Chapter 2. Input, Processing and Output

Starting out with Python

Example Source Files

Kyu Lee. Ph. D.

Computer Science

The first python program

Example program to print some values

- Print the string values using the double quotation or single quotation
    - `print (“Hello world”)`
    - `print (“How to print a special character like ‘ “)`
    - `print (" ' ")`
    - `print (' " ')`
    - `# print (" " ") error`
    - `# print (' ' ') error`

Comments

- Notes of explanation that document lines or sections of a program.
- Python interpreter ignores the comments

- Single line comment
  - #
- Multiple lines
  - “””
  - “””

```
"""
Multiple line comments
This section is comment box
All lines are ignored by Python Interpreter
"""
print ("Code starts from here")
```

Variables

A variable is a name that a value stored in the real computer memory

Variables

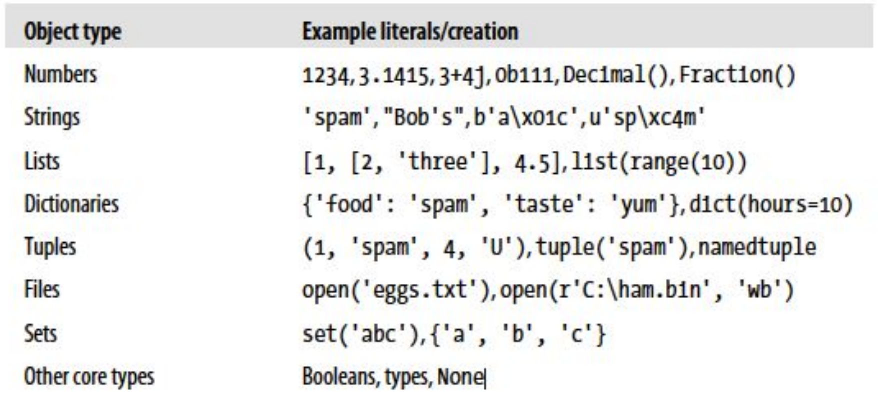
- A variable is a name that represents a value in the computer's memory.

- Creating Variables with Assignment Statement
  - typical example of variables

```
num1 = 10
print (num1)
```

- Python Built-in Data Types
  - https://docs.python.org/3/library/stdtypes.html

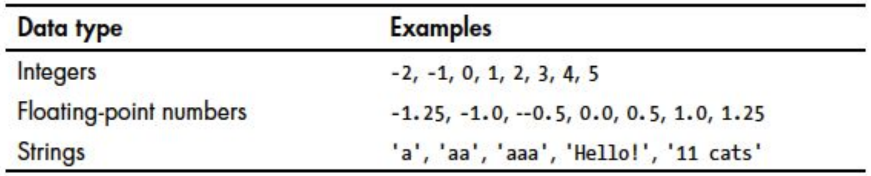
Built-in type preview



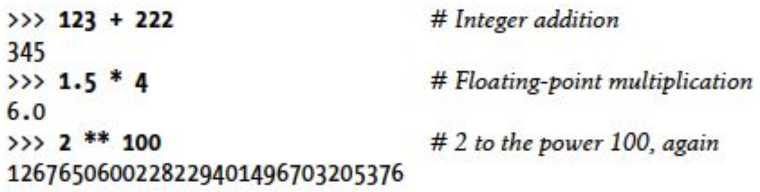
Object type	Example literals/creation
Numbers	1234, 3.1415, 3+4j, 0b111, Decimal(), Fraction()
Strings	'spam', "Bob's", b'a\x01c', u'sp\xc4m'
Lists	[1, [2, 'three'], 4.5], list(range(10))
Dictionaries	{'food': 'spam', 'taste': 'yum'}, dict(hours=10)
Tuples	(1, 'spam', 4, 'U'), tuple('spam'), namedtuple
Files	open('eggs.txt'), open(r'C:\ham.bin', 'wb')
Sets	set('abc'), {'a', 'b', 'c'}
Other core types	Booleans, types, None

