



fit@hcmus

VNUHCM - UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

University of Science - VNU-HCM
Faculty of Information Science
Department of Computer Science

MTH083 - Advanced Programming for Artificial Intelligence

Slot 02- Condition and Repetition

Advisor:

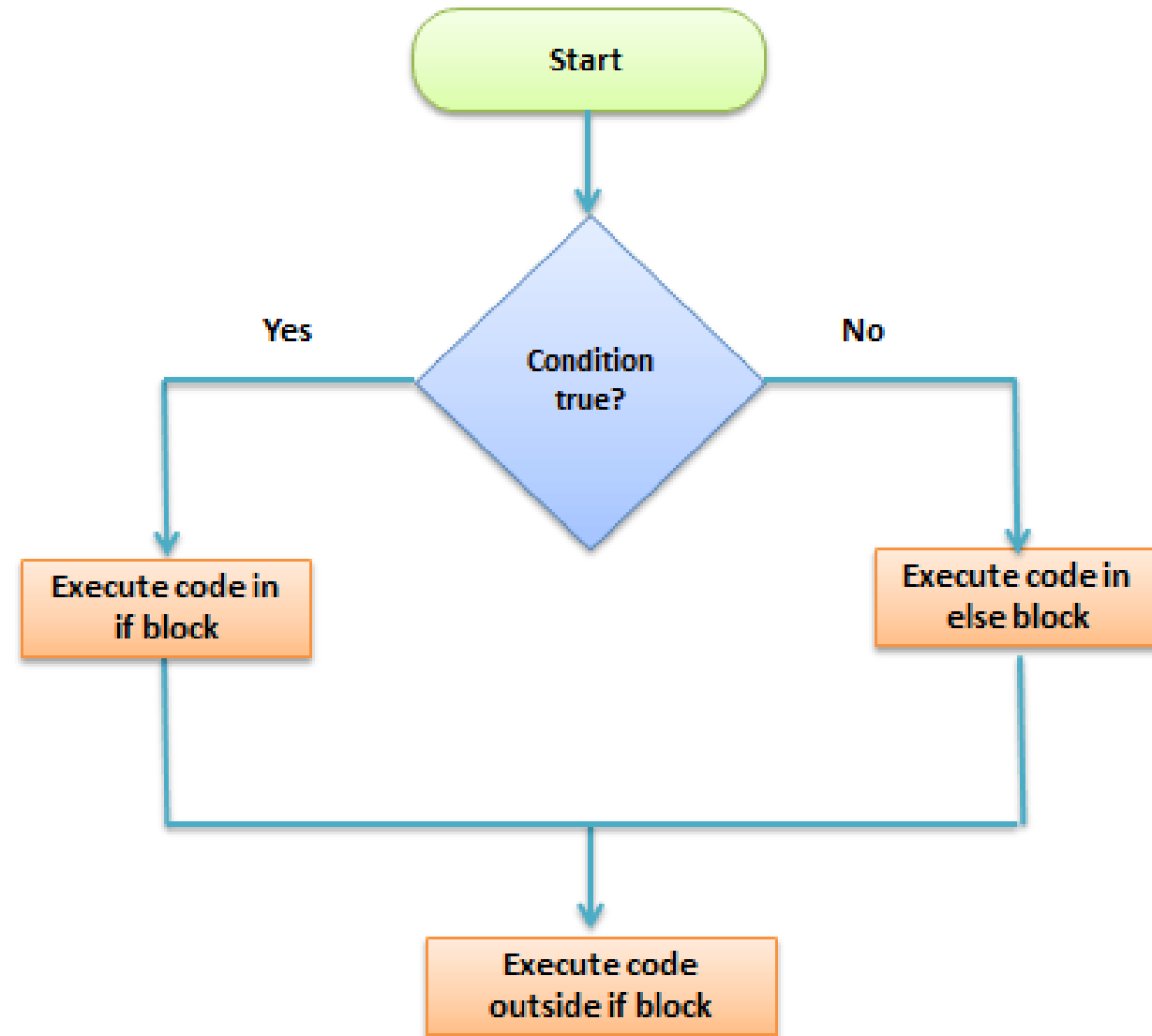
Dr. Nguyễn Tiến Huy

Dr. Lê Thanh Tùng

- 1 Condition
- 2 Repetition
- 3 Try/Except

Part 1: Condition

- Condition or selection, need to detect:
- A condition
- True action
- False action (optional)



