Python Crash Course

A language that doesn't affect the way you think about programming, is not worth knowing. — Alan J. Perlis

Say “Hello World!”

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

- It is a tradition (since C programming language book) to display “Hello World!” in a computer console (terminal).
- **print** is a Python built-in function name. Each function performs some tasks.
- Here it prints the message (a string) `"Hello World!"` to the console.

Function Call

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

- A function, like a computer, processes its input and performs some tasks.
- A function name followed by a pair of parentheses is a function call – the computer runs/executes the function.
- The string `"Hello World!"` is the input to the print function.
- A string is written inside a pair of double quotes. A pair of single quotes also works: `'Hello World!'`

Python is Simple and Easy

```python
print("Hello World!")
```

Hello World in Java

```java
public class Main
{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

A Computer Can Compute

```
2 + 3
```

It is a good idea to put a space before and after the + operator to make it easy to read.
Python interpreter will ignore those spaces.

Two Execution Modes

- Interactive mode: type code directly inside the Python interpreter
  - The result is displayed directly in the console
  - Convenient for small code snippets

```
>>> 2 + 3
5
```

- Script mode: write a script file such as “calculate.py” and run it with “python3 calculate.py”
  - Good for non-trivial code and real applications
  - To see the result, you need the built-in “print” function to display it in the console like “print(2 + 3)

More Computation with Comments

```
# this is a comment, ignored by Python Script
1 + 2

5 - 3

3 * 5 - 2

7 / 5

# exponential
2**10

# quotient
7 // 5

# remainder
7 % 5
```

Operation Order

```
# use parenthesis for operation order
# they are optional here but better
(3 * 5) - 2

# subtraction before multiplication
3 * (5 - 2)

# not clear
2**3**2

# no ambiguity, pay attention to code format
(2**3) ** 2

# no ambiguity
2 ** (3**2)
```

Complex Math

- Python put more math functions in the “math” module.
- You need to import it first before use.
- A function may return a value that can be used as an input to other functions.

```
import math

print(math.sqrt(2))

print(math.log(2))
```

Python Functions

- Built-in: they are essential functions that are part of the Python programming language.
- Standard library: these are common functions that come with Python installation but need ”import” before use .
- Installation packages: you need to download and install them online first, then ”import” before use. For example, all AI packages.

Built-in Functions

- Python has about 80 built-in functions such as “print”, “input”, “abs”, “sum” etc., for the very basic programming tasks required by almost all programs.
- You can use these functions directly.
- The operators such as “+”, “-”, “*” etc. are another type of built-in functions that are presented as natural math operations.
- List of built-in functions: https://docs.python.org/3/library/functions.html

Standard Library

- Python provides more functions in so-called standard library.
- Functions are organized into service modules such as “math”, “statistics”, “os”, “time” etc., that provide additional common functions.
- You need to import the module first to use it functions.
- You call the function using its module prefix such as “math.sqrt(2)”.
- Documentation: https://docs.python.org/3/library/

Comparison

```
import math

# the comparison result is a Boolean value
# either True or False

3 > 2

3 == 2

3 != 2

(7 / 5) >= 1

math.sqrt(3) > 2

# can be chained
2 > math.sqrt(3) > math.sqrt(2)
```

Boolean (Logical) Operations

```
not (3 == 2)

not (3 != 2)

(3 > 2) and ((7 / 5) > 3)

