CSIT110

Fundamental Programming with Python

Input Output

Basic Data Types

Goh X. Y.

In this lecture

- Terminology
- Variables & Data types
- Convert between data types
- Input
  - `input()` function
- Display to console
  - `print()` function

Terminology

Code

Variable

Comments are not code: `# Comments start with a hash sign`

Syntax

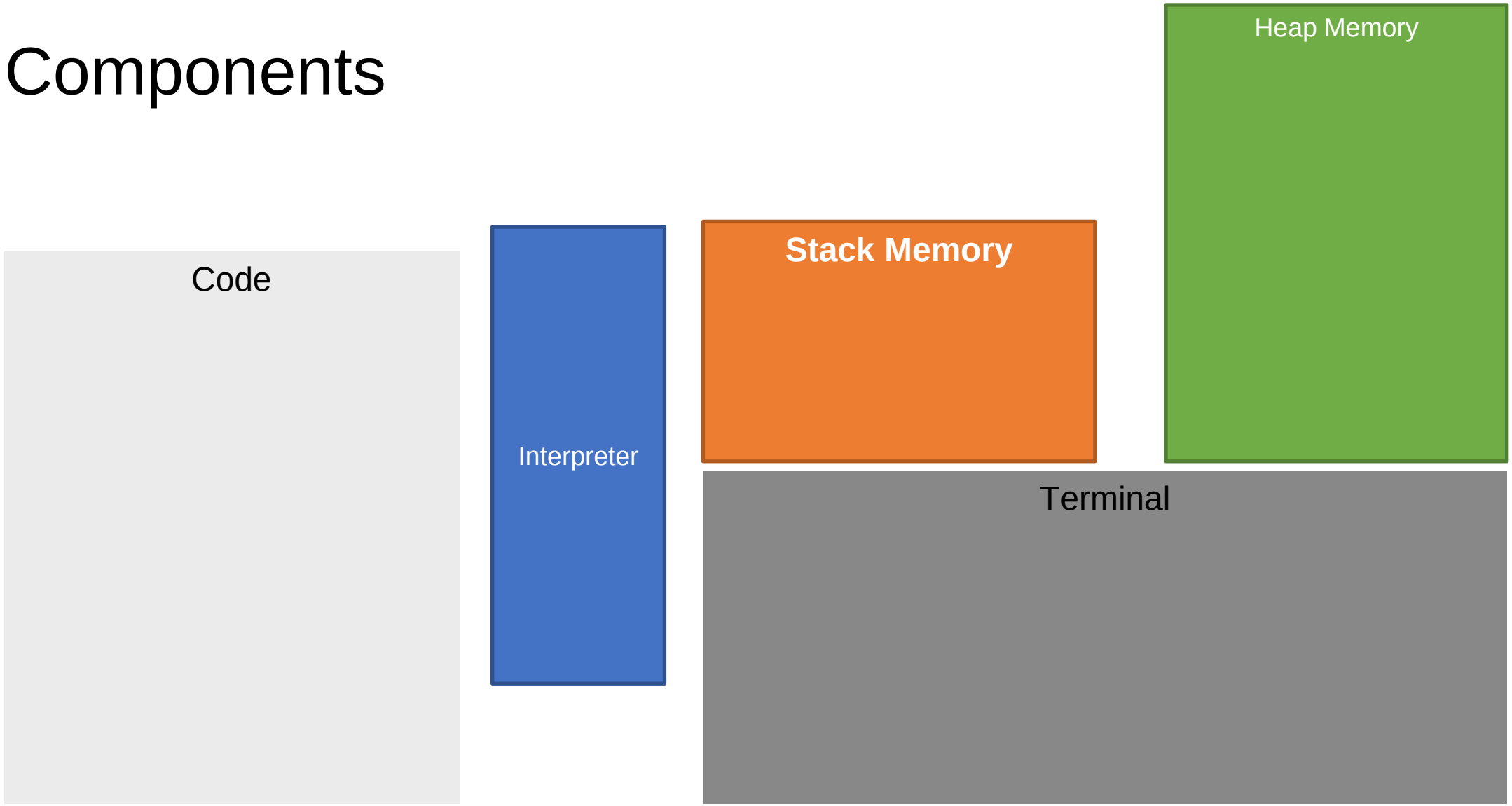
Debug

Runtime

Source file: .py

Instantiate

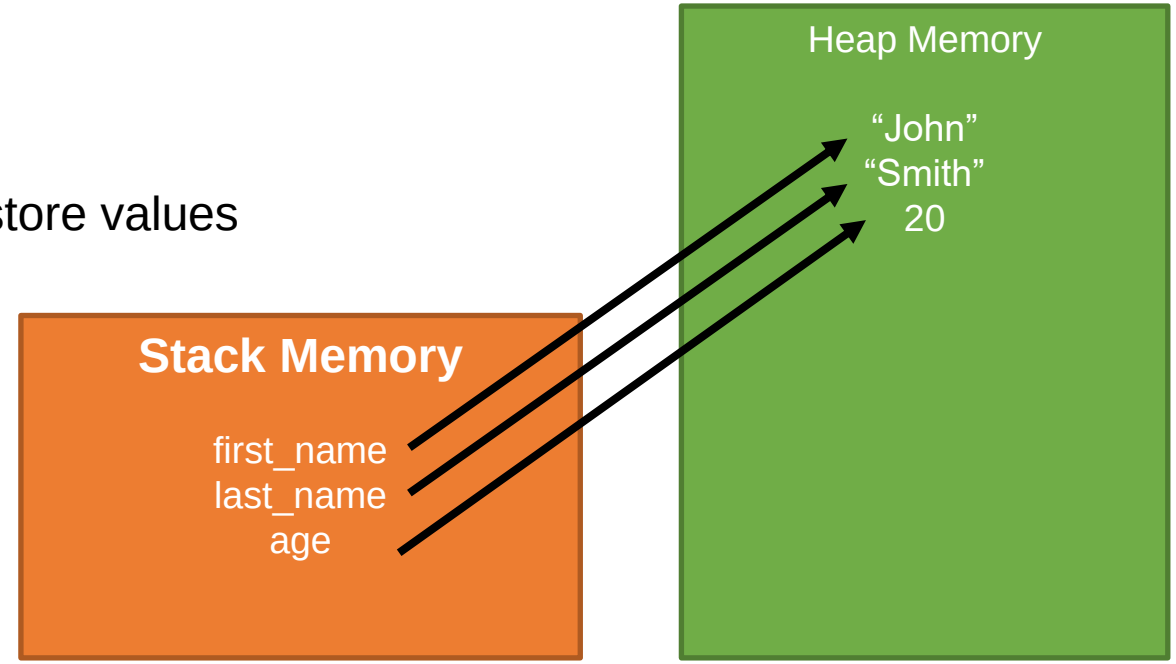
Components



Variables

Variables are reserved memory locations to store values

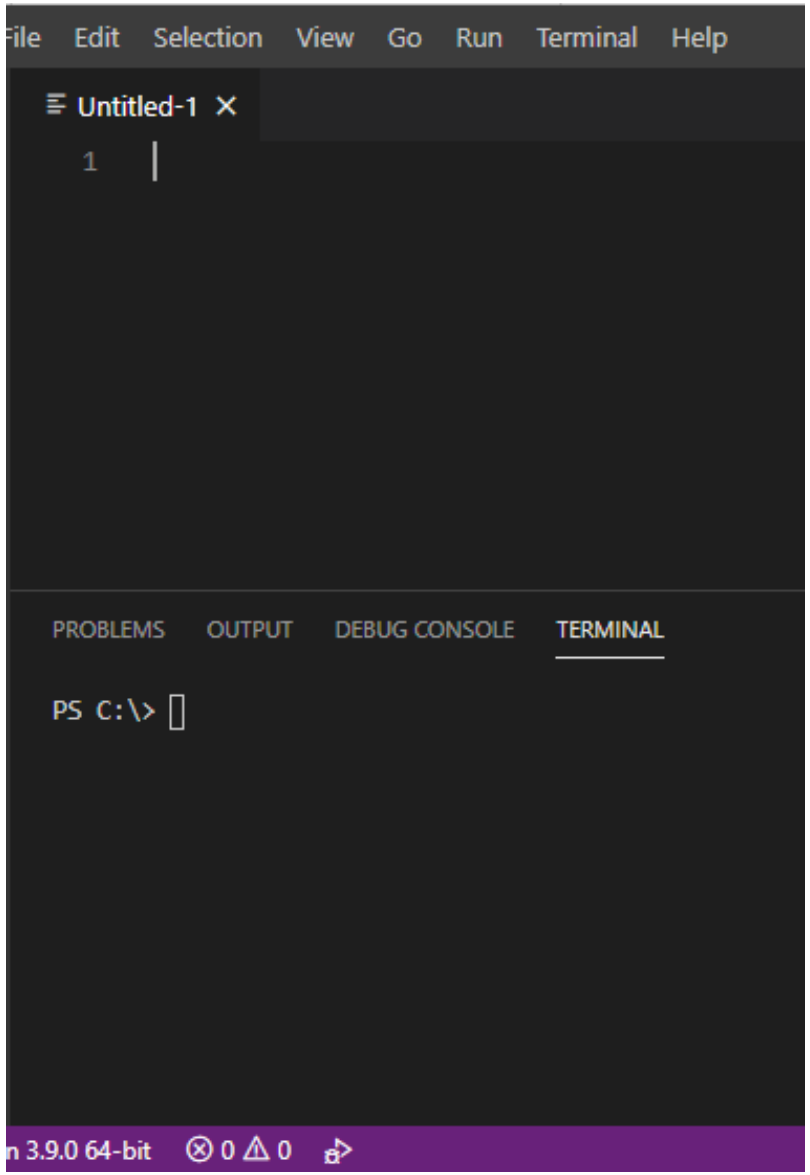
- Name
- Content / values



Use = to assign the values to the variable name

```
first_name = "John"
last_name = "Smith"
age = 20
```

Interpreter



Text in the box is what you will see in the terminal

Code example in the terminal

```
>>>x = 1 + 2
>>>x
3
>>> x + 3
6
```

Code example in a .py file

```
x = 1 + 2
print(x)
print(x+3)
```

```
3
6
```

Variables

ALWAYS use variables with **meaningful names** and **correct data types**

```
first_name = "John"
last_name = "Smith"
age = 20
```

NEVER use variable like a, b, c, x, y, z, or blah...

```
n1 = "John"
n2 = "Smith"
a = 20