Numbers

- Common Data Types



Data type	Examples
Integers	-2, -1, 0, 1, 2, 3, 4, 5
Floating-point numbers	-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25
Strings	'a', 'aa', 'aaa', 'Hello!', '11 cats'

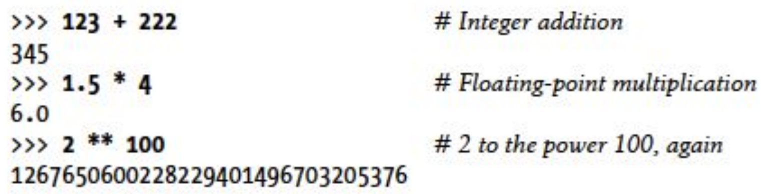


```
>>> 123 + 222                # Integer addition
345
>>> 1.5 * 4                  # Floating-point multiplication
6.0
>>> 2 ** 100                 # 2 to the power 100, again
1267650600228229401496703205376
```

In python 3, there is no limit for the integer number values.
Value of an integer is not restricted by the number of bits and can expand to the limit of the available memory

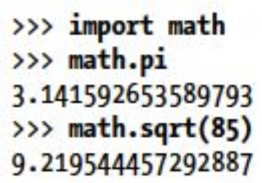
Numbers

- Numbers Examples

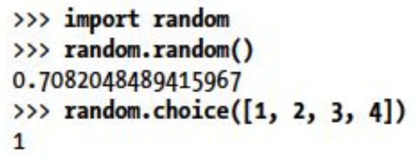


```
>>> 123 + 222                    # Integer addition
345
>>> 1.5 * 4                      # Floating-point multiplication
6.0
>>> 2 ** 100                     # 2 to the power 100, again
1267650600228229401496703205376
```

In python 3, there is no limit for the integer number values.
Value of an integer is not restricted by the number of bits and can expand to the limit of the available memory



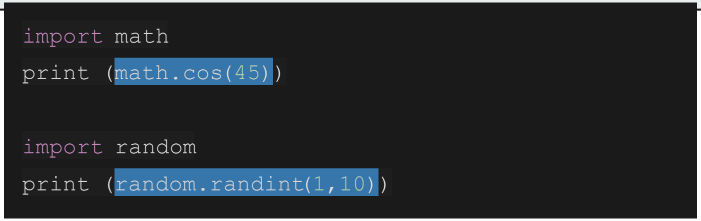
```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(85)
9.219544457292887
```



```
>>> import random
>>> random.random()
0.7082048489415967
>>> random.choice([1, 2, 3, 4])
1
```

modules: a packages of additional tools that we can import to use

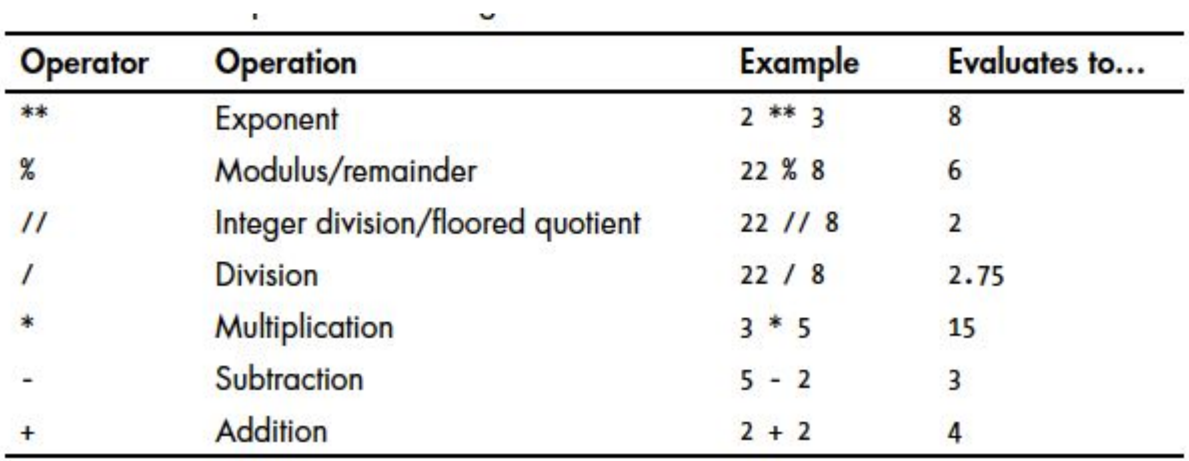
The math module contains more advanced tools for math



```
import math
print (math.cos(45))

import random
print (random.randint(1,10))
```