(3 == 2) or (3 >= 2)
```

CPU and ALU

- A core component of a CPU is called ALU: Arithmetic and Logic Unit
- Essentially, a computer performs arithmetic and logic operations on binary data.
- All complex functions such as scientific calculation, image processing, and AI are based on arithmetic and logic operations.

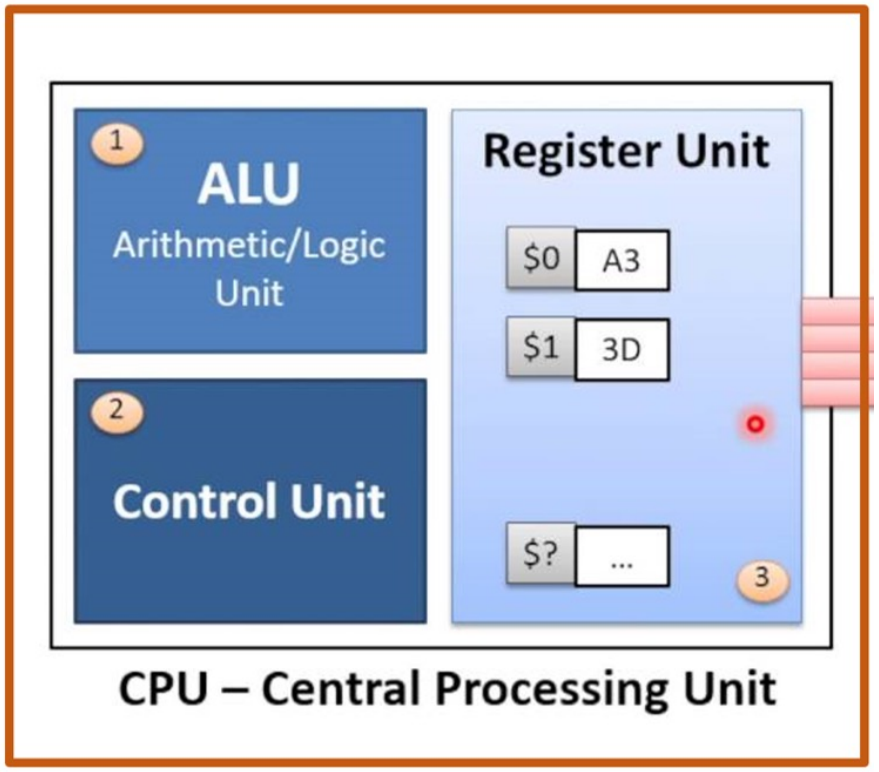

Image Source: https://oer-studentresources.gesci.org/wp-content/courses/Computer/CS-F1-cpu/the_cu_the_alu_and_the_register.html

Literals Are Not Enough

- So far, the operation data are literals such as numbers: `5, 3` or text `"Hello World!"`
- But you need something when you want to
  - Name a data !!!
  - Store the result of an operation
  - Use a short reference in multiple places
  - Refer a non-existing data from input

Variables

- A Python variable is a symbolic name that is a reference to a data.
- Examples:
  - Name a data: `pi = 3.141592653589793`
  - Store the result of an operation: `diameter = 2 * radius`
  - Use a short reference in multiple places:

    ```
    circumference = pi * diameter
    ```
  - Refer a non-existing data from input:

    ```
    radius_input = input("Please input radius: ")
    ```

A Simple Program

```
pi = 3.14159

# display a message and take user input
radius_input = input("Please input radius: ")

# convert input string into a float number
radius = float(radius_input)

diameter = 2 * radius
circumference = pi * diameter

print(circumference)
```

A Python Program

- You create a program by writing a code/script file.
- Python code file has a .py postfix.
- A script file usually has multiple lines.
- Python runs the script file line by line, from top to the bottom.
- IDE (Integrated Development Environment) such as VS Code provides syntax highlight and checks code errors.

Better Output

- You can use f-string (formatted string literals) to create a text string from variables.
- An f-string is created by prefixing the string with `f` or `F` and writing variable as {variable}.
- For example:
  ```
  f"The circumference of radius {radius} is {circumference}."
  ```
- You can also add format modifiers:
  - ```
    f"The circumference of radius {radius:.2f} is
    {circumference:.4f}."
    ```
  - The `:.2f` is a format modifier that shows 2 places after the decimal.
    ```

A Better Version

```
import math

# display a message and take user input
radius_input = input("Please input radius: ")

# convert input text into a float number
radius = float(radius_input)

diameter = 2 * radius

# Python has a predefined pi variable in the math module
circumference = math.pi * diameter

print(f"The circumference of radius {radius:.2f} is {circumference:.4f}.")
```

Life Is Not Linear

```
# display a message and take user input
score_input = input("Please input your
score: ")
# convert input text into an integer
score = int(score_input)

if score >= 900:
    grade = "A"
elif score >= 800:
    grade = "B"
else:
    grade = "C"

result = f"Grade is {grade}."
print(result)
```

Python uses indentation (4 spaces) to mark a code block. Python runs a code block if its above condition is True.

Go back to same level as the if statement.

You Repeat Yourself a Lot

```
score_input = input("Please input your score: ")
score = int(score_input)

while score < 900:
    print("Plese keep studying and practing....")
    score_input = input("Please input new score: ")
    score = int(score_input)

print("Great, you made it!")
```

Exit the loop if the condition is False

A code block may have more than one lines of code at the same indentation level. They are executed top-down.