```

Variables & Data Types

Each variable has a data type.

Basic Data Types

Numeric

- Integer e.g. 1,2,3,4
- Float e.g. 1.234
- Complex Number 12+34j

Str

Boolean

Date

NoneType

Variables & Data Types

Checking data type:

```
type(...variable_name...)
```

Output to console:

```
print(...variable_name...)
```

Data Types - Integer

Integer: whole numbers

```
age = 20
temperature = -5
credit_point = 6
type(age)
print(type(temperature))
print(type(credit_point))
```

```
<class 'int'>
<class 'int'>
```

Data Types - Float

Float: decimal numbers

```
price = 30.5
interest_rate = 3.18
print(type(price))
print(type(interest_rate))
```

```
<class 'float'>
```

Some useful math constants

```
import math
PI = math.pi
EULARS_NUMBER = math.e
TAU = math.tau
print(PI)
print(EULARS_NUMBER)
print(TAU)
```

```
3.141592653589793
2.718281828459045
6.283185307179586
```

Data Type - Complex

```
>>>impedance = 1+0.5j
>>>impedance.real
1.0
>>> impedance.imag
0.5
```

Supports mathematical operations

- Addition
- Subtraction
- Multiplication and division e.g. `j*j = -1`

Numeric Data Type

All numeric types (except complex) support the following operations

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	floored quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`float(x)`	*x* converted to floating point
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

→ Rounded to -inf

```
  1 //  2  >>>  0
(-1)//  2  >>> -1
  1 //(-2) >>> -1
(-1)//(-2) >>>  0
```

Note: the order of operator precedence >> docs

Numeric Data Type

Constructors: `int()`, `float()`, `complex()`

There are many ways to instantiate a float!

```
>>> float('+1.23')
1.23
>>> float('   -12345\n')
-12345.0
>>> float('1e-003')
0.001
>>> float('+1E6')
1000000.0
>>> float('-Infinity')
-inf
```

Numeric Data Type

Constructors: **`int(), float(), complex()`**

You can use the constructor to create or convert data types

```
>>>int(3.7)
3

>>>float(3)
3.0

>>>complex(1,2)
(1+2j)
```

Data Type - Boolean

bool()

`True` or `False`

```
virus_scan_completed = True
virus_found = False
print(type(virus_scan_completed))
print(type(virus_found))
temperature = -5
temperature_negative = (temperature < 0)
print(temperature_negative)
temperature_positive = (temperature > 0)
print(temperature_positive)
```

```
<class 'bool'>
<class 'bool'>
```

```
True
```

```
False
```