Operators for Numbers

- Math Operators (Preceden order from highest to lowest)



Operator	Operation	Example	Evaluates to...
**	Exponent	2 ** 3	8
%	Modulus/remainder	22 % 8	6
//	Integer division/floored quotient	22 // 8	2
/	Division	22 / 8	2.75
*	Multiplication	3 * 5	15
-	Subtraction	5 - 2	3
+	Addition	2 + 2	4

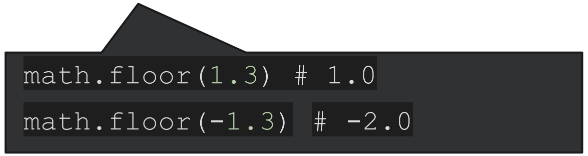
Precedence

$\uparrow$ High

$\downarrow$ Low

Operators for Numbers

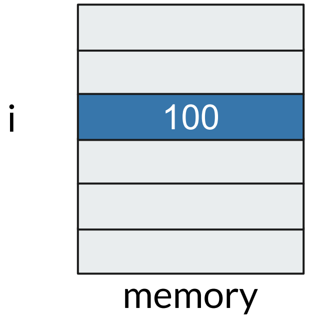
- Division (float) vs Division(floor)
    - 5 / 2
        - 2.5
    - 5 // 2
        - 2
    - The // operator works like this:
        - When the result is positive, it is truncated, which means that its fractional part is thrown away.
        - When the result is negative, it is rounded away from zero to the nearest integer.
    - -5 // 2
        - -3
    - 10 // -3
        - -4



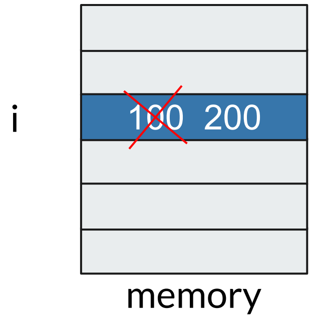
```
math.floor(1.3)  # 1.0
math.floor(-1.3)  # -2.0
```

Assignment Operator =

- i = 100
  - means that the integer value 100 is stored in a memory space which is named with the variable ‘i’



i =100
i = 200



Input

Input

- **Built-in `input` function**
  - **reads input from keyboard**
  - Returns the data as a string
- Format
  - `variable = input(prompt)`
  - `prompt` is typically a string instructing user to enter a value

Input Example Code

- Code

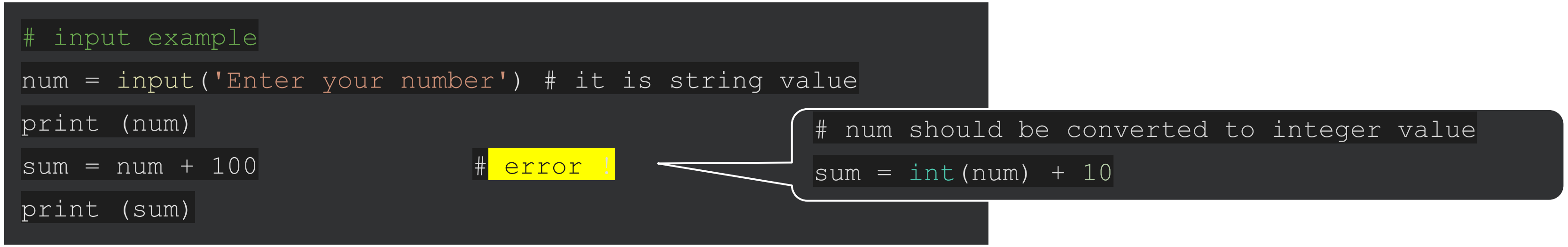
```
# input example

num = input('Enter your number')
print (num)
```

