FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Topic 4:
Strings

CSGE601020 - Dasar-Dasar Pemrograman 1

Acknowledgement

Several materials are reused from ‘Strings’ slides used in Dasar-Dasar Pemrograman 1 dengan Python (CSGE601020/4 SKS) Course (https://ocw.ui.ac.id/course/view.php?id=142)
by **Fariz Darari, Ph.D.**

Some of the design assets used in these slides were provided by ManyPixels under an nonexclusive, worldwide copyright license to download, copy, modify, distribute, perform, and use the assets provided from ManyPixels for free, including for commercial purposes, without permission from or attributing the creator or ManyPixels.

Some additional contents, illustrations and visual design elements are provided by **Lintang Matahari Hasani, M.Kom.** *(lintang.matahari01[at]cs.ui.ac.id)*

Outline

- String Datatype
- Indexing and Slicing
- String Operations
- F-String

String: Sequence of characters

- We've talked about strings being a sequence of characters.
- A string is indicated between ' ' or " "
- The exact sequence of characters is maintained
- It is also a collection
- Create the object with string literal or `str` constructor

Triple-Quote String

- Triple quotes preserve both the vertical and horizontal formatting of the string
- Allows you to type tables, paragraphs, and other kinds while preserving the formatting
- Example:

```
x = """this is
a
    string in
multiline"""
print(x)
```

Python special characters

Escape Code	Meaning
\\	\ (backslash)
\n	newline
\t	tab
\'	' single quote character
\"	" double quote character

Quotes for strings

- Can use single or double quotes:

```
s = "spam"
s = 'spam'
```

- Just don't mix them

```
my_str = 'hi mom" # error!
```

- Inserting an apostrophe:

```
a = "knight's"
b = 'knight\'s'
```

String representation

- Every character is "mapped" (associated) with an integer
- UTF-8, subset of Unicode, is such a mapping
- The function ord() takes a character and returns its UTF-8 integer value
- The function chr() takes an integer and returns the UTF-8 character

```
>>> ord('a')
97
>>> ord('?')
63
>>> ord('\n')
10
>>> chr(10)
'\n'
>>> chr(63)
'?'
>>> chr(97)
'a'
```

Character-to-int mapping

!	33	EXCLAMATION MARK
"	34	QUOTATION MARK
#	35	NUMBER SIGN
$	36	DOLLAR SIGN
%	37	PERCENT SIGN
&	38	AMPERSAND
'	39	APOSTROPHE
(	40	LEFT PARENTHESIS
)	41	RIGHT PARENTHESIS
*	42	ASTERISK
+	43	PLUS SIGN
,	44	COMMA
-	45	HYPHEN-MINUS
.	46	FULL STOP
/	47	SOLIDUS

Character-to-int mapping

0	48	DIGIT ZERO
1	49	DIGIT ONE
2	50	DIGIT TWO
3	51	DIGIT THREE
4	52	DIGIT FOUR
5	53	DIGIT FIVE
6	54	DIGIT SIX
7	55	DIGIT SEVEN
8	56	DIGIT EIGHT
9	57	DIGIT NINE

Character-to-int mapping

A	65	LATIN CAPITAL LETTER A
B	66	LATIN CAPITAL LETTER B
C	67	LATIN CAPITAL LETTER C
D	68	LATIN CAPITAL LETTER D
E	69	LATIN CAPITAL LETTER E
F	70	LATIN CAPITAL LETTER F
G	71	LATIN CAPITAL LETTER G
H	72	LATIN CAPITAL LETTER H
I	73	LATIN CAPITAL LETTER I
J	74	LATIN CAPITAL LETTER J
K	75	LATIN CAPITAL LETTER K
L	76	LATIN CAPITAL LETTER L
M	77	LATIN CAPITAL LETTER M

Character-to-int mapping

a	97	LATIN SMALL LETTER A
b	98	LATIN SMALL LETTER B
c	99	LATIN SMALL LETTER C
d	100	LATIN SMALL LETTER D
e	101	LATIN SMALL LETTER E
f	102	LATIN SMALL LETTER F
g	103	LATIN SMALL LETTER G
h	104	LATIN SMALL LETTER H
i	105	LATIN SMALL LETTER I
j	106	LATIN SMALL LETTER J
k	107	LATIN SMALL LETTER K
l	108	LATIN SMALL LETTER L
m	109	LATIN SMALL LETTER M

