > 100 or grade < 0:  
    print("Error: The grade must be between 0 and 100.")  
elif grade >= 95:  
    print("A+")  
elif grade >= 86:  
    print("A")  
elif grade >= 80:  
    print("A-")  
# ...  
# and so on down to  
# ...  
elif grade >= 50:  
    print("D")  
elif grade >= 0:  
    print("F")
```

Case Study: Testing

- Let's test this code for a few grades:

99, 13, 27

- What values do you think we should choose for testing?

```
# Version 3.  
print(" ... ") # Instructions to  
grade = eval(input("..."))  
if grade > 100 or grade < 0:  
    print("Error: The grade must  
elif grade >= 95:  
    print("A+")  
elif grade >= 86:  
    print("A")  
elif grade >= 80:  
    print("A-")  
# ...  
# and so on down to  
# ...  
elif grade >= 50:  
    print("D")  
elif grade >= 0:  
    print("F")
```

Case Study: Testing

- Let's test this code for a few grades:

99, 13, 27

- Programs usually break due to uncommon values and unexpected inputs

- * 75.3 (float input)
- * edge cases: 0, 100
- * out of range inputs: -1, 101 etc
- * garbage input: can program handle it?

```
# Version 3.  
print(" ... ") # Instructions to  
grade = eval(input("..."))  
if grade > 100 or grade < 0:  
    print("Error: The grade must  
elif grade >= 95:  
    print("A+")  
elif grade >= 86:  
    print("A")  
elif grade >= 80:  
    print("A-")  
# ...  
# and so on down to  
# ...  
elif grade >= 50:  
    print("D")  
elif grade >= 0:  
    print("F")
```

Case Study: Performance Analysis

	Number of comparison & logical operations		
	Minimum	Average	Maximum
Version 1	12 if/elifs @ 3 = 36	12 if/elifs @ 3 = 36	12 if/elifs @ 3 = 36
Version 2	3	(12 if/elifs @ 3)/2 = 18	12 if/elifs @ 3 = 36
Version 3	3	((1 if @ 3) + (11 if/elifs @ 1))/2 = 7	((1 if @ 3) + (11 if/ellifs @ 1)) = 14

- further improvement: check for the most likely grades first
does performance gain justify a less readable code?
- above performance analysis involves overestimation
as Python 'short circuits' test expressions when it can

Summary: selection using if

```
if test-expression:  
    true-statement
```

```
if test-expression:  
    true-statement  
else:  
    false-statement
```

```
if test-expression:  
    true-statement  
elif test-expression:  
    true-statement  
...  
else:  
    false-statement
```

- three ways of using **if**
- **else** is usually used for catching errors or invalid values!

Programming Exercise: Gazinta (as in "2 Gazinta 8, but 3 doesn't")

sample runs:

```
Wondering if one number "goes into" another?  
Give me the numbers (big one first) and I'll tell you.  
First number = 8  
Second number = 2  
2 does go into 8 exactly.
```

```
Wondering if one number "goes into" another?  
Give me the numbers (big one first) and I'll tell you.  
First number = 8  
Second number = 3  
3 does NOT go into 8 exactly.
```

```
Wondering if one number "goes into" another?  
Give me the numbers (big one first) and I'll tell you.  
First number = 8  
Second number = 0  
Sorry division by 0 is undefined!
```

Problem:

Write a program that accepts two numbers, and outputs a message telling the user whether the second number divides exactly into the first or not.

Programming Exercise:

Gazinta (as in "2 Gazinta 8, but 3 doesn't")

```
1 # gazinta.py -- this code takes in two numbers, and determines
2 #   one divides the first one exactly.
3 # S. Bulut May 2019
4
5 print("""Wondering if one number "goes into" another?
6 Give me the numbers (big one first) and I'll tell you.""")
7
8 a = eval(input("First number = "))
9
10 b = eval(input("Second number ="))
11
12 if b == 0:
13     print("Sorry division by 0 is undefined!")
14 elif a%b == 0:
15     print(b, " does go into", a, "exactly")
16 else:
17     print(b, " does NOT go into", a, "exactly")
```

Programming exercise: Pythagorean theorem

Problem:

Write a program that accepts three integer values, and outputs a message stating whether they could be the sides of a right-angled triangle or not.

```
Enter the lengths of the sides of your triangle and I will tell you
if it has a right angle or not.
Length of first side = 3
Length of second side = 4
Length of third side = 5
That is a right-angled triangle.
```

```
Enter the lengths of the sides of your triangle and I will tell you
if it has a right angle or not.
Length of first side = 3
Length of second side = 5
Length of third side = 4
That is a right-angled triangle.
```

```
Enter the lengths of the sides of your triangle and I will tell you
if it has a right angle or not.
Length of first side = 3
Length of second side = 4
Length of third side = 6
That is NOT a right-angled triangle.
```

```
Enter the lengths of the sides of your triangle and I will tell you
if it has a right angle or not.
Length of first side = 4
Length of second side = 4
Length of third side = 4
That is NOT a right-angled triangle.
```

Programming exercise: Pythagorean theorem (Solution)

```
# pythagoras.py
# S. Bulut 2019
# T. Topper 2015

print('''
=====
Right-angle triangle tester
-----

Enter the lengths of the sides of your triangle and I will let you know
if it has a right angle or not
''')

a = eval(input("Length of the first side ="))
b = eval(input("Length of the second side ="))
c = eval(input("Length of the third side ="))

if a**2 + b**2 == c**2 or a**2 + c**2 == b**2 or b**2 + c**2 == a**2:
    print("That is a right angled triangle")
else:
    print("That is NOT a right angled triangle.")
```

Programming exercise: Pythagorean theorem (Solution)

```
# pythagoras_b.py
# S. Bulut 2019
# T. Topper 2015

print('''
=====
Right-angle tirangle tester
-----

Enter the lengths of the sides of your triangle and I will let you know
if it has a right angle or not
''')

a = eval(input("Length of the first side ="))
b = eval(input("Length of the second side ="))
c = eval(input("Length of the third side ="))

# find hypoteneuse
hypo = a
if b > hypo:
    hypo = b

if c > hypo:
    hypo = c

if a**2 + b**2 + c**2 == 2*hypo**2:
    print("That is a right angled triangle")
else:
    print("That is NOT a right angled triangle.")
```

Programming Exercise: Utility Bills

Problem:

A utility company takes readings each month and charges its customers according to the table shown below.

Write a program to calculate a customer's bill amount given the number of kilo-watt hours used.

kWh	Rate
Under 500	\$20.00
500 to 1000	\$20.00 + \$0.03 per kWh over 500
Over 1000	\$35.00 + 0.02 per kWh above 1000

```
=====
                        Bill calculator
                        -----
Enter kilowatt-hours used: 300
Bill amount = $20.00
```

```
=====
                        Bill calculator
                        -----
Enter kilowatt-hours used: 600
Bill amount = $23.00
```

```
=====
                        Bill calculator
                        -----
Enter kilowatt-hours used: 1200
Bill amount = $39.00
```

Programming Exercise: Utility Bills (Solution)

```
# utilitybill.py: calculates utility bill for a give consumption in kWh
# S. Bulut 2019
# T. Topper 2015

print('''
=====
Bill Calculator
-----
''')

kwh = eval(input("Enter kWh used: "))
if kwh < 0:
    print('\nRecheck input value!')
    print("It was negative and meters\ndon't run backwards")
else:
    if kwh <= 500.0:
        amount = 20.0
    elif kwh <= 1000.0:
        amount = 20.0 + 0.03*(kwh-500.)
    else:
        amount = 35.0 + 0.02*(kwh-1000.)
```