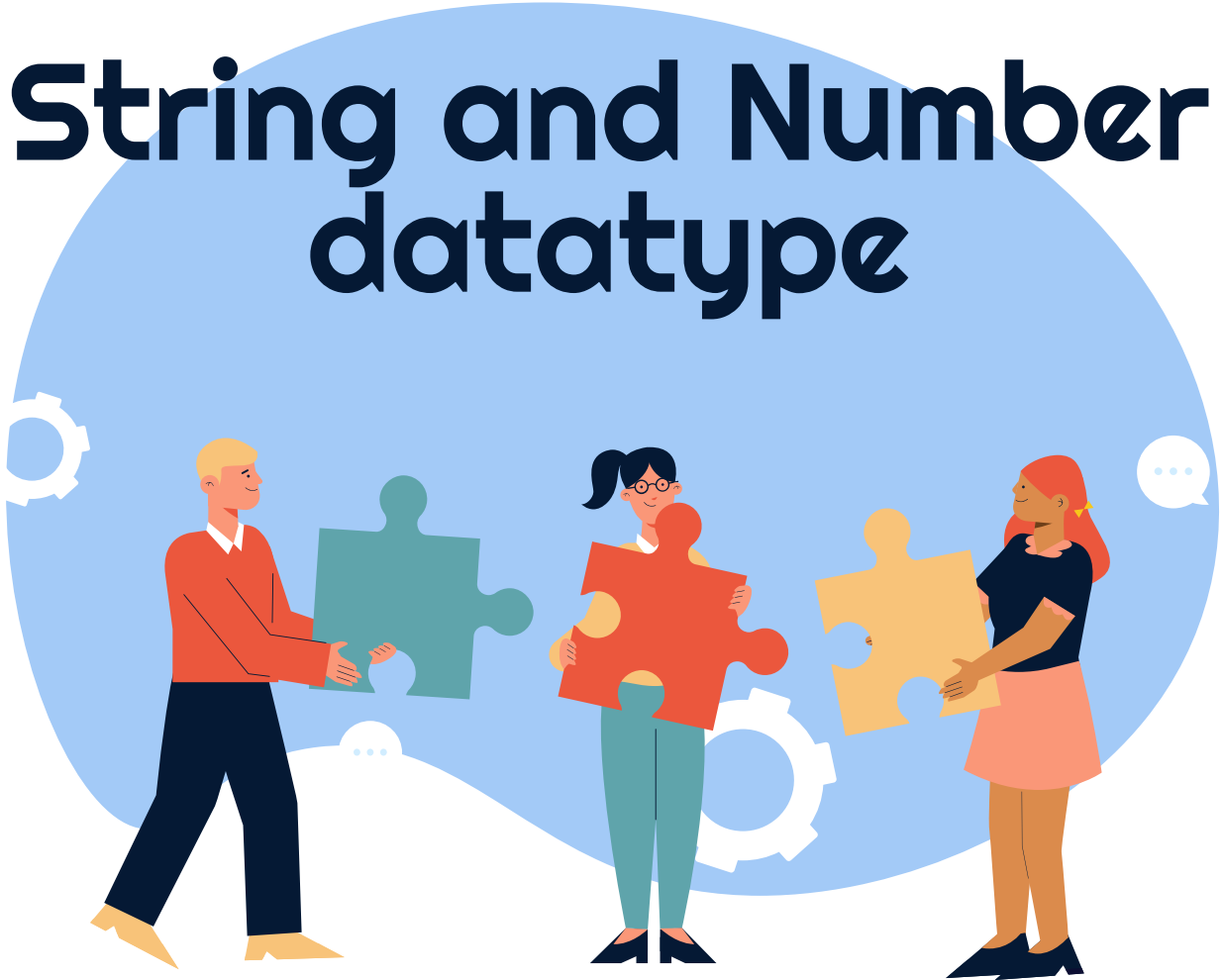
String and Number datatype



mindX
Technology & Startup School

01

STRING

Represent text

String represent any type of text, symbols, numbers

String is defined by using a pair of single quotes, or double quotes

But.... why???

```
"This is a string"
```

```
'This is also a string'
```

String can span multiple lines by using three double quote, or three single quotes

```
>>> a = """This
is
a
multi line
string"""
>>> print(a)
This
is
a
multi line
string
```

```
>>> b = '''This
is
another
multi line
string'''
>>> print(b)
This
is
another
multi line
string
```