Index for strings

characters	H	e	l	l	o		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10
									...	−2	−1

Because the elements of a string are a sequence (= rentetan),

we can associate each element with an index,

alias a location in the sequence:

positive values count up from the left, beginning with index 0

negative values count down from the right, starting with -1

Accessing an element

characters	H	e	l	l	o		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10
									...	−2	−1

A particular element of the string is accessed by the index of the element surrounded by square brackets [ ]

```
hello_str = 'Hello World'
print(hello_str[1])    # prints e
print(hello_str[-1])   # prints d
print(hello_str[11])   # Error!
```

Middle characters Exercise

Masukkan string: MAHASISWA
Karakter paling awal adalah M
Karakter paling akhir adalah A
Karakter tengah adalah S

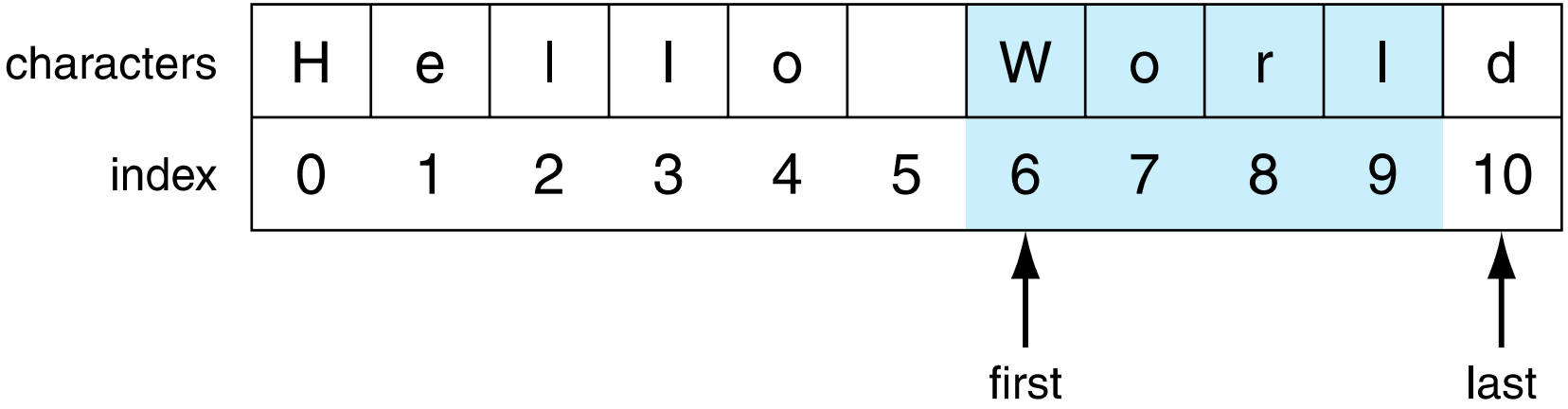
Masukkan string: WOW
Karakter paling awal adalah W
Karakter paling akhir adalah W
Karakter tengah adalah O

Slicing strings

- Slicing (= pengirisan) is the ability to select a subsequence of the overall sequence
- Uses the syntax `[start:finish]`, where:
  - `start` is the index of where we start the subsequence
  - `finish` is the index of one after where we end the subsequence
- If either start or finish are not provided, it defaults to the beginning of the sequence for start and the end of the sequence for finish