- output

```
Enter your number 100
100
```

Input returns a string value

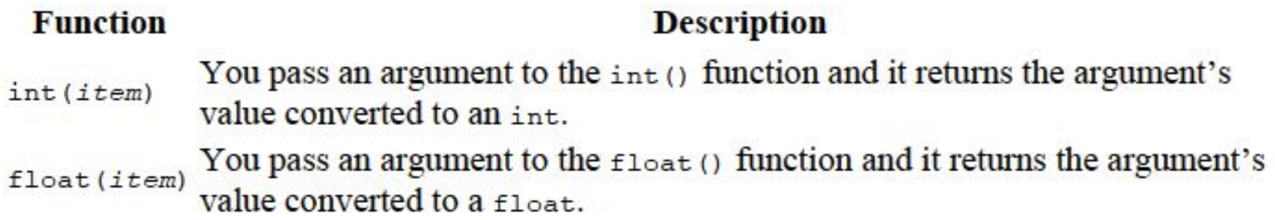
- Code



```
# input example
num = input('Enter your number') # it is string value
print (num)
sum = num + 100                  # error !
print (sum)
```

```
# num should be converted to integer value
sum = int(num) + 10
```

- Data Conversion Functions



Function	Description
`int(item)`	You pass an argument to the `int()` function and it returns the argument's value converted to an `int`.
`float(item)`	You pass an argument to the `float()` function and it returns the argument's value converted to a `float`.

Exercise Lab 1: Calculating the sales price

- Problem definition
  - Make a program that calculates the sales price.
    - The stores sales the product with the particular amount of percent off price.
  - Input:
    - Original Price (e.g, 100)
    - Discount Rate(percentage, e.g, 20 means 20%)
  - Output:
    - Original Price
    - Discount Amount
    - Final Price

Use the same variable names to pass the test program

Please complete the following steps

1. Input an integer value for the original price(dollar). Save it as a variable "**original_price** "
2. Input an integer value for the discount rate(**percentage**). Save it as a variable "**rate**"
3. Calculate the discount amount(dollar). Save it as a "**discount_amount** "
4. Calculate the final price. Save it as a “**final_price** ”
5. Print original price
6. Print discount amount
7. Print the final sale price

Input

```
200
20
```

Expected Output

```
Original Price :        $200
Discount Amount :       $40
The final price :       $160
```

Exercise Lab 2: Calculating the salary

- Problem definition
  - Write a program that calculates and displays an employee's total wages for the week.
    - The regular hours for the work week are 40,
    - and any hours worked over 40 are considered overtime.
    - The employee earns $18.25 per hour for regular hours and $27.78 per hour for overtime hours.
    - Ask for the employee's work hour and then save it as a variable "`workhours`"
    - For example,
      - The employee has worked 50 hours this week.

We assume that this "`workhours`" is greater than 40

- Input
  - one integer value for work hours
- Output (print in separate lines)
  - Regular charge
  - Overtime charge
  - total wages

```
[Run Example]
Enter your work hours: 50
Regular Charge: 730.0
Overtime Charge: 277.80
Total wage : 1007.80
```

input

output

```
1.   Assign the given values to the variables
     reg_hours = 40
     reg_rate = 18.25
     ov_rate = 27.78
2.   Calculate
      a.   the overtime_hours
      b.   the regular_wage
      c.   the overtime_wage
3.   Calculate and print the total_wage
```

Use the same variable names to pass the test

More About Data Output

1

Formatted Output

format(value, format)

2

%-formatting

“ %s %s” % (val1, val2)

3

String Formatting Method

“string value {0}”.format(value)

4

F-String PEP 498

f”String Value {value:format}”

1. Formatting

1. Formatting

- Simple Formatting Numbers

```
print (format(123456789, '15d'))
print (format(12345.678, '.2f'))
      123456789
12345.68
```