Data Type - Boolean

`bool()`

`True` or `False`

```
print(bool(0))
print(bool(1))
print(bool(0.2))
print(bool(-1))
```

```
False
True
True
True
```

Data Type - Date

Date data type: including year, month, day, (not the time)

```
import datetime
today_date = datetime.date.today()
us_election_2020 = datetime.date(2020, 11, 3)
print(type(today_date))
print(type(us_election_2020))
```

```
<class 'datetime.date'>
<class 'datetime.date'>
```

Data Type - Datetime

Datetime data type: including year, month, day, hour, minute, second, ...

```
import datetime
current_date_time = datetime.datetime.now()
christmas_2020 = datetime.datetime(2020, 12, 25)
random_date_time = datetime.datetime(2000, 12, 20, 14,
     20, 39, 555)
print(type(current_date_time))
print(type(christmas_2020))
print(type(random_date_time))
```

```
<class 'datetime.datetime'>
```

Data Type - Str

Str: Text
using either double quote or single quote

```
first_name = "John"
last_name = 'Snow'
```

String literals

Single quotes e.g. `string1 = 'allows embedded "double" quotes'`

Double quotes e.g. `string2 = "allows embedded 'single' quotes"`

Triple quotes e.g. `string3 = """Three double quotes
Can span multiple lines"""`

e.g. `string4 = ''' three single quotes
works too '''`

Data Type - String

- Concatenation

```
# name details
first_name = "John"
last_name = "Snow"
# use string addition to formulate the full name
full_name = first_name + " " + last_name
# display the full name
print("My name is " + full_name + ".")
```

```
My name is John Snow.
```

Data Type - String

- Multiplication with number

```
# display some silly strings
silly1 = "frog" * 7
silly2 = 5 * "I am Sam"
print(silly1)
print(silly2)
```

```
frogfrogfrogfrogfrogfrogfrog

I am SamI am SamI am SamI am SamI am Sam
```

Our first Python program

```
# My first Python program
print("PPP  Y  Y  TTTTT  H  H  OO  N  N")
print("P  P  Y  Y    T    H  H  O  O  NN  N")
print("PPP    Y    T    HHHH  O  O  N  N  N")
print("P      Y    T    H  H  O  O  N  NN")
print("P      Y    T    H  H  OO  N  N")
# print blank lines
print()
print()
# print greetings
print("Welcome to Python - Class of 2020!")
```

What do you think this program will do?

Write this python code and run it.
See what the code produces.

Our first Python program

```
# print hello and greeting
print("Hello World!")
print('Welcome to Python!')
```

```
# print hello and greeting and silly stuff :-)
print("Hello World!", end="frog")
print("Welcome to Python!", end="cat")
print("How are you?")
```

What is the purpose of

```
print("...")
print('...')
print("...", end="...")
print()
```

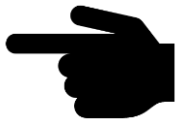
What is wrong with this code?

```
print(Hello World!)
```

Getting input from the user

When we want to ask the user some information, use the `input()` function.

In the `input()` function, we can specify the **prompt**.



The information that the user has entered will be stored in the **variable** as a **str**.

```
# ask the user to enter some information
variable_here = input("Put the prompt here: ")
```

Getting input from the user

Example 1:

```
# ask the user to enter first name and last name
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
# use string addition to formulate the full name
full_name = first_name + " " + last_name
# display the full name
print("My name is " + full_name + ".")
```

```
Enter your first name: Frodo
Enter your last name: Baggins
My name is Frodo Baggins.
```

Getting input from the user

Example 2:

```
# Ask the user to enter 3 subjects
print("You must choose 3 subjects.\n")
subject1 = input("Enter the 1st subject: ")
subject2 = input("Enter the 2nd subject: ")
subject3 = input("Enter the 3rd subject: ")
# Display subjects
print("\nYou have chosen: " + subject1 + ", " + subject2 + ", " +
subject3 + ".")
```

```
You must choose 3 subjects.

Enter the 1st subject: ISIT111
Enter the 2nd subject: MATH101
Enter the 3rd subject: ACCY113

You have chosen: ISIT111, MATH101, ACCY113.
```

Getting input from the user

Example 2:

```
# Ask the user to enter 3 subjects
print("You must choose 3 subjects.\n\n")
subject1 = input("Enter the 1st subject: ")
subject2 = input("Enter the 2nd subject: ")
subject3 = input("Enter the 3rd subject: ")
# Display subjects
print("\nYou have chosen: "
      + subject1 + ", "
      + subject2 + ", "
      + subject3 + ".")
```

Rewrite the code to make it clearer.