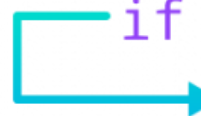
- Obtain an integer score. If score is bigger than 5.0, display "Pass", otherwise "Not pass". Draw a flowchart to solve this problem

- In python, condition is represented by:

```
if condition:  
    # body of if statement
```


Condition is True

```
number = 10  
if number > 0:  
    # code  
  
# code after if
```



Condition is False

```
number = -5  
if number > 0:  
    # code  
  
# code after if
```

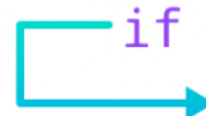


Condition: if ... else

```
if condition:  
    # block of code if condition is True  
  
else:  
    # block of code if condition is False
```

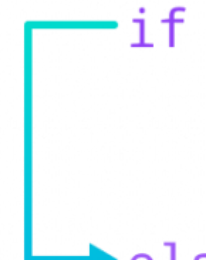
Condition is True

```
number = 10  
if number > 0:  
    # code  
  
else:  
    # code  
  
# code after if
```



Condition is False

```
number = -5  
if number > 0:  
    # code  
  
else:  
    # code  
  
# code after if
```



- Obtain an integer score. If score is bigger than 5.0, display "Pass", otherwise "Not pass"
- Draw a flowchart

- In python, condition is represented by:
- Input: int x
- Output: Pass, Not Pass
- Condition: $x > 5.0$
- True statement: print – Pass
- False statement: print – Not pass

Python

```
if <expr>:  
    <statement(s)>  
else:  
    <statement(s)>
```

Python

```
if <expr>:  
    <statement(s)>  
else:  
    <statement(s)>
```

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

- Indentation is a way to separate the statement's block in Python

Example

If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

- Write a program to calculate the absolute value of integer n

- Write a program to find the biggest number between two integers from keyboards

Condition: if ... elif ... else

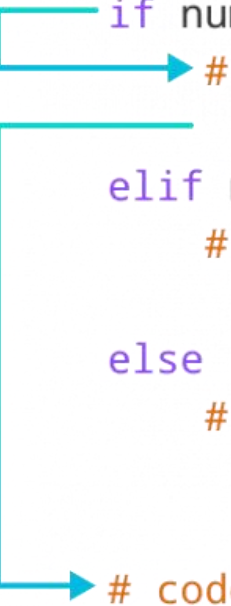
```
if condition1:  
    # code block 1
```

```
elif condition2:  
    # code block 2
```

```
else:  
    # code block 3
```

1st Condition is True

```
let number = 5  
if number > 0 :  
    # code  
  
elif number < 0 :  
    # code  
  
else :  
    # code  
  
# code after if
```



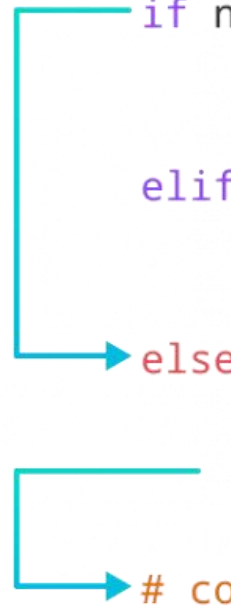
2nd Condition is True

```
let number = -5  
if number > 0 :  
    # code  
  
elif number < 0 :  
    # code  
  
else :  
    # code  
  
# code after if
```



All Conditions are False

```
let number = 0  
if number > 0 :  
    # code  
  
elif number < 0 :  
    # code  
  
else :  
    # code  
  
# code after if
```



- For statements in condition2: it also meet the condition1
- Can be rewritten as:
- if (condition1 and condition2):

```
# outer if statement
if condition1:
    # statement(s)

    # inner if statement
    if condition2:
        # statement(s)
```