- Formatting in Scientific Notation

```
print (format(12345.6789, 'e'))
print (format(12345.6789, '.2e'))
1.234568e+04
1.23e+04
```

- Inserting Comma Separator

```
#? Inserting comma separator

print (format(123456789.12345 , ',.2f'))
123,456,789.12
```

- Floating-point numbers as a percentage

```
print (format(12345.6789, '.2%'))
1234567.89%
```

Suppressing the print function’s ending newline

```
print ('one')
print ('two')
print ('three')

print ('one', end=' ')
print ('two', end=' ')
print ('three', end=' ')
```

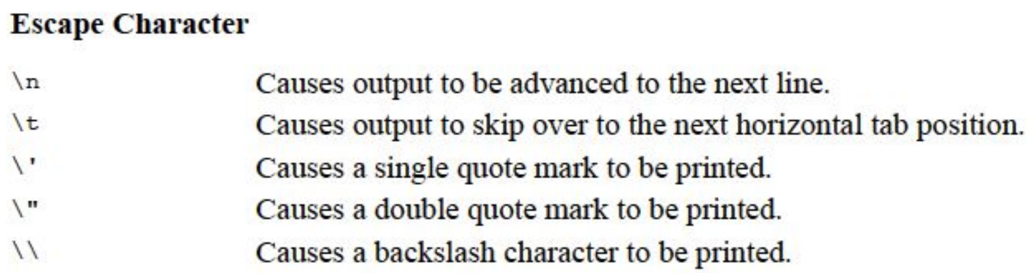
- Item separator

```
#? Specifying an Item Separator
print ('one', 'two', 'three', sep='/')
```

Escape Characters

- New Line

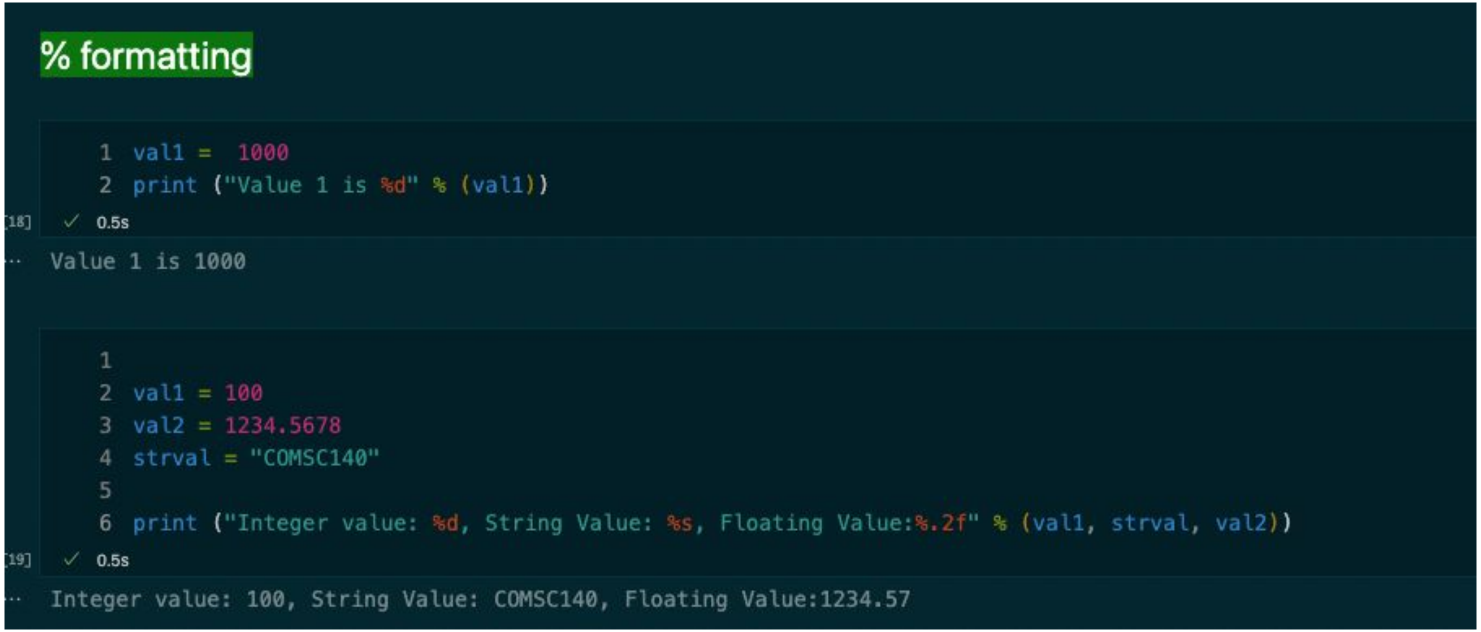
```
#? New Line Char.
print ('one\n two \n three \n')
```