When we have a lot of string additions, write it this way make the code clearer!

Convert number into string

```
# A program to display a favourite number

# favourite number
fav_number = 7

# display favourite number
print("My favourite number is " + fav_number)
```

Copy this python code and run it.

You will see that the code cannot run because there is an error.

What is wrong with this code?

Convert number into string

```
# A program to display a favourite number

# favourite number
fav_number = 7

# display favourite number
print("My favourite number is " + fav_number)
```

this is a string

this is a number

Python cannot add a string to a number

(some other programming languages can)

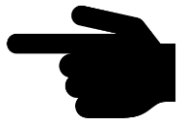
Convert number into string

```
# A program to display a favourite number

# favourite number
fav_number = 7

# display favourite number
print("My favourite number is " + fav_number)
```

1. Convert a number to a str



```
fav_number                    7

str(fav_number)  ------------>  "7"
```

2. now we can do string addition

```
"My favorite number is " + "7"
```

```
My favorite number is 7
```

Convert number into string

```
# A program to display a favourite number

# favourite number
fav_number = 7

# display favourite number
print("My favourite number is " + str(fav_number))
```

```
My favorite number is 7
```

Convert a string to number

```
# Ask the user to enter 2 integers and display the sum
number1 = input("Enter the 1st integer: ")
number2 = input("Enter the 2nd integer: ")
# calculate the sum
number_sum = number1 + number2 # display the sum
print("The sum is " + number_sum)
```

```
Enter the 1st integer: 100
Enter the 2nd integer: 50
The sum is 10050
```

why the output is like this

Convert a string to number

```
# Ask the user to enter 2 integers and display the sum
number1 = input("Enter the 1st integer: ")
number2 = input("Enter the 2nd integer: ")
# calculate the sum
number_sum = number1 + number2 # display the sum
print("The sum is " + number_sum)
```

```
Enter the 1st integer: 100
Enter the 2nd integer: 50
The sum is 10050
```

When we ask the user to enter an input, the input returns a **str**.

```
number1 is a str "100"
number2 is a str "50"
string addition means
number_sum is a str "10050"
```

Convert a string to number

```
# Ask the user to enter 2 integers and display the sum
user_input1 = input("Enter the 1st integer: ")
number1 = int(user_input1)
user_input2 = input("Enter the 2nd integer: ")
number2 = int(user_input2)

# calculate the sum
number_sum = number1 + number2
# display the sum
print("The sum is " + str(number_sum))
```

```
Enter the 1st integer: 100
Enter the 2nd integer: 50
The sum is 150
```

What did we change?

Convert a string to number

```
# Ask the user to enter 2 integers and display the sum
user_input1 = input("Enter the 1st integer: ")
number1 = int(user_input1)
user_input2 = input("Enter the 2nd integer: ")
number2 = int(user_input2)

# calculate the sum
number_sum = number1 + number2
# display the sum
print("The sum is " + str(number_sum))
```

```
user_input1 is a str "100"
number1 is an int

user_input2 is a str "50"
number2 is an int

number addition means number_sum is a number 150
```

```
Enter the 1st integer: 100
Enter the 2nd integer: 50
The sum is 150
```

Convert a string to number

```
# Ask the user to enter 2 integers and display the sum
user_input = input("Enter the 1st integer: ")
number1 = int(user_input)
user_input = input("Enter the 2nd integer: ")
number2 = int(user_input)

# calculate the sum
number_sum = number1 + number2

# display the sum
print("The sum of "
  + str(number1)
  + " and "
  + str(number2)
  + " is "
  + str(number_sum)
)
```

```
Enter the 1st integer: 100
Enter the 2nd integer: 50
The sum is 150
```

We can use just one variable `user_input` to save memory space

Convert a string to a decimal number

```
# Ask the user to enter 2 decimal numbers and display the sum
user_input = input("Enter the 1st number: ")
number1 = float(user_input)

user_input = input("Enter the 2nd number: ")
number2 = float(user_input)

# calculate the sum
number_sum = number1 + number2

# display the sum
print("The sum of "
  + str(number1)
  + " and "
  + str(number2)
  + " is "
  + str(number_sum)
)
```

We use `number1 = float(user_input)` to convert the string `user_input` into a decimal number `number1`

```
Enter the 1st number: 2.5
Enter the 2nd number: 3.1
The sum of 2.5 and 3.1 is 5.6
```

Convert between data types

Convert to a string: `str(...variable_name...)`

```
fav_number          7

str(fav_number)     "7"
```

! str() can be used to convert other data types into string, such as boolean, list, dictionary, etc.