String concatenation

Python string allows the use of the “+” operator to concat 2 strings together

```
>>> a = "This " + "is"
>>> b = " a string"
>>> print(a + b)
This is a string
```

String interpolation

String can contain variable placeholder, or an expression

To use it, just prefix the string with ‘f’ character, and put the variable/expression in a pair of curly brackets ‘{ }’

```
>>> height = 170
>>> print(f"You are {height} cm tall")
You are 170 cm tall
```

```
>>> radius = 5
>>> print(f'The area of the circle is {radius * radius * 3.14}')
The area of the circle is 78.5
```

String length

To get the length of a string, pass the string to the len() function

```
>>> a = ""
>>> b = "this string has 29 characters"
>>> len(a)
0
>>> len(b)
29
```

Substring, get character at position

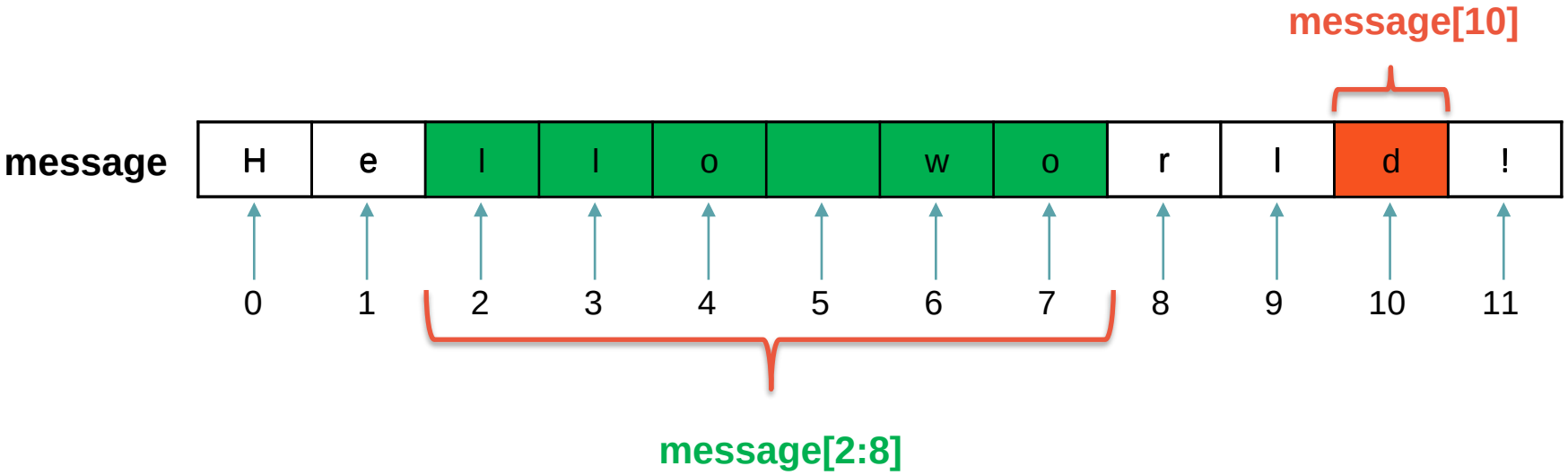
Characters in string start at 0, end at len() - 1

Substring: get part of the string, using the syntax **str[start_position:end_position]**

Where end_position is exclusive

Get a character at a position: **str[position]**

Substring, get character at position




```
>>> message = "Hello world!"
>>> message[6]
'w'
```