Escape Character	
`\n`	Causes output to be advanced to the next line.
`\t`	Causes output to skip over to the next horizontal tab position.
`\'`	Causes a single quote mark to be printed.
`\"`	Causes a double quote mark to be printed.
`\\`	Causes a backslash character to be printed.

2. % formatting

2. % formatting



% formatting

```
val1 =  1000
print ("Value 1 is %d" % (val1))
```

[18] ✓ 0.5s

```
Value 1 is 1000
```

```

val1 = 100
val2 = 1234.5678
strval = "COMSC140"

print ("Integer value: %d, String Value: %s, Floating Value:%.2f" % (val1, strval, val2))
```

[19] ✓ 0.5s

```
Integer value: 100, String Value: COMSC140, Floating Value:1234.57
```

3. String Formatting Method

3. String formatting method

- string format method

```
>>> print('{0} and {1}'.format('spam', 'eggs'))
spam and eggs
>>> print('{1} and {0}'.format('spam', 'eggs'))
 eggs and spam
```

```
>>> print('This {food} is {adjective}.'.format(
...       food='spam', adjective='absolutely horrible'))
 This spam is absolutely horrible.
```

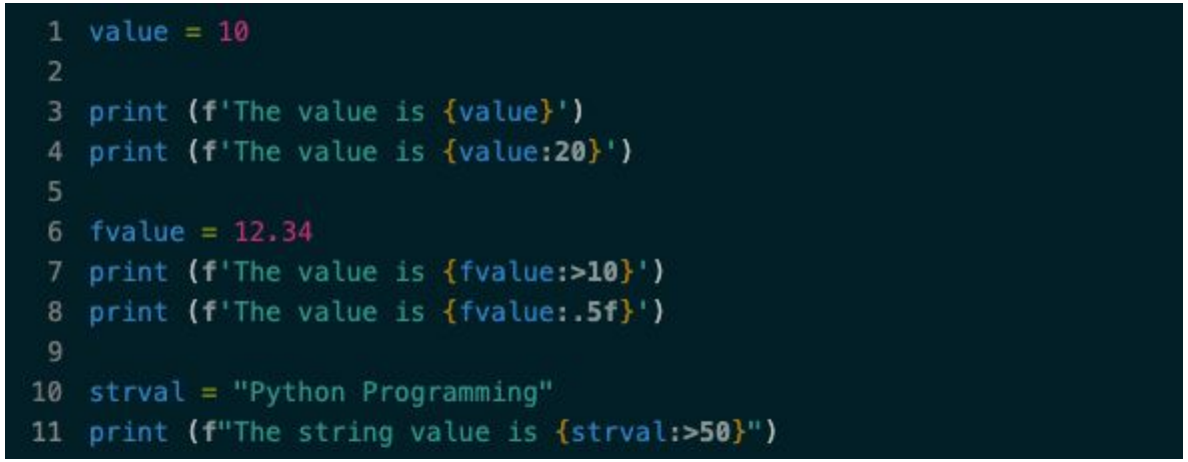
```
>>> print('{0:2d} {1:3d} {2:4d}'.format(num1, num2, num3))
```

```
>>> print('The average is {0:.2f}'.format(59.999999))
The average is 60.00
```