Slicing strings

- `hello_str[6:10]`



Slicing strings

- `hello_str[6:]`

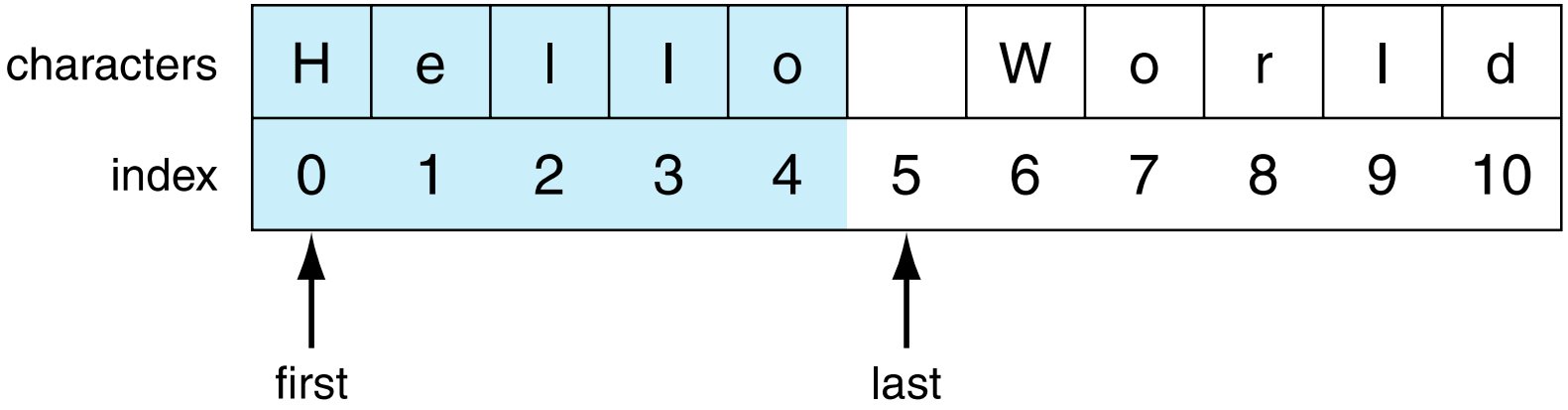
characters	H	e	l	l	o		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10

first

last

Slicing strings

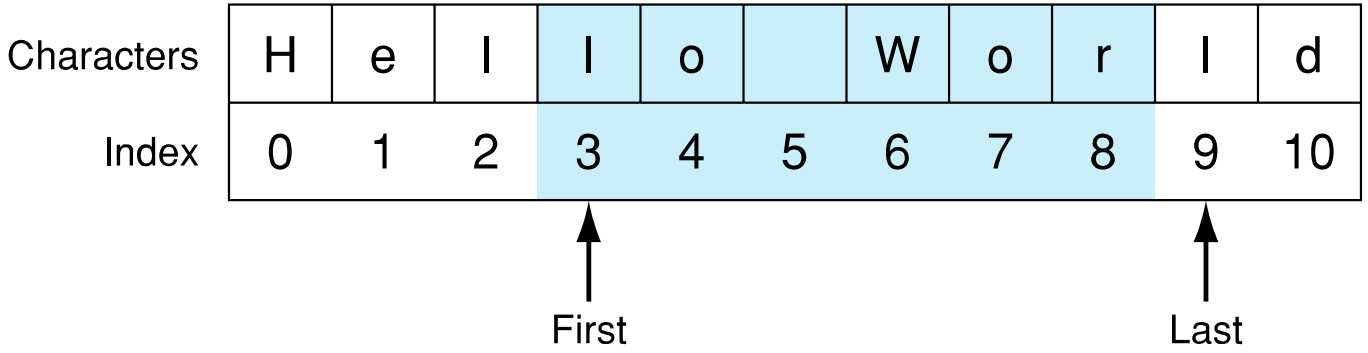
- `hello_str[:5]`



Slicing strings

- `hello_str[3:-2]`

characters	H	e	l	l	o		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10
									...	−2	−1



Extended slicing

- Extended slicing takes three arguments:

  `[start:finish:step]`
- Defaults are:

  start is beginning, finish is end, step is 1
- Example:

```
hello_str = 'Hello World'
print(hello_str[0:11:2]) # 'HloWrd'
```

Extended slicing

- `hello_str[::2]`

Characters	H	e	l	l	o		W	o	r	l	d
Index	0	1	2	3	4	5	6	7	8	9	10

Extended slicing for copying and reversing a string

```
hello_str = 'Hello World'
hello2 = hello_str[:]
print(hello2) # Hello World
hello_rev = hello_str[::-1]
print(hello_rev) # dlroW olleH
```

```
s = 'Dasar-Dasar Powerpoint'
print(s[:11]) #Dasar-Dasar
print(s[4:]) #r-Dasar Powerpoint
print(s[::4]) #DrsPrn
```