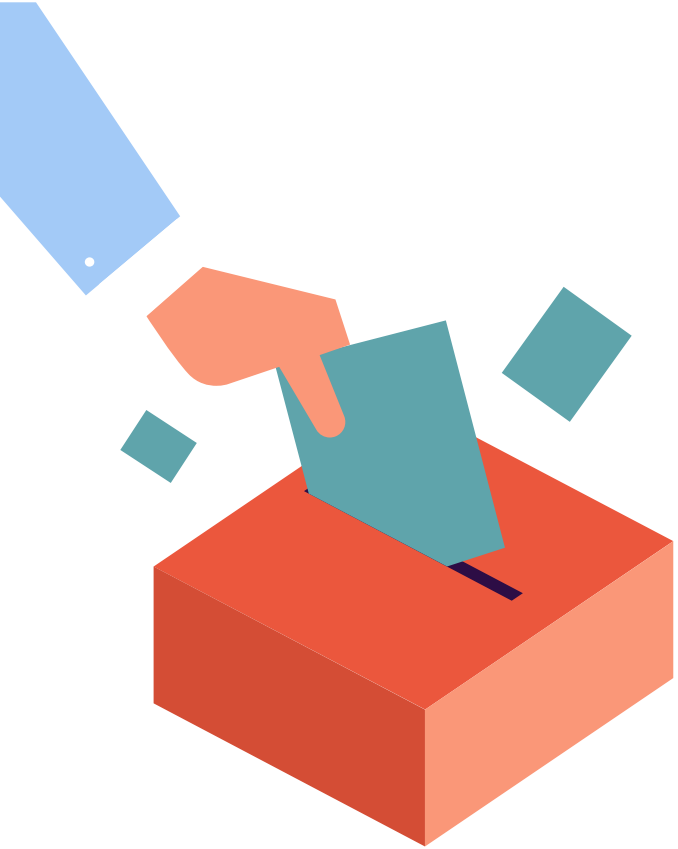
```
>>> message[2:7]
'llo w'
```

Strings are immutable!!!

You’re not allowed to change a character inside a string just by assigning it to a new character!!

mindX
Technology & Startup School

02



Number types

Simply remember: there are only Int and Float

type()

int

Integer denotes whole number

number_of_students = 20

late_students = 0

score = -2

float

Floating point represents decimal number. It's defined by the "symbol ."

speed = 2033
height = -6..0
weight = 12.
Score = .5

mindX
Technology & Startup School

03

Mathematical operators

\+ - * /

% **

Any variable of type int or float can perform mathematical operation with each other

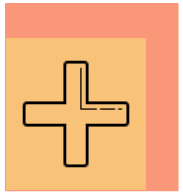
The result produced from the operation depends on the larger type

Int with int => int

Float with float => float

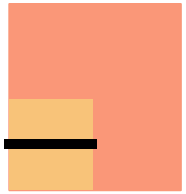
Int with float => float

Common operations



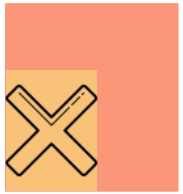
Addition

```
>>> a = 5
>>> b = 6.2
>>> a + b
11.2
```



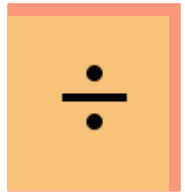
Subtraction

```
>>> a = 10
>>> b = 20.
>>> a - b
-10.0
```



Multiplication

```
>>> a = 10.5
>>> b = 0
>>> a * b
0.0
```

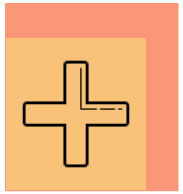


Division

```
>>> a = 9
>>> b = 2
>>> a / b
4
```

```
>>> a = 9.0
>>> b = 2
>>> a / b
4.5
```