4. PEP 498; f-string

4. f-string

https://peps.python.org/pep-0498/



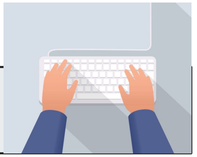
```python
value = 10

print (f'The value is {value}')
print (f'The value is {value:20}')

fvalue = 12.34
print (f'The value is {fvalue:>10}')
print (f'The value is {fvalue:.5f}')

strval = "Python Programming"
print (f"The string value is {strval:>50}")
```

Exercise Lab 3: Calculating an Average

- Problem definition
  - Determine the average of a group of values:
    - input all three **integer** values(user input) then divide the sum by the number of values.
    - print the `total` and `average`(float) of the values

Use the same variable names to pass the test

```
1.  Input the first integer value and assign it to the variable
    'val1'
2.  Input the second integer value and assign it to the variable
    'val2'
3.  Input the third integer value and assign it to the variable
    'val3'
4.  Get sum of three values and assign it to the variable total
5.  Get average and save it as average
6.  Print three values in a line
7.  Print the total
8.  Print the average with two fractional digits (ex, 123.45)
```



Input

```
100
90
110
```

Expected Output

```
Values: 100 90 110
Total:          300
Average :       100.00
```

```
print ('Average: \t {0:.2f} ' .format(avg))
or
print (f'Average: \t {avg:.2f}')
```

String

String Method in Python Documentation
W3 School Examples

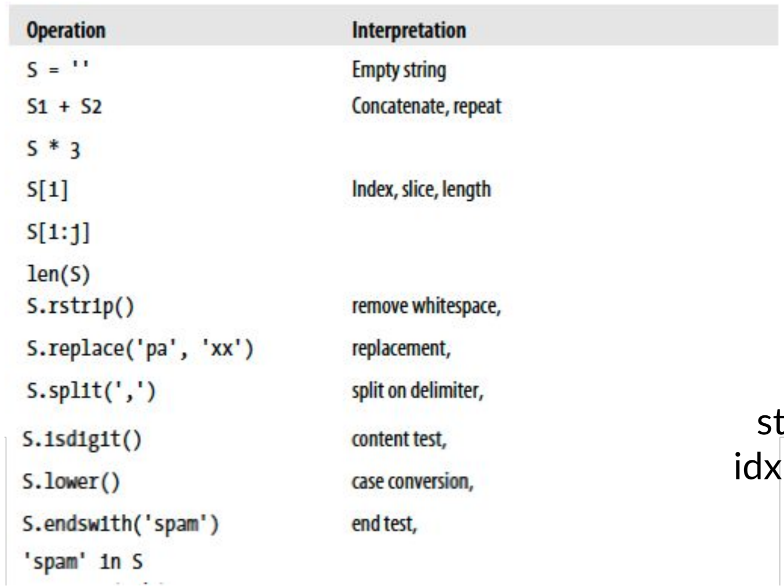
String

- You can use double or single quotation for string values
  - 'Alice'
  - "Bob"
- String Concatenation
  - 'Alice' + 'Bob'
- Sting Replication
  - 'Alice' * 3   #AliceAliceAlice
- Errors
  - 'Alice' + 42
  - 'Alice' * 'Bob'

```
FirstName = "Kyu"
LastName = "Lee"

print (FirstName + LastName)
print (len(FirstName))
```

