**IDX G9 CS H STUDY GUIDE ISSUE 1**

**By Joyce**

Lesson 1: Computer Category and Programming Language

Computers are classified into two:

1. A **special-purpose** computer
2. A **general-purpose** computer

Special-Purpose Computers

* Definition: Computer designed for a particular function.
* Executes the **same** stored set of instructions when requested
* Examples:

1. Microwave ovens
2. Washing machine
3. Medical diagnostic equipment

General-Purpose Computers

* Used for solving **many** different types of problems
* Available in various sizes
* Has a wide range of capabilities
* Classified into:

1. **Microcomputers**

* Sometimes referred to as personal computer (PC)

1. **Laptop computers (Notebook computers)**

* Smallest microcomputers

1. **Desktop computers**

* Compact microcomputer systems that fit on a desk
* Designed for individual use

1. **Workstations**

* Largest type of microcomputer
* Used in scientific and engineering applications

1. **Mainframe computers**

* Largest computer
* Massive memory + extremely rapid processing power
* Used to handle massive amounts of data or complex processes
* Areas of use: large-scale business, military, scientific fields

1. **Supercomputers**

* Used for tasks requiring rapid and complex calculations involving numerous variables
* Areas of use: scientific research, weather prediction, aircraft design, nuclear weapon, etc.

Machine Language

* Natural language of computer
* Only language that can be **directly** used by computer
* Does **not** need translation
* Immediate execution
* Instruction is a binary string of 1s and 0s
* Advantage:

1. Runs **fast** and **efficiently**

* Disadvantage:

1. Programs difficult to write and debug
2. Each computer has its own machine language
3. **Not** portable because programs written in one type of computer cannot be run on another type

Assembly Language

* Consists of English-like abbreviations
* Programs **cannot** be directly processed by computer
* Because of variations in assembly languages, programs written using them are **not** portable
* Useful for……

1. implementing system hardware
2. small embedded system applications

* Advantages:

1. Programmer in control of hardware
2. Space and time **efficiency**

* Assembly Language 🡪Assembler+Linker🡪Machine Language
* **Assembler**= Language translators that convert assembly language to machine code

High-Level Languages

* Ex: Basic, Java, FORTRAN
* **Machine independent**
* Programmers don’t need to know anything about the internal structure of the computer
* Deal with high-level coding
* Enable programmers to write instructions using….

1. English words
2. Familiar mathematical symbols and expressions

* High-Level Language 🡪 Compiler or Interpreter 🡪 Machine Code 🡪 Hardware

Lesson 2: Programming Intro

Hardware & Software

* User🡪Software🡪Hardware
* Hardware:

1. Only understand **binary** system (machine language)
2. Hard for human understanding and use

* Software:

1. **Interface** between humans and hardware
2. Step-by-step instructions for hardware operations
3. Translate human intention🡪 form computers can understand  
   Feedback from hardware🡪form humans can understand

Programming

* **Computer programming**: the process of writing, testing, debugging, and maintaining the source code of computer programs
* Use software development to do programming
* Through programming, we create software
* Use programming to solve hard/large-scale problems

Computational Thinking

* Involves the ability to:

1. Think procedurally
2. Think logically
3. Think ahead

Methods for running the code

* Two ways to run the code

1. **Interactive mode** (start with >>>)
2. **Script mode** (using .py file)

* Script mode VS Interactive mode

1. Script mode: File must be created and saved before executing the code to get results  
   Interactive mode: Result returned immediately after pressing enter
2. Script mode: Direct way of editing your code  
   Interactive mode: No direct method to edit code
3. Script mode: Does not have the >>> prompt

Interactive