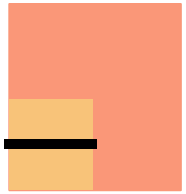
Other operations



Modulus

```
>>> a = 5
>>> b = 2
>>> a % b
1
```

The remainder is 1



Exponential

```
>>> a = 10
>>> b = 3
>>> a ** b
1000
```

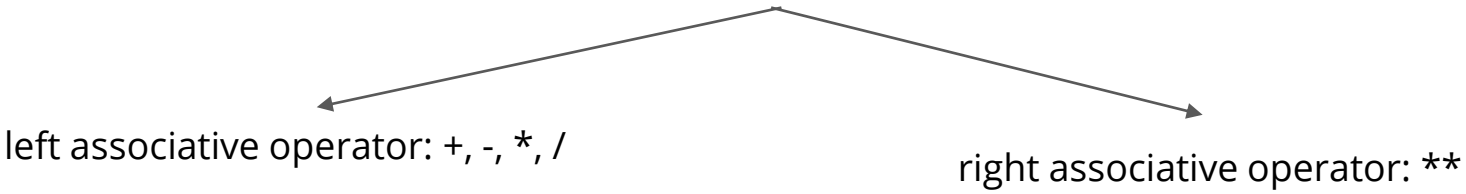
10 ^ 3 = 10 * 10 * 10 = 1000

Associativity

$$10 - 6 - 1 = \text{ 5 or 3}$$

Associativity

The associativity of an operator tells you what to do when you have a run of multiple instance of the same operator



Precedence

20 : 5 * 2 = 2 or 8

Precedence

The precedence of an operator tells you what to do when you have a mix of different operator

Notes: Sometimes you need to import Math library to use more functions

Operators	Meaning
`()`	Parentheses
`**`	Exponent
`+x`, `-x`, `~x`	Unary plus, Unary minus, Bitwise NOT
`*`, `/`, `//`, `%`	Multiplication, Division, Floor division, Modulus
`+`, `-`	Addition, Subtraction
`<<`, `>>`	Bitwise shift operators
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==`, `!=`, `>`, `>=`, `<`, `<=`, `is`, `is not`, `in`, `not in`	Comparisons, Identity, Membership operators
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

The operator precedence in Python (in descending order)


BONUS

Casting between data types

mindX Technology & Startup School

You can cast to specify a type on to a variable

int() -> Construct an integer number

float() -> Constructs a float number

str() -> Constructs a string

```
1   num = 10
2
3   string = str(num)
4
5   print(f"Type of num = {num} is", type(num))
6   print(f"Type of string = {string} is", type(string))
```

```
Type of num = 10 is <class 'int'>
Type of string = 10 is <class 'str'>
```

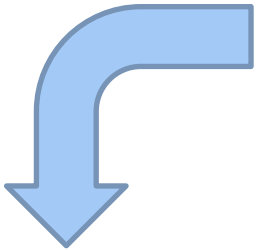
Commenting

Comment in Python

```
#Đây là một comment
##Đây cũng là 1 comment

### Vẫn thế...
################################ Comment....

print("SoMeThInG sTuPid") #Comment ở đây cũng được luôn

#nhưng mà nên comment bên trên câu lệnh như này nhé
nhap_mot_cai_gi_do_vao_day = input()

print("#Trong này thì say No với comment nhé")

Bơ vơ ngoài này thì cũng không phải comment đâu :)

```

Commenting

Docstring in Python



```python
#Trong khi cái này chỉ comment đc 1 dòng thôi
""" Thì dùng " " " Ta có thể comment
rất
là
nhiều
dòng
bằng cách này
    nhé!!! """

#Thậm chí còn có thể print ra được luôn bằng cách dùng __doc__ nhé (có 2 dấu _ ở 2 bên)
print(__doc__)

```

```
PS C:\Users\Viet> & C:/Users/Viet/AppData/Local/Programs/Python/Python39/python.exe d:/MindX/Class/Python/Demo/Untitled-1.py
 Thì dùng " " " Ta có thể comment
rất
là
nhiều
dòng
bằng cách này
    nhé!!!
```

STRING

Multiply a string

A string can be multiplied by a number to get a new string

```
>>> s = "a"
>>> print(s)
a
>>> s = "a" * 3
>>> print(s)
aaa
>>>
```

STRING

Escape Sequence

To include special characters in strings
=> by adding a backslash before the character.