Buat program yang menerima sebuah string, kemudian cetak pola string segitiga berisi karakter-karakter awal string masukan, mulai dari string kosong sampai string masukan penuh.
Contoh:
Masukkan string: FASILKOM
F
FA
FAS
FASI
FASIL
FASILK
FASILKO
FASILKOM

Basic string operations

```
s = 'spam'
```

- length operator len()

  ```
  len(s) # 4
  ```

- Concatenate operator +

  ```
  s+s # 'spamspam'
  ```

- Repeat operator *

  ```
  s*4 # 'spamspamspamspam'
  ```

type function

You can check the type of the value associated with a variable using type

```
hello_str = 'Hello World'
print(type(hello_str)) # <class 'str'>
```

for-loop for string

The for loop iterates through each element of a string in order, that is, character by character

```
hello_str = 'Hello World'
for i in hello_str:
    print(i)
```

for-loop for string with enumerate

- The enumerate function gets two values:
  the index of an element and the element itself
- Can use it to iterate through both the index and element
  simultaneously, doing dual assignment

```
hello_str = 'Hello World'
for index, ch in enumerate(hello_str):
    print(index, ch)
```

String comparisons

- String can be compared
- Single character comparison is by simply comparing the UTF-8 representation of string

```
print('A' < 'B') # True
print('z' < 'a') # False
print('A' < 'a') # True
```

String comparisons, multiple characters

- Compare the first element of each string.
- If they are equal, move on to the next character in each.
- If they are not equal, the relationship between those to characters are the relationship between the string.
- If one ends up being shorter (but equal), the shorter is smaller.

```
print('a'   < 'b') # True
print('aab' < 'aac') # True
print('aa'  < 'aaz') # True
```

Quiz: What's the output?

```
s = 'python'
for i in range(len(s)-1):
    print(s[i] < s[i+1])
```

Membership operator `in`

- You can check if a substring exists in a string
- This is by the **`in`** operator, returning True or False depending on the substring checking

```
my_str = 'aabbccdd'
print('a' in my_str) # True
print('abb' in my_str) # True
print('x' in my_str) # False
```

Strings are immutable

- You cannot change the content of a string
- But you can create a new string by slicing the content of existing string

```
a_str = 'spam'
a_str[1] = 'c' # TypeError: 'str' object does not support item assignment
new_str = a_str[0] + 'c' + a_str[2:]
print(new_str) # scam
```

String methods

```
s = 'indONEsia'
print(s.capitalize())
print(s.lower())
print(s.swapcase())
print(s.upper())
print(s.upper().isupper())
print(s.upper().islower())
```

String methods

```
s = 'indonesia merdeka'
print(s.title())
print(s.count('d'))
print(s.find('o'))
print(s.find('nes'))
print(s.find('xyz')) # prints -1
print(s.index('o'))
print(s.index('nes'))
print(s.index('xyz')) # raise ValueError
```

String methods

```
s = 'indonesia merdeka'
print(s.replace('e','a'))
print(s.replace('merdeka','jaya'))
print(s.split()) # ['indonesia','merdeka']
print('#'.join(['indonesia','merdeka']))
```

String methods

```
print("HiBo22".isalnum())
print("Hi Bo22!!".isalnum())
print("HiBo22".isalpha())
print("HiBosss".isalpha())
print("HiBo22".isdigit())
print("8055".isdigit())
```

Outline

- → String Datatype
- → Indexing and Slicing
- → String Operations
- → **F-String**

Presenting the output of a program, the fancy way

- You wish to print data in a fancy, human-readable form
- You may want more control over the formatting of your output
- f-string is the answer : -)

f-string

- f-string is a more concise, more readable way to format strings in Python
- In other words, f-string provides an option to embed expressions inside string literals, using a minimal syntax
- In source code, f-strings are strings prefixed by the letter 'f' or 'F'
- f-string is short for **formatted-string** (or if you wish, *fancy-string* ; -)
- f-string is supported from Python 3.6 (in 2015) onwards (PEP 498)