It is Turing Complete

- Variables
- Sequential execution
- Branch (if else)
- Loop (while …)

Real Word Data is Complex

- A class has 50 students.
- Each student has name, age, major, and GPA.
- An AI model may use billions of record.
- …

- You want to represent the above compound data.

List

- A sequence of values can be written as a list of comma separated items: in a pair of square brackets.

```
scores = [930, 790, 367, 827]
grades = ["A", "B", "C", "D", "F"]
```

- You can access an item by its index number in a pair of square brackets, starting from 0.

```
print(scores[0])
print(grades[4])
```

- If the index is out of range, your code crashes with an IndexError exception

```
print(grades[6])
```

For Loop: Access Every List Item

```
scores = [930, 790, 367, 827]

# in each iteration of this for loop
# the score is assigned the next value in the list
# starting from index 0 to the last one.
for score in scores:
    if score >= 900:
        print(f"Great, you got an A.")
else:
    difference = 900 - score
    print(f"You need {difference} points to get an A.")

print("Done.")
```

Dictionary: Associate a Key with a Value

```
# a dictionary has a comma-separated list of key:value
pairs within the braces
student = {"Name": "Alice", "Age": 20, "Major": "IS"}

# use a key to read or change a dictionary value
print(student["Name"])
student["Age"] = 21
print(student["Age"])

# access every key and its value
for key in student.keys():
    message = f"Key {key} has a value of {student[key]}"
    print(message)
```

- ***Dictionary is the most important data structure used in Python implementation.***
- ***The more you know dictionary, the more you understand Python.***

Same Logic, Many Use

Too many redundant code.
It is not clear what's the purpose of the code

```
import math

radiuses = [1, 5, 10]

diameter = 2 * radiuses[0]
circumference = math.pi * diameter
print(f"The circumference is {circumference:.4f}.")

diameter = 2 * radiuses[1]
circumference = math.pi * diameter
print(f"The circumference is {circumference:.4f}.")

diameter = 2 * radiuses[2]
circumference = math.pi * diameter
print(f"The circumference is {circumference:.4f}.")
```

The Loop Solution

No redundant code. But

- It is not clear what's the purpose of the code
- It can not be used in other places because it is part of the loop body block.

```
import math

radiuses = [1, 5, 10]

for radius in radiuses:
    diameter = 2 * radiuses
    circumference = math.pi * diameter
    print(f"The circumference is {circumference:.4f}.")
```

The Function Solution

- No redundant code.
- Function name shows its purpose.
- It can be used in other places.

```
import math

# define a function with a parameter radius
# the body is a code block to do the real work
def get_circumference(radius):
    diameter = 2 * radius
    circumference = math.pi * diameter
    return circumference

radiuses = [1, 5, 10]
for radius in radiuses:
    # here the radius is the argument of the function call
    circumference = get_circumference(radius)
    print(f"The circumference is {circumference:.4f}.")
```

Function Concepts

- Functions name: describe the purpose.
- Function call: execute the function by appending a pair of parentheses after the function name.
- Parameters: zero, one or more inputs used by the function body.
- Arguments: the actual input data used in function call.
- Function body: the code block in function definition. It is executed in function call.
- Return value: the output of a function call.

Objects and Types

- In Python, all data are objects.
- Every object has a type that determines the valid operations of the object. You can use built-in function `type(obj)` to get an object's type. For example:
  - `type("hi") # <class 'str'>`
  - `type(3) # <class 'int'>`

Methods and Attributes

- Function defined inside a type is called a method. You call a method using the dot notation. For example:
  - `"hi".title() # Hi`
- An object may contain data that is called an attribute. You use a dot notation to access an attribute. For example, the math module is an object that has an object called pi:

```
import math

print(math.pi) # 3.141592653589793
```

Computational/Algorithmic thinking

- Arithmetic and logic operations
- Variables
- Control flow
  - Sequential execution
  - Branch
  - Loop
- Composite data
- Function
- Type (method and attribute)

- These are essential constructs of any programming language.
- Everything else is nice/bad-to-have optional.
- They are there to provide handy functions.
- Turing complete is all you need for any computational task.

- You will learn more details get a deep understanding of this concepts.
- You will learn other nice-to-have options.