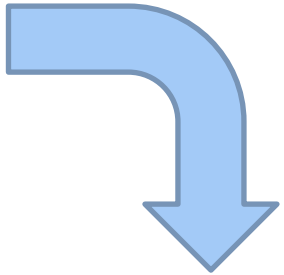
Escape-sequence	Purpose
\n	New line
\\	Backslash character
\'	Apostrophe '
\"	Quotation mark "
\a	Sound signal
\b	Slaughter (backspace key symbol)
\f	The conversion of format
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\xhh	Character with hex code hh
\ooo	Character with octal value ooo
\0	Character Null (not a string terminator)
\N{*id*}	Identifier ID of Unicode database
\uhhhh	16-bit Unicode character in hexadecimal format
\Uhhhhhhhh	32-bit Unicode character in hexadecimal format
\другое	Not an escape sequence (\ character is stored)

STRING

Raw String

To ignore the escape sequence
=> By prefixing the string with ‘r’ or ‘R’

```
#Without Raw String
p = "Đây là đường dẫn tới thư mục bí mật: C:\data\invisible\nothing"
print("Without raw string:")
print(f"=> path: {p}")

print("")

#With Raw String
p = r"Đây là đường dẫn tới thư mục bí mật: C:\data\invisible\nothing"
print("With raw string:")
print(f"=> path: {p}")
```

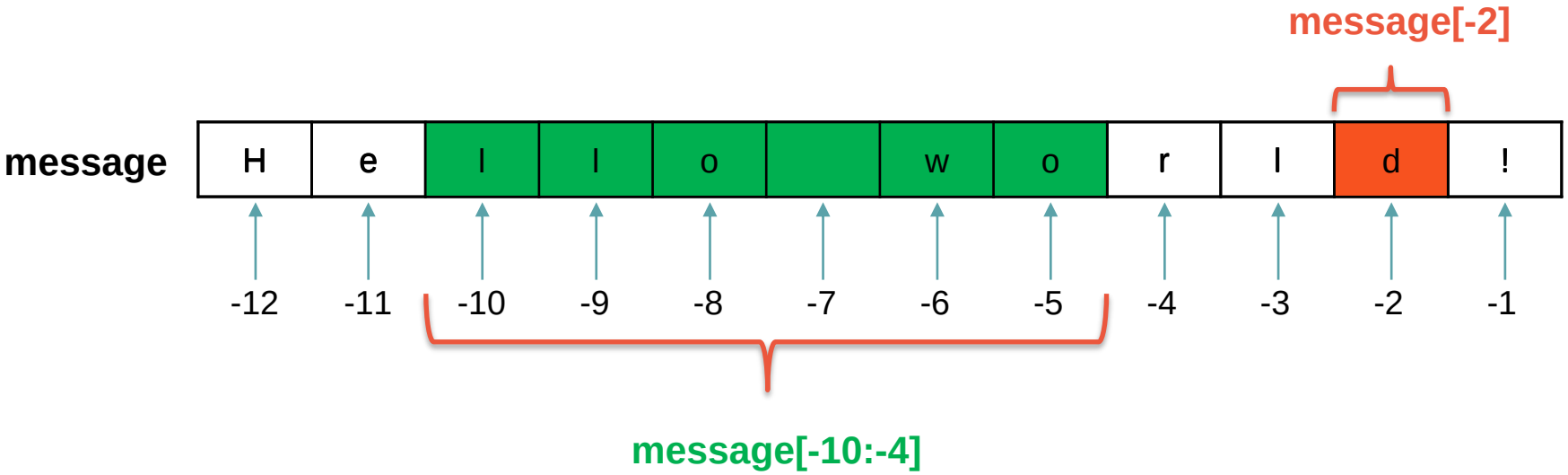


```
Without raw string:
=> path: Đây là đường dẫn tới thư mục bí mật: C:\data\invisible
othing

With raw string:
=> path: Đây là đường dẫn tới thư mục bí mật: C:\data\invisible\nothing
```