Convert to an integer: `int(...variable_name...)`

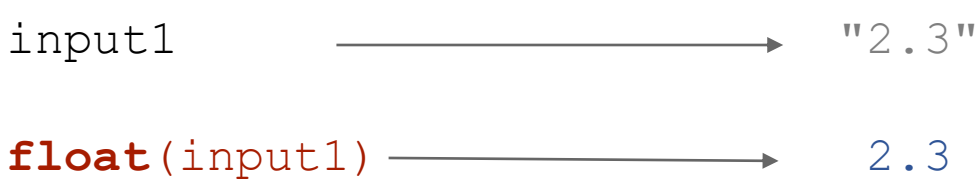
```
user_input = input("Enter an integer: ")
number = int(user_input)
```

```
user_input          ──────────────►  "50"

int(user_input)     ──────────────►  50
```

Convert between data types

Convert to a decimal number: `float(...variable_name...)`



We can also convert integer to float, float to integer, etc...

Convert between data types

Convert between string and date

```
import datetime
# ask the user enter dob in DD/MM/YYYY format
user_input = input("Enter your dob (DD/MM/YYYY): ")
# convert string type to date type
date_format = '%d/%m/%Y'
dob = datetime.datetime.strptime(user_input, date_format).date()
# convert date to string
print("Your dob is " + dob.strftime("%d/%b/%Y"))
print("Your dob is " + dob.strftime("%d-%m-%Y"))
```

```
Enter your dob (DD/MM/YYYY): 26/03/2000
Your dob is 26/Mar/2000
Your dob is 26-03-2000
```

Comments

```
# print blank lines
print()
print()
# print greetings
print("Welcome to Python - Class of 2020!")
```

comment

comment

We can put comments anywhere in the program:

- to **make the program clearer** for people to read and maintain
- to **help people understand** our program better, especially, if our program has a special logic that needs explanation
- comments are not code, so they will NOT be executed

Important programming rules

ALWAYS write comments first, then code.

NEVER write code first, then insert comments.

ALWAYS use variables with **meaningful names**

NEVER use variable like a, b, c, x, y, z, or blah...

Important programming rules

Variable contains data information only

Bad example:

```
subject = "MATH111: Abstract Algebra"
```

The colon (:) is not part of the information and should not be stored in variable.
What if we want to display like this:

```
MATH111 - Abstract Algebra
```

or this:

```
Abstract Algebra (MATH111)
```

Good example:

```
subject_code = "MATH111"
subject_title = "Abstract Algebra"

print(f'{subject_code} - {subject_title}')
print(f'{subject_title} ({subject_code})')
```

Important programming rules

Variable must be in correct data type

Bad example:

```
unit_price = "$10.50"
```

Unit price should be a number, not a string.

Good example:

```
unit_price = 10.50
quantity = 12
cost = unit_price * quantity
```

Important programming rules

Variable must be in correct data type

```
mobile_number = 1231231234
student_number = 1234567
```

Mobile number should be a string, not a number.
Student number should be a string, not a number.

```
mobile_number = "0980980987"
student_number = "0043210"
```

- Prevents data loss
- Stores leading zeros or symbols

Naming Convention

```
first_name = "John"
last_name = "Smith"
full_name = first_name + " " + last_name
fav_number = 7
subject1 = "ISIT111"
subject2 = "MATH101"
subject3 = "ACCY113"
SECOND_PER_MINUTE = 60
minute = 5
second = minute * SECOND_PER_MINUTE
```

ALWAYS use variables with **meaningful names**

`lower_case_with_underscores` for normal variables

`UPPER_CASE_WITH_UNDERSCORES` for constants

Keywords

The following list shows the Python keywords. These are reserved words and we **CANNOT** use them as constant or variable or any other identifier names.

and	elif	if	print
as	else	import	raise
assert	except	in	return
break	exec	is	try
class	finally	lambda	while
continue	for	not	with
def	from	or	yield
del	global	pass	

Any questions?