common string operations

- Useful operations

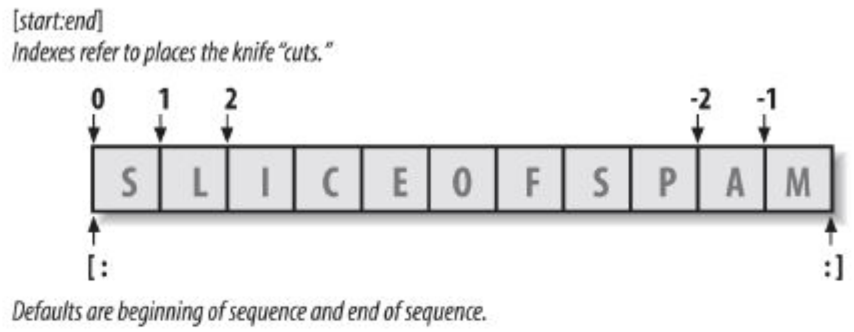


Operation	Interpretation
`S = ''`	Empty string
`S1 + S2`	Concatenate, repeat
`S * 3`	
`S[i]`	Index, slice, length
`S[i:j]`	
`len(S)`	
`S.rstrip()`	remove whitespace,
`S.replace('pa', 'xx')`	replacement,
`S.split(',')`	split on delimiter,
`S.isdigit()`	content test,
`S.lower()`	case conversion,
`S.endswith('spam')`	end test,
`'spam' in S`	

str = ‘python’
idx = str.find(‘t’)

String: Indexing and Slicing

- Offsets and slices:
  - positive offsets start from the left end (offset 0 is the first item), and
  - negatives count back from the right end (offset −1 is the last item).
  - Either kind of offset can be used to give positions in indexing and slicing operations.


[start:end]
Indexes refer to places the knife "cuts."
0
1
2
-2
-1
S L I C E O F S P A M
[:
:]
Defaults are beginning of sequence and end of sequence.

Exercise Lab 4: String

- Here is the original string.
  - `original_str = “Python Programming”`

1) Extract “Python” from original string with index slicing
   - `sub1` = original_str[index_slicing]   # extract the first substring ‘Python’
2) Extract “Programming” from original string with index slicing
   - `sub2` = original_str[index_slicing]   # extract the second substring ‘Programming’
3) Using the string concatenation(‘+’),

   merge two substrings `sub1` and `sub2` and save it to `“merged_str”`

   ```
   merge_str = sub2 + sub1
   “Programming Python”
   ```

Use the same variable names to pass the test

Input: none

Expected Output

```
Programming
Python
Programming Python
```

```
print (sub2)
print (sub1)
print (merged_str)
```

See the example code
https://github.com/LPC-CSDept/CS07/blob/main/Chap02/ch02.ipynb

Assignments

Introduction to Python Programming

See the example code
https://github.com/LPC-CSDept/CS07/blob/main/Chap02/ch02.ipynb

Guide to submit your program assignment

- Submit your programs to Github classroom and documents to Canvas
  1. Github classroom **Link** to your program file
  2. **Elaboration** on your program code and algorithm
     a. Input/Output Description
     b. Explanation of all variables
     c. Flow Chart
     d. Errors and Lessons you experienced.
        i. The kinds of errors
        ii. How to fix those errors

Example Documents for Programming Assignment

Assignment 1

- Male and Female Percentages
  - Write a program that asks the user for the number of **males, females and non-binary** registered in a class.
  - The program should display **the percentage of males, females and non-binary** in the class.

- Variable Names that should be used
  - `m_perc: percentage of male students`
  - `f_perc: percentage of female students`
  - `nb_perc: percentage of non-binary students`
- Input
  - number of male students: 40
  - number of female students: 40
  - number of non-binary students: 20
- Expected Output
  - The total number of students: 100
  - The number of males, females and non-binary 40 40 20
  - The percentage of males, females and non-binary 40.00% 40.00% 20.00%

Use the **same variable names** to save your results

```
print (f'Some message \t {m_perc:.2f} \t {f_perc:.2f} \t {nb_perc:.2f}')
```

Assignment 2

- Celsius to Fahrenheit Temperature Converter
  - Write a program that converts Celsius temperatures to Fahrenheit temperatures. The formula is as follows:

$$F = \frac{9}{5}C + 32$$

```
fahrenheit
celsius
```

Use the same variable names to save your results

  - The program should ask the user to enter a temperature in Celsius, then display the temperature converted to Fahrenheit(two fractional values).

- Input
  - 23
- Expected Output
  - Farenheit: 73.40

```
print (f'Some message \t {fahrenheit:.2f}')
```