STRING

Substring, get character at position

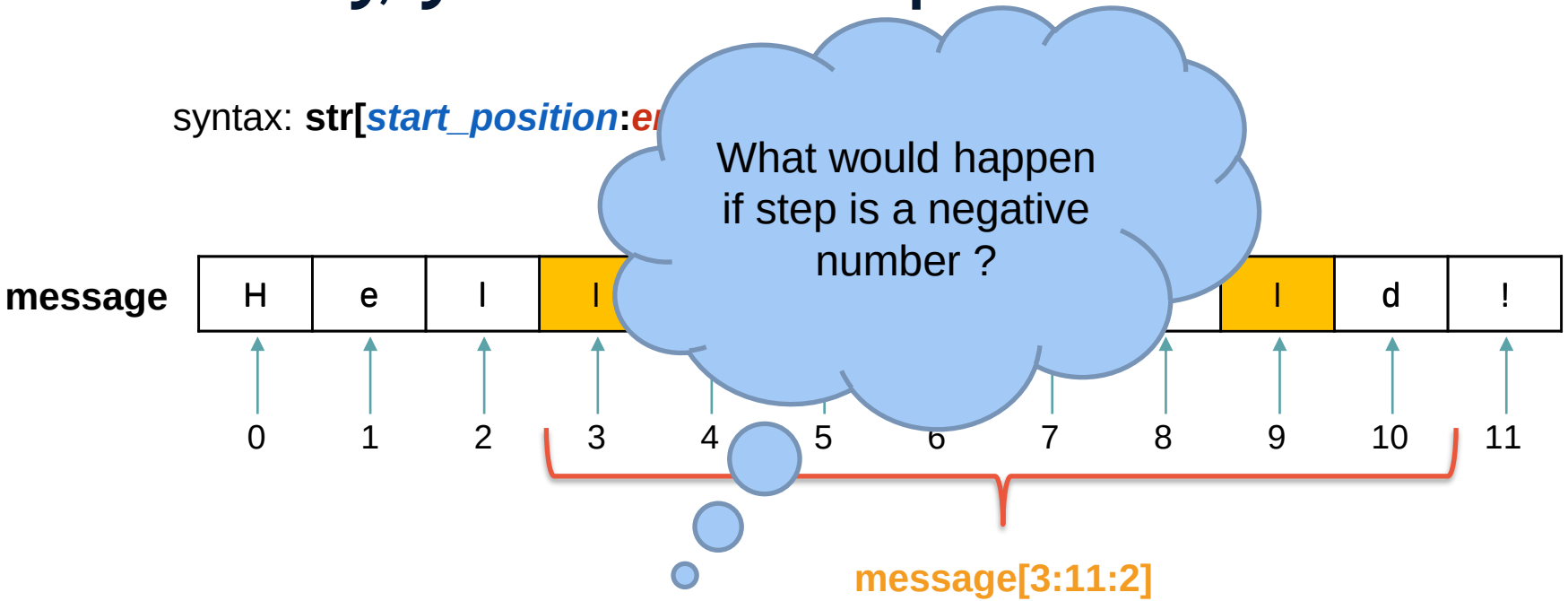
Slice with negative indices



STRING

Substring, get character at position

syntax: **str[*start_position*:*en*



STRING

String format() method

A string can be multiplied by a number to get a new string

syntax: *"**string {}** "*.**format(***value1*, *value2*, *value3*…)**

"string" includes the placeholder:
- Defined using curly brackets { }
- Identified using named indexes {name} by user ; numbered indexes {0}, {1}… counts from 0 ; or empty {}

Values (of any data type) to be inserted in the string

STRING

String format() method

mindX Technology & Startup School

The Placeholders:

```
fact = "I am very hungry. I need to {} {} and {} {}.".format()
```

1. Named indexes: (by user

```
1   fact = "I am very hungry. I need to {verb1} {noun1} and {verb2} {noun2}.".format
        (verb1 = "eat", noun1 = "a banana", verb2 = "drink", noun2 = "some beer")
```

2. Numbered indexes: (values inside format() are counted from 0, in this example, we have 4 values are in order: 0,1,2,3)

```
2   fact = "I am very hungry. I need to {0} {1} and {2} {3}.".format("eat", "a banana","drink", "some beer")
```

3. Empty placeholders: (placeholders are assigned values from *left* to *right*)

```
3   fact = "I am very hungry. I need to {} {} and {} {}.".format("eat", "a banana","drink", "some beer")
```

=> They have the same OUTPUT:

```
I am very hungry. I need to eat a banana and drink some beer.
```

String format() method

You can also add a formatting type to format the result

=> Try it now: Python String format() Method (w3schools.com)

STRING

String methods

There are many built-in methods that you can use on strings.
(e.g: Convert all characters to uppercase, remove extra whitespaces...)

=> Try it now: Python String Methods (w3schools.com)