- Write a program to input a positive integer n . Check whether n is a squared number or not? (A squared number is a number that when taking the square root of 2, the result is an integer)

- Write a function that checks for a leap year (leap year: divisible by 4 and not divisible by 100 or divisible by 400)

- Enter 1 month and year, show how many days that month (month has 31 days: 1, 3, 5, 7, 8, 10, 12; month has 30 days: 4, 6, 9, 11; February have 28 or 29 days)

- Print out these 3 numbers in ascending order

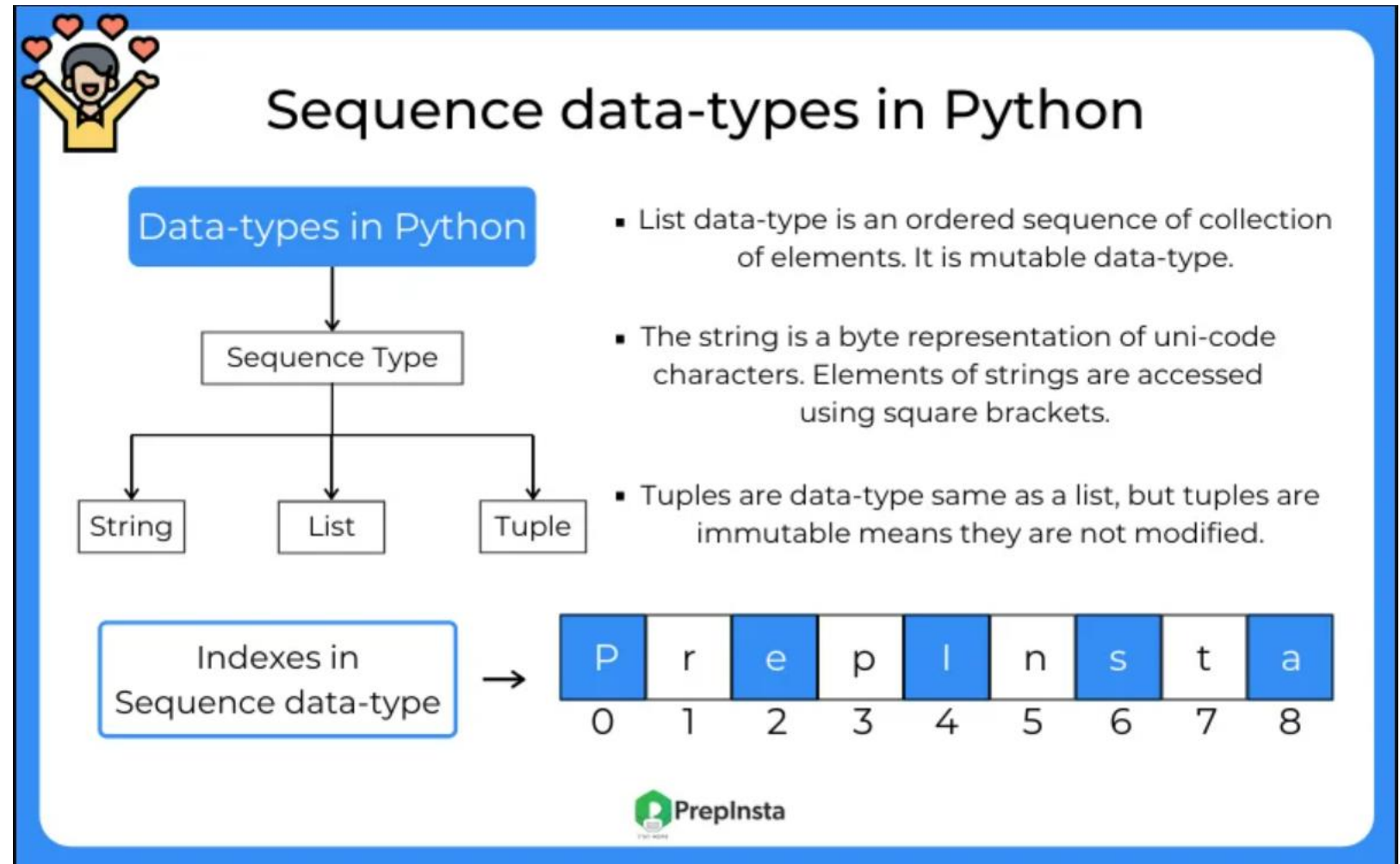
- Enter the lengths of 3 sides a , b , c of a triangle. Check whether a, b, c are 3 sides form a triangle or not?
- Display: "TRUE" -> if yes
- Display: "FALSE" -> if no