Hello! (before f-string)

```
name = "Guido"
print("Hello, " + name + "!")
```

Hello! (before f-string)

```
name = "Guido"
print("Hello, " + name + "!")
```

```
Hello, Guido!
```

Hello! (after f-string)

```
name = "Guido"
print(f"Hello, {name}!")
```

Hello! (after f-string)

```
name = "Guido"
print(f"Hello, {name}!")
```

```
Hello, Guido!
```

Hello. it's me!

```
name = "Guido"
me = "Fariz"
print(f"Hello, {name}! It's me, {me}!")
```

Hello. it's me!

```
name = "Guido"
me = "Fariz"
print(f"Hello, {name}! It's me, {me}!")
```

```
Hello, Guido! It's me, Fariz!
```

Number and string

```
year = 2022
name = "World"
print(f"Hello, {name} in {year}!")
```

Number and string

```
year = 2022
name = "World"
print(f"Hello, {name} in {year}!")
```

```
Hello, World in 2022!
```

pi without rounding

```
import math
print(f'Value of pi: {math.pi}')
```

pi without rounding

```
import math
print(f'Value of pi: {math.pi}')
```

```
Value of pi: 3.141592653589793
```

pi rounding to three places after the decimal point

```
import math
print(f'Value of pi: {math.pi:.3f}')
```

pi rounding to three places after the decimal point

```
import math
print(f'Value of pi: {math.pi:.3f}')
```

```
Value of pi: 3.142
```

Yellow pages printing (before f-string)

```
yellow_pages = {'Bob':125422, 'Tommy':88888888, 'Alexandra':7}
for name, phone in yellow_pages.items():
    print(name + ' >>> ' + str(phone))
```

Yellow pages printing (before f-string)

```
yellow_pages = {'Bob':125422, 'Tommy':88888888, 'Alexandra':7}
for name, phone in yellow_pages.items():
    print(name + ' >>> ' + str(phone))
```

```
Bob >>> 125422
Tommy >>> 88888888
Alexandra >>> 7
```

Yellow pages printing (after f-string)

```
yellow_pages = {'Bob':125422, 'Tommy':88888888, 'Alexandra':7}
for name, phone in yellow_pages.items():
    print(f'{name:10} >>> {phone:10d}')
```

Yellow pages printing (after f-string)

```
yellow_pages = {'Bob':125422, 'Tommy':88888888, 'Alexandra':7}
for name, phone in yellow_pages.items():
    print(f'{name:10} >>> {phone:10d}')
```

```
Bob        >>>     125422
Tommy      >>>   88888888
Alexandra  >>>          7
```

f-string evaluation

```
year = 2022
print(f"Three year from {year} is {year+3} ")
```

f-string evaluation

```
year = 2022
print(f"Three year from {year} is {year+3} ")
```

```
Three year from 2022 is 2025
```

Call functions in f-string

```
import math
print(f"Four squared is: {math.pow(4,2)}")
```

Call functions in f-string

```
import math
print(f"Four squared is: {math.pow(4,2)}")
```

```
Four squared is: 16.0
```

Quiz: Square table from 1 to 10

```
# create a square table as shown below
# you must use f-string that you just learned
```

```
 1 *  1 =   1
 2 *  2 =   4
 3 *  3 =   9
 4 *  4 =  16
 5 *  5 =  25
 6 *  6 =  36
 7 *  7 =  49
 8 *  8 =  64
 9 *  9 =  81
10 * 10 = 100
```

Contoh Soal

Terima sebuah string dari user, cetak segitiga yang tersusun dari karakter-karakter stringnya dari awal sampai akhir.

Contoh:

Masukkan string: "PYTHON"

P

PY

PYT

PYTH

PYTHO

PYTHON

Contoh Soal

Terima sebuah string dari user, cetak string yang sama tapi semua kemunculan huruf vocalnya dihilangkan.

Contoh:
Masukkan pesan: "Halo, apa kabar?"
Hl, p kbr?

Q&A Session