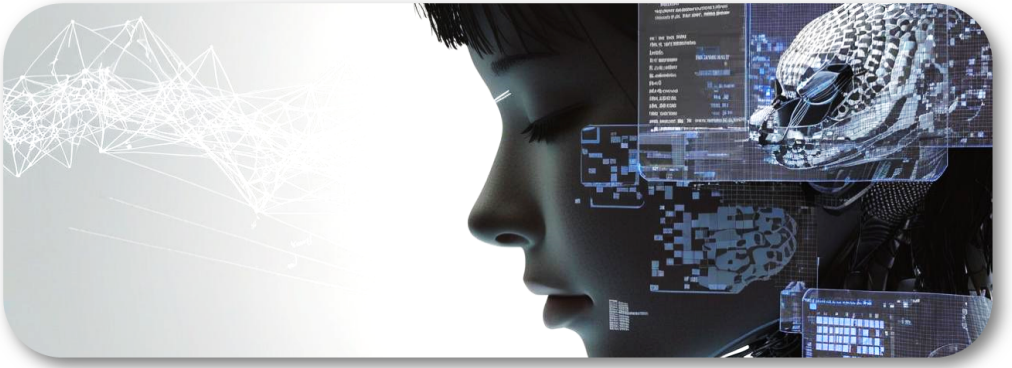

COMP 1023 Introduction to Python Programming

Tutorial 2: Python Fundamentals

COMP 1023 Teaching Team

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology, Hong Kong SAR, China

Warm-up: iPRS Question 1

You took a photo and its size is 1 MB. How many KB does the photo have?

A. 1000 KB

B. 1024 KB

Warm-up: iPRS Question 1 - Answer

You took a photo and its size is 1 MB. How many KB does the photo have?

A. 1000 KB

B. 1024 KB

Explanation

Computers use binary representation.

- 1000 KB: It is equal to 10^3 KB, which is a decimal representation.
- 1024 KB: It is equal to 2^{10} KB, which is a binary representation.

Basic Concepts

- Computer components:
  - Hardware: physical components of a computer
    - CPU (Central Processing Unit)
    - Memory (RAM): consists of an ordered sequence of bytes used from programs
    - Storage: Hard Disk, Solid-State Drive (SSD)
    - Input/Output Devices: Keyboard, Mouse, Monitor
  - Software: programs and data that run on the hardware
- Bits and bytes:
  - Bit: smallest unit of data (0 or 1)
  - Byte: 8 bits, can represent 256 values (0 - 255)
  - Kilobyte (KB), Megabyte (MB), Gigabyte (GB), Terabyte (TB)

Basic Concepts - Cont.

- Algorithms:
  - A step-by-step description/procedure for solving a problem
  - Can be represented in various forms:
    - Flowcharts: graphical representation of the algorithm
    - Pseudocode: informal high-level description of the algorithm
    - Programs: actual code written in a programming language
- Programming languages:
  - High-level languages: Python, Java, C++, etc. which are easier for humans to read and write
  - Assembly languages: low-level languages that are closer to machine code
  - Machine languages: binary code that the computer can directly execute
  - Compiled vs. Interpreted languages
    - Compiled: code is translated to machine code before execution (e.g., C++)
    - Interpreted: code is executed line by line (e.g., Python)

Basic Python Programming Elements

- Program Structure
  - The main function: serves as the entry point of a Python script
    - Not required but is a good practice :)
  - Comments: essential for clarifying code intent
    - Single-line: prefix with #
    - Multi-line: enclose in triple quotes or triple double-quotes

```
"""
This is a multi-line comment.
Hello World!
"""

def main():
    # This is a single-line comment
    print("Welcome to Python programming!")

if __name__ == "__main__":
    main()  # Executes when script runs directly
```

Data Types in Python

Basic Types:

- int: whole numbers (42, -3, 0)
- float: decimal numbers (3.14, 1.23e-5)
- complex: complex numbers (1j, 2+3j, 4-5j)
- bool: logical values (True, False)
- str: character sequences ('Hello', "World")
- None: represents absence of a value

Fun Facts

- We use j to denote imaginary numbers, which differ from the mathematical notation (a+bi), as i is used to represent electrical current in engineering.
- The Boolean (bool) data type is named after George Boole, an English mathematician who developed algebraic logic in the mid-19th century.

Data Types in Python - Cont.

Container Types:

- list: ordered, mutable collections [10, “apple”, 3.14]
- tuple: ordered, immutable sequences (“a”, “b”, 5)
- dict: key-value pairs {“name”: “Alice”, “age”: 25}
- set: unordered, unique elements {“apple”, “banana”}

Variables and Dynamic Typing

- Variable Naming Rules:
  - Start with a letter or underscore (_)
  - Contain only alphanumeric characters and underscores
  - Avoid Python's reserved words (if, for, class)
- Dynamic Typing:
  - Variables can hold values of different types at different points during a program's execution
  - The type is associated with the value, not the variable name.

```
# Valid variable names
student_name = "Charlie"
_private_var = 42
course1023 = "Python Programming"

# Dynamic typing example
x = 10      # x is an integer
x = "ten"   # x now is a string

# Type hinting (optional but recommended)
num_students: int = 30
course_name: str = "Python Basics"
```

iPRS Question 2

Which of the following is an invalid variable name?

A. `salary`

B. `_address`

C. `phone_number`

D. `1023comp`

iPRS Question 2 - Answer

Which of the following is an invalid variable name?

A. `salary`

B. `_address`

C. `phone_number`

D. `1023comp`

Explanation

A variable cannot start with a digit.

Type Conversion

- Built-in conversion functions:
  - int(): converts to integers
  - float(): converts to floating-point numbers
  - str(): converts to strings
  - bool(): evaluates as True or False

```
# Type conversion examples
int(3.9)         # 3 (truncates decimal)
int("42")        # 42
float(5)         # 5.0
float("3.14")    # 3.14
str(100)         # "100"
str(True)        # "True"
bool(0)          # False
bool("abc")      # True
```

Input and Output

- Output with print():
  - sep: specifies separator between values
  - end: defines ending character
  - f-strings: simplify string formatting
- Input with input(): reads user input as a string

```
# Output examples
print(1, 2, 3, sep=":", end="!\n")    # Output: 1:2:3!

name = "David"
age = 28
print(f"{name} is {age} years old.") # Output: David is 28 years old.

# Input example
user_age = input("Enter your age: ")
age = int(user_age)
print(f"Next year, you'll be {age + 1}.")
```

iPRS Question 3

Which of the following can correctly print "`Hello World!`"?

A.
```
print("Hello", "World", "!")
```

B.
```
print("Hello", "!", sep="World")
```

C.
```
hello = "Hello"
world = "World"
print(f"{hello} {world}!")
```

iPRS Question 3 - Answer

Which of the following can correctly print "`Hello World!`"?

A.
```
print("Hello", "World", "!")       # Incorrect, it prints "Hello World !".
```

B.
```
print("Hello", "!", sep="World")   # Incorrect, it prints "HelloWorld!"
```

C.
```
hello = "Hello"
world = "World"
print(f"{hello} {world}!")         # Correct
```

Explanation

- A: The default separator between strings is space, so there is an additional space between "World" and "!".
- B: There is no space between "Hello" and "World".

Random Number Generation

- Core Functions:
  - random.random(): returns float in [0.0, 1.0)
  - random.randint(a, b): returns random integer between a and b inclusive
  - random.choice(): picks random element from sequence
  - random.seed(): initializes for reproducible results
- Random results will always be the same if the same random seed is set.

```
import random

# Generate random numbers
random_float = random.random()         # 0.78456321...
random_int = random.randint(1, 10)     # 7
fruits = ["apple", "banana", "cherry"]
random_fruit = random.choice(fruits)  # "banana"

# Reproducible random numbers
random.seed(42)
print(random.randint(1, 10))  # Always outputs: 8
```

Mutability

- Mutable: list, dict, set (can be modified)
- Immutable: int, float, str, tuple (cannot be changed)

```
# Mutable vs Immutable
my_list = [1, 2, 3]
my_list[0] = 4   # Works: [4, 2, 3]

my_str = "hello"
my_str[0] = 'H'  # Error: strings are immutable

my_int = 10
my_int = 5  # Works: creates a new int, does not modify the original
```

Indexing Operator

- Syntax: `variable[index]`
- You can assess one item in a list and modify it.
- You can also access one character in a string, but cannot modify it.
- Index starts from 0: Index 0 is the first item, index 1 is the second item, and so on.

Good Programming Practices

- Use descriptive names: `student_grades` not `sg`
- Define named constants in uppercase: `MAX_ATTEMPTS = 3`
- Keep consistent 4-space indentation
- Write One statement per line
- Include meaningful comments

iPRS Question 4

Which of the following is NOT a good programming practice?

A. Define a variable to be: `student_name = "Chris Wong"`
B. Define a constant to be: `PI_VALUE = 3.141592654`
C. Write statements to be: `age = 18; gender = "M"`
D. Write a comment to each function to describe what it does

iPRS Question 4 - Answer

Which of the following is NOT a good programming practice?

A. Define a variable to be: `student_name = "Chris Wong"`

B. Define a constant to be: `PI_VALUE = 3.141592654`

C. Write statements to be: `age = 18; gender = "M"`

D. Write a comment to each function to describe what it does

Explanation

Writing multiple statements on a single line is not a good practice.
Fun fact: You can literally do so by separating statements using a semicolon, but not recommended. :)

Reminders on Lab 2 Assignment

- You need to submit this lab assignment!
- You will learn how to read user's input and print the content in Python.
- You will also learn how to do basic math calculations in Python.

```
# Sample program to do math calculations
number_1 = int(input("Enter a number: "))                    # 3
number_2 = int(input("Enter another number: "))              # 4
addition_result = number_1 + number_2
multiplication_result = number_1 * number_2
print(f"{number_1} + {number_2} = {addition_result}")        # 3 + 4 = 7
print(f"{number_1} * {number_2} = {multiplication_result}")  # 3 * 4 = 12
```

Note on Multiplication

In math, we can write $(p \times q)$ as (pq). However, in programming, we must use the asterisk (*) to show multiplication, so we write it as (p * q).

Think More about the Print Statement

After you write the print statement in your lab, you can think: Are there other ways to print the same thing? Let's see this example:

```
course_code = input("Enter a course code you are taking: ") # COMP 1023
print("I hope you are doing great in", course_code, "this semester!")
print("I hope you are doing great in ", course_code, " this semester!", sep="")
print("I hope you are doing great in " + course_code + " this semester!")
print(f"I hope you are doing great in {course_code} this semester!")
# I hope you are doing great in COMP 1023 this semester!
```

- The above print statements can print the exact same things!
- In the lab assignment, there are no standard answers.
- You can get full credits as long as you can correctly print the desired content.

Common Mistake

- ZINC will check your program's output against the expected output.
- Please ensure the output format is **EXACTLY** the same as the provided example.
  - Refer to the lab page on the course website.
- Common mistake: Missing or having additional spaces.

```
print("I hope you are doing great in ", course_code, " this semester!")
# I hope you are doing great in  COMP 1023  this semester!
print(f"I hope you are doing great in{course_code}this semester!")
# I hope you are doing great inCOMP 1023this semester!
```

Tutorial Questions (Self Study) - Part 1

- Question 1: Which of the following is NOT a hardware component of a computer system?

  A. CPU
  B. Python interpreter
  C. Keyboard
  D. Hard drive

- Question 2: What will be the output of the following Python code?

```
print(1, 2, 3, sep="-", end="!")
```

  A. `1 2 3`
  B. `1-2-3!`
  C. `1-2-3`
  D. `123!`

- Question 3: What is a step-by-step description/procedure for solving a problem called in computer science?

Tutorial Questions (Self Study) - Part 2

- Question 4: In Python, which symbol is used to create single-line comments?
- Question 5: What data type does the input() function always return in Python?
- Question 6: Explain the main differences between a compiler and an interpreter. Provide examples of programming languages for each type and discuss their advantages and disadvantages.
- Question 7: Write a Python program that asks the user for their current age and calculates how many years they have left until they turn 100 years old. Display the result using an f-string.

Answers to Tutorial Questions - Part 1

- Answer 1: B. Python interpreter
  - Python interpreter is software, not hardware
  - Hardware includes physical components like CPU, keyboard, hard drive
- Answer 2: B. 1-2-3!
  - `sep="-"` puts dashes between values
  - `end="!"` replaces the default newline with exclamation mark
- Answer 3: algorithm
  - An algorithm is a systematic description/procedure for solving problems
- Answer 4: # (hash symbols)
  - Single-line comments start with #
- Answer 5: string (str)
  - The input() function always returns a string, even for numbers
  - e.g. If the user inputs 1, it is read as string "1", not the integer 1.

Answers to Tutorial Questions - Part 2

- Answer 6: Difference between compiler and interpreter:
  - Compiler: Translates entire source code to machine code before execution
    - Examples: C++, C, Rust
    - Faster execution, but slower compilation
    - Creates executable files
  - Interpreter: Executes code line by line at runtime
    - Examples: Python, JavaScript
    - Slower execution, but faster development cycle
    - No separate compilation step needed
- Answer 7: Python program for age calculation:

```python
def main():
    age = int(input("Enter your age: "))
    years_left = 100 - age
    print(f"You have {years_left} years left until you turn 100.")

if __name__ == "__main__":
    main()
```

That's all!

Any questions?