- Calculate taxi fare from the number of kilometers entered, knowing:
- First 1 km costs 15,000 VND,
- From the 2nd to the 5th km, the price is 13,500 VND,
- From the 6th kilometer onwards, the price is 11,000 VND,
- If you go more than 120km, you will receive a 10% discount on the total amount

Part 2: Repetition

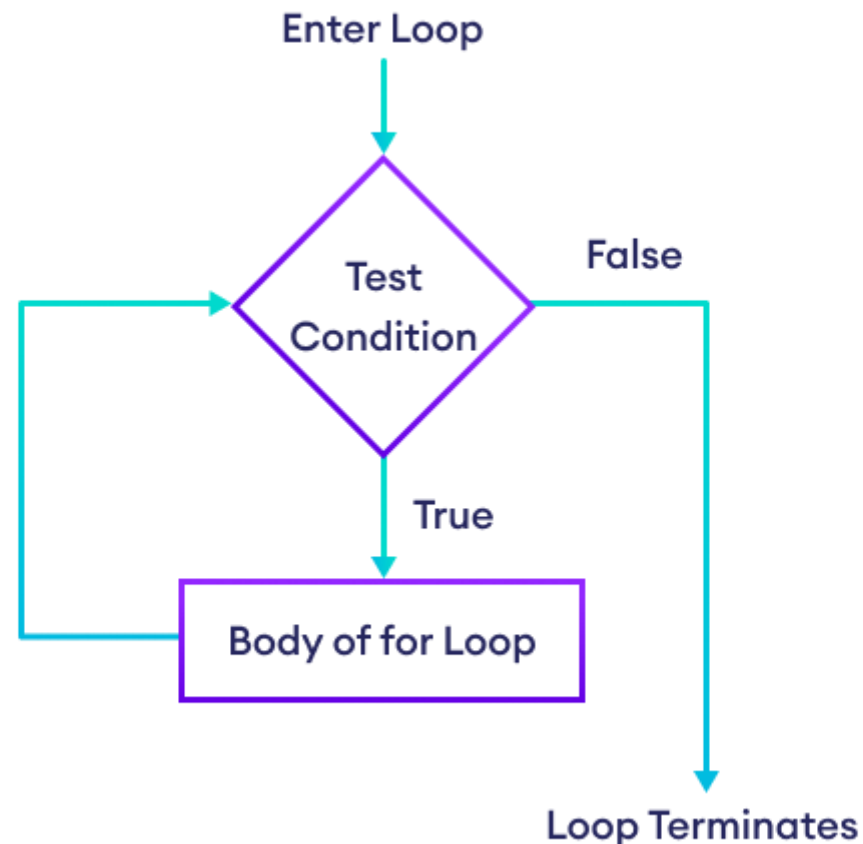
- Sequence data
- List
- String
- Tuple

is a collection of data
Indexed from **0**



- Loop the statements with each val in sequence

```
for val in sequence:  
    # statement(s)
```



- Example:

```
languages = ['Swift', 'Python', 'Go', 'JavaScript']  
  
# access items of a list using for loop  
for language in languages:  
    print(language)
```

```
Swift  
Python  
Go  
JavaScript
```

- No loop: `4 = len(languages)`
- Statements: `print(language)`
- Condition: ?

```
languages = ['Swift', 'Python', 'Go', 'JavaScript']  
  
# access items of a list using for loop  
for language in languages:  
    print(language)
```

```
range(start, stop, step)
```

Parameter Values

Parameter	Description
<i>start</i>	Optional. An integer number specifying at which position to start. Default is 0
<i>stop</i>	Required. An integer number specifying at which position to stop (not included).
<i>step</i>	Optional. An integer number specifying the incrementation. Default is 1

- We can use *range()* to generate the sequence for loop

```
# numbers from 0 to 3 (4 is not included)
numbers = range(4)
print(list(numbers))    # [0, 1, 2, 3]

# if 0 or negative number is passed, we get an empty sequence
numbers = range(-4)
print(list(numbers))    # []
```

```
# numbers from 2 to 4 (5 is not included)
numbers = range(2, 5)
print(list(numbers))    # [2, 3, 4]

# numbers from -2 to 3 (4 is not included)
numbers = range(-2, 4)
print(list(numbers))    # [-2, -1, 0, 1, 2, 3]

# returns an empty sequence of numbers
numbers = range(4, 2)
print(list(numbers))    # []
```

```
# numbers from 2 to 10 with increment 3 between numbers
numbers = range(2, 10, 3)
print(list(numbers))    # [2, 5, 8]

# numbers from 4 to -1 with increment of -1
numbers = range(4, -1, -1)
print(list(numbers))    # [4, 3, 2, 1, 0]

# numbers from 1 to 4 with increment of 1
# range(0, 5, 1) is equivalent to range(5)
numbers = range(0, 5, 1)
print(list(numbers))    # [0, 1, 2, 3, 4]
```

- The ***range()*** function is commonly used in a for loop to iterate the loop a **certain number of times**

```
# iterate the loop 5 times
for i in range(5):
    print(i, 'Hello')
```

- Write a program to calculate $S = 1 + 2 + 3 + \dots + n$ with for loop

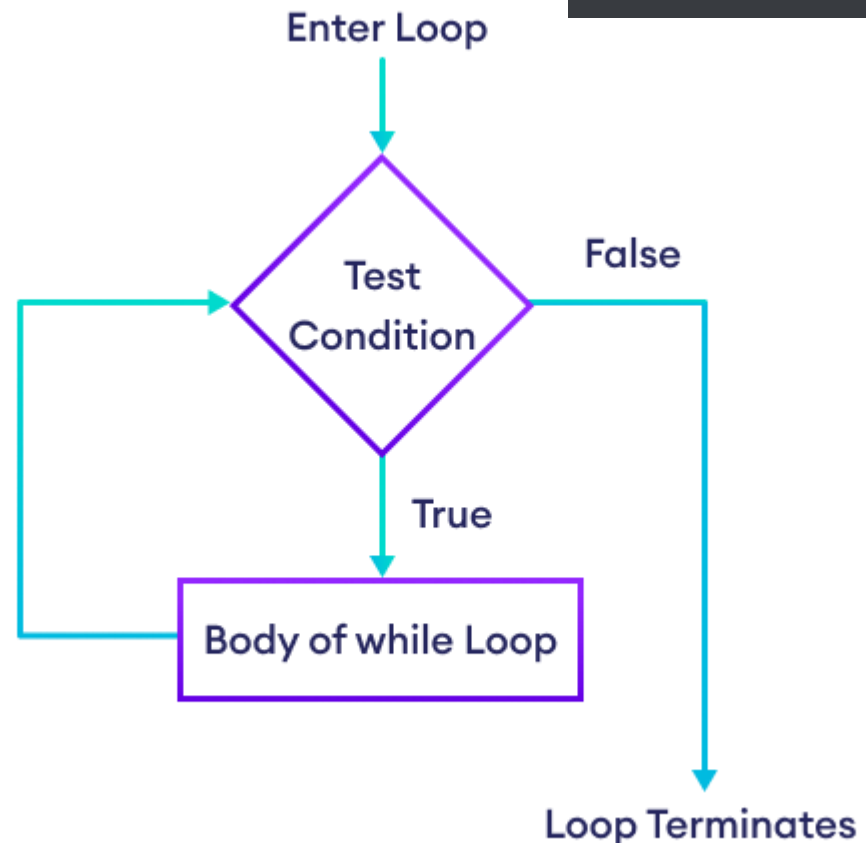
- Write a program to count the number of even integer in $[A, B]$ where A, B are obtained by keyboards

- Write a program to check whether a positive number is prime or not?

- Write a program to print all the divisors of positive integer number n

- **Repeat** <statements>
until the condition is also correct

```
while condition:  
    # body of while loop
```



- Example

```
# program to display numbers from 1 to 5

# initialize the variable
i = 1
n = 5


# while loop from i = 1 to 5
while i <= n:
    print(i)
    i = i + 1
```

- If the **condition** of a loop is **always True**, the loop runs for infinite times (until the memory is full)

```
# infinite while loop
while True:
    # body of the loop
```

For vs While Loop

Comparison Chart

For Loop	While Loop
The for loop is used for definite loops when the number of iterations is known.	The while loop is used when the number of iterations is not known.
For loops can have their counter variables declared in the declaration itself.	There is no built-in loop control variable with a while loop.
This is preferable when we know exactly how many times the loop will be repeated.	The while loop will continue to run infinite number of times until the condition is met.
The loop iterates infinite number of times if the condition is not specified.	If the condition is not specified, it shows a compilation error. 

- Write a program to find the greatest common divisors of two positive number
- by for loop
- by while loop

- Write a program to calculate the sum of all odd numbers in $[A, B]$ with A, B from keyboards

- Write a program to count the total number of digits in a number using a while loop

- What is the output of the following codes?

```
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
```

- Write a program to count the number of even digits in a positive integer number

- Write a program to reverse a given positive integer number. It means that the reversed value is number, not only displaying the result.

- Write a program to use for loop to print the following reverse number pattern

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

- Write a program to display the sequence of k numbers in Fibonacci sequence
- Hint: The Fibonacci Sequence is a series of numbers. The next number is found by adding up the two numbers before it. The **first two numbers are 0 and 1**.
- With $k = 10$, the output is

Fibonacci sequence:

0 1 1 2 3 5 8 13 21 34

Part 3: Try/Except

- Error in Python can be of two types:
- Syntax errors
- Exceptions: are raised when some internal events occur which changes the normal flow of the program

Some of common Exception Errors:

- **IOError**: if the file can't be opened
- **KeyboardInterrupt**: when an unrequired key is pressed by the user
- **ValueError**: when built-in function receives a wrong argument
- **EOFError**: if End-Of-File is hit without reading any data
- **ImportError**: if it is unable to find the module

- Debug in Pycharm: <https://www.youtube.com/watch?v=69m0ZToyR5o>

- It is used to handle these errors within our code in Python
- Try: check some code for errors
- Except: execute whenever the program encounters some error

```
try:  
    # Some Code  
except:  
    # Executed if error in the  
    # try block
```

```
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 2)
```

```
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 2)
```

- $y = 0$
- $x // y \rightarrow$ error (Run-time)

```
try:
    # Some Code
except:
    # Executed if error in the
    # try block
else:
    # execute if no exception
```

```
# Program to depict else clause with try-except

# Function which returns a/b
def AbyB(a , b):
    try:
        c = ((a+b) // (a-b))
    except ZeroDivisionError:
        print ("a/b result in 0")
    else:
        print (c)

# Driver program to test above function
AbyB(2.0, 3.0)
AbyB(3.0, 3.0)
```

```
try:
    # Some Code
except:
    # Executed if error in the
    # try block
else:
    # execute if no exception
finally:
    # Some code .....(always executed)
```



```
# Python program to demonstrate finally

# No exception Exception raised in try block
try:
    k = 5//0 # raises divide by zero exception.
    print(k)

# handles zerodivision exception
except ZeroDivisionError:
    print("Can't divide by zero")

finally:
    # this block is always executed
    # regardless of exception generation.
    print('This is always executed')
```

- Write a program to calculate the result of the following equation:

$$S = \sqrt{\frac{1}{2} + \sqrt{\frac{3}{4} + \sqrt{\frac{5}{6} + \dots + \sqrt{\frac{n-1}{n}}}}}$$

- Write a program to check a perfect number. We know that In number theory, a perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself

Input	Output
28	TRUE
30	FALSE

THANK YOU
for YOUR ATTENTION