VNUHCM - University of Science

fit@hcmus

Fundamentals of Programming for Artificial Intelligence

Session 02
Repetition & Function

Instructors:
Dr. Lê Thanh Tùng
Dr. Nguyễn Tiến Huy

Content

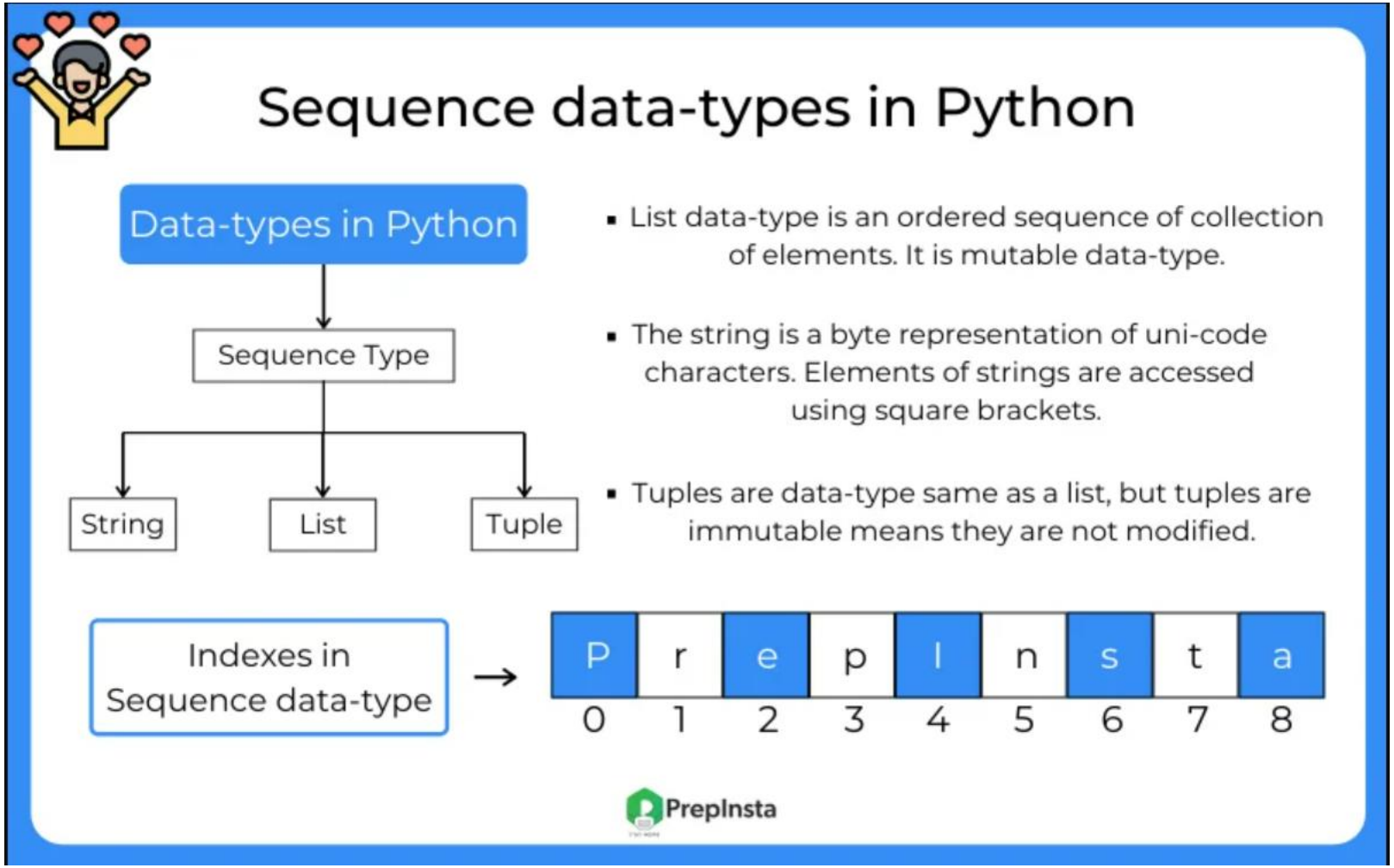
1. Function
2. Try/Except
3. Function

1. Repetition

- Sequence data
  - List
  - String
  - Tuple

is a collection of data

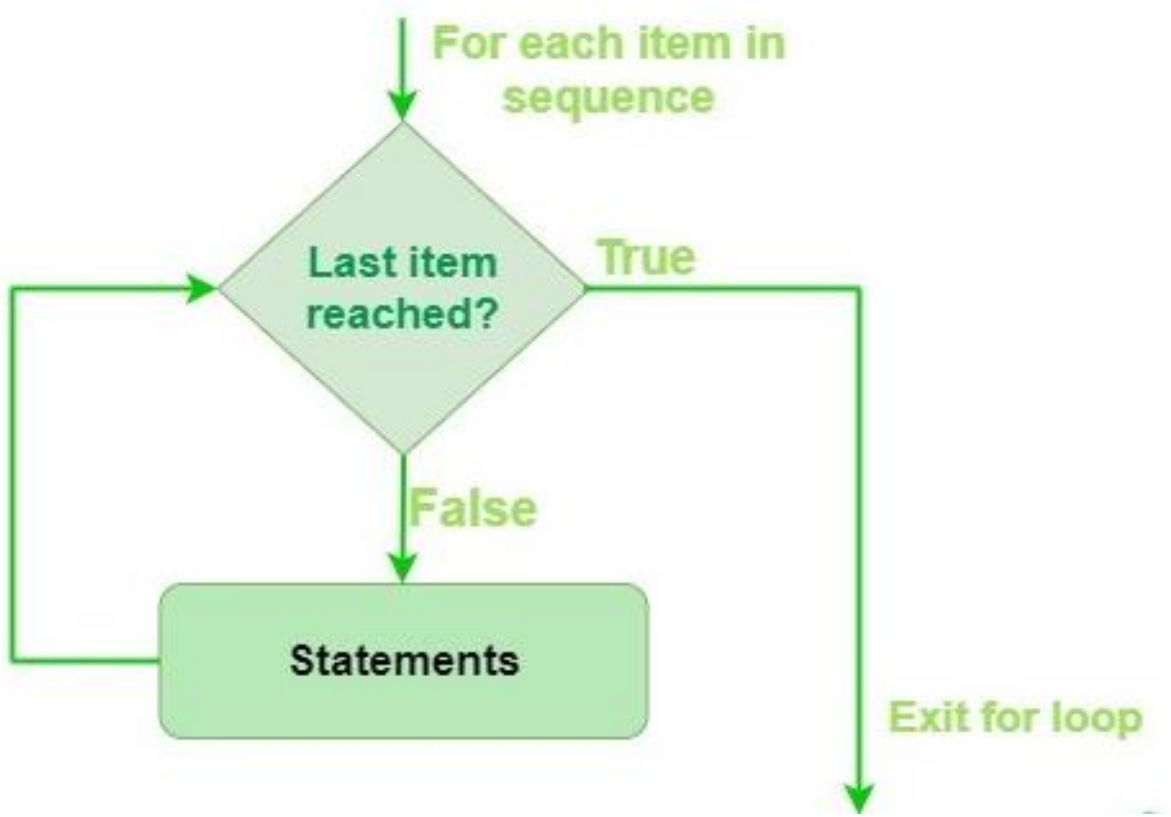
Indexed from **0**



For Loop

- Loop the statements with each val in sequence

```
for item in sequences:
    # statements
```



For Loop

- Example:

```
languages = ["Swift", "Python", "Go", "Javascript"]

#access items of a list using for loop
for language in languages:
    print(language)
```

Output:

```
Swift
Python
Go
Javascript
```

For Loop

```
languages = ["Swift", "Python", "Go", "Javascript"]

#access items of a list using for loop
for language in languages:
    print(language)
```

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

For Loop with range()

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

Parameter Values

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

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

For Loop with range()

```
# create a sequence from 0 to 3 (4 is not included)
numbers = range(4)

# convert to list and print it
print(list(numbers))
# Output: [0, 1, 2, 3]
```

For Loop with range()

```
# create a sequence from 0 to 3 (4 is not included)
numbers = range(4)

# convert to list and print it
print(list(numbers))
# Output: [0, 1, 2, 3]
```

For Loop with range()

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

# create a sequence from -2 to 3
numbers = range(-2, 4)
print(list(numbers))    # [-2, -1, 0, 1, 2, 3]

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

For Loop with range()

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

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

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

For Loop with range()

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

```
# iterate the loop five times
for i in range(5):
    print(f'{i} Hello')
```

- Write a program to calculate this equation with for loop

$$S(n) = 1 + \frac{1+2}{2!} + \frac{1+2+3}{3!} + \ldots + \frac{1+2+\ldots+n}{n!}$$

$$S_2(n) = \frac{1}{n!} + \frac{1+2}{(n-1)!} + \frac{1+2+3}{(n-2)!} + \ldots + \frac{1+2+\ldots+n}{1}$$

Exercise

- Write a program to calculate the average of prime numbers in [A, B] where A, B are obtained by keyboards

Exercise

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

Exercise

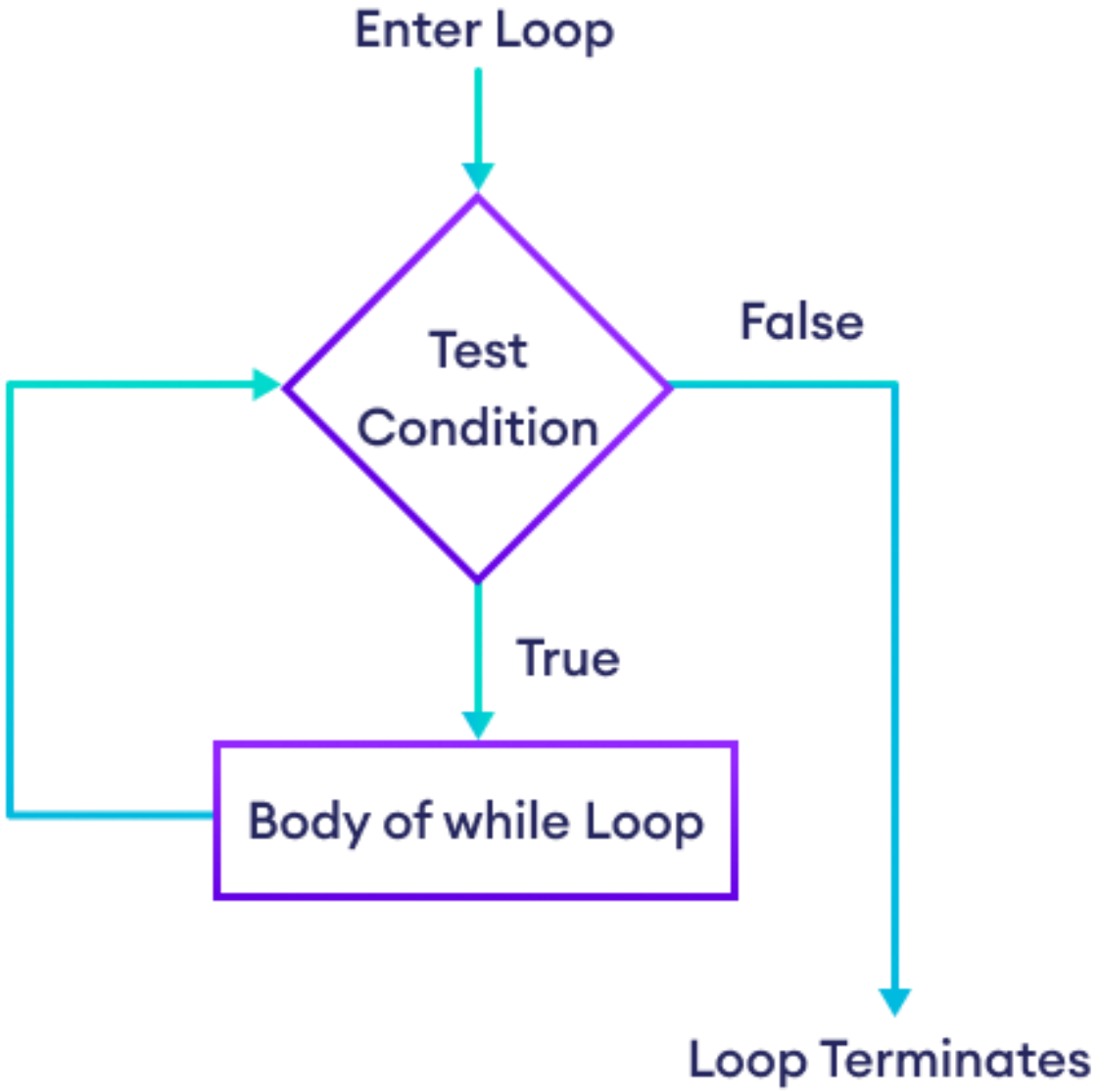
- Write a program to print all perfectly squared number in the [1, n] where n is from the keyboard

While loop

- **Repeat** <statements>

  as the condition is also correct

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



While loop

```
#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
```

While loop: Infinite loop

- 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

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.

Exercises

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

Exercises

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

Exercises

- Write a program to check whether the digits of the number n is increasingly from left to right

Interesting !!!

- 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 find the biggest digit in the integer n

- Write a program to reverse a given positive integer number. It mean 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
```

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

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

- Given an integer n, when we delete a digit d (all positions with digit d in n), we can obtain the integer m. Write a program to find m such that m is the biggest number in all cases.

Input	Output
288	88
3434	44

2. Try/Except

Errors

- 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

Try/Except:

- 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
```

Example

```
# 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)
```

Example

```
# 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 → error (Run-time)

Try/Except ... else

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

Try/Except ... else

```
# 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/Except ... else ... finally

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

Try/Except ... else ... finally

```
# 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')
```

3. Function

Function

- A function is a block of code that performs a specific task
- Dividing a complex problem into smaller chunks makes our program easy to understand and reuse

Python Function Declaration

The syntax to declare a function is:

```
def function_name(arguments):
    # function body

    return
```

Here,

- `def` - keyword used to declare a function
- `function_name` - any name given to the function
- `arguments` - any value passed to function
- `return` (optional) - returns value from a function

Calling a function

- First, declare and define a function

```
def greet():
    print('Hello World!')
```

- However, it does not run until we call it

```
# call the function
greet()
```

Detailed declaration

- Think about data types of parameters and return value

```
def function_name(parameter: data_type) -> return_type:
    """Doctring"""
    # body of the function
    return expression
```

```
def add(num1: int, num2: int) -> int:
    """Add two numbers"""
    num3 = num1 + num2

    return num3
```

Arguments of a Python Function

- Information can be passed into functions as arguments
- All parameters (arguments) in the Python language are passed by reference
- Types of arguments:
  - Default argument
  - Variable-length arguments

Default argument

- A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument
- Once we have a default argument, all the arguments to its right must also have default values

```
def myFun(x, y=50):
    print("x: ", x)
    print("y: ", y)


# Driver code (We call myFun() with only
# argument)
myFun(10)
```

Exercise

- Write a function to check whether an integer is prime or not?

Exercise

- Write a function to find the smallest prime number which is bigger than the positive integer k

Variable-length

- In Python, we can pass a variable number of arguments to a function using special symbols. There are two special symbols:
  - *args (Non-Keyword Arguments) – tuple of arguments
  - **kwargs (Keyword Arguments) – a dictionary of arguments

```
def myFun(*argv):
    for arg in argv:
        print(arg)


myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

```
def myFun(**kwargs):
    for key, value in kwargs.items():
        print("%s == %s" % (key, value))


# Driver code
myFun(first='Geeks', mid='for', last='Geeks')
```

Docstring

- The first string after the function is called the Document string or Docstring in short
- This is used to describe the functionality of the function

```
def evenOdd(x):
    """Function to check if the number is even or odd"""

    if (x % 2 == 0):
        print("even")
    else:
        print("odd")


# Driver code to call the function
print(evenOdd.__doc__)
```

Pass by Reference or pass by value fit@hcmus

- In Python every variable name is a reference
- When we pass a variable to a function, a new reference to the object is created

```
# Here x is a new reference to same list lst
def myFun(x):
    x[0] = 20


# Driver Code (Note that lst is modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun(lst)
print(lst)
```

Output

```
[20, 11, 12, 13, 14, 15]
```

Pass by Reference or pass by value fit@hcmus

- When we pass a reference and change the received reference to something else, the connection between the passed and received parameter is broken

```
def myFun(x):

    # After below line link of x with previous
    # object gets broken. A new object is assigned
    # to x.
    x = [20, 30, 40]


# Driver Code (Note that lst is not modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun(lst)
print(lst)
```

Output

```
[10, 11, 12, 13, 14, 15]
```

Pass by Reference or pass by value fit@hcmus

- When we pass a reference and change the received reference to something else, the connection between the passed and received parameter is broken

```
def myFun(x):

    # After below line link of x with previous
    # object gets broken. A new object is assigned
    # to x.
    x = 20


# Driver Code (Note that lst is not modified
# after function call.
x = 10
myFun(x)
print(x)
```

Output

```
10
```

Exercise

- Guest the result of the following code:

```
def swap(x, y):
    temp = x
    x = y
    y = temp


# Driver code
x = 2
y = 3
swap(x, y)
print(x)
print(y)
```

Exercise

- Guest the result of the following code:

```
def swap(x, y):
    temp = x
    x = y
    y = temp


# Driver code
x = 2
y = 3
swap(x, y)
print(x)
print(y)
```

Output

```
2
3
```

Exercise

- Write a function to find the maximum value in a list of integer

Exercise

- Write a function to count the prime number in a list of integer

Exercise

- Write a function to sort a list of integer via selection sort

Exercise

- Write a function to determine whether a list of integer is symmetrical or not ?

Exercise

- Write a function to simplify a fraction
  - Find GCD

THANK YOU
for YOUR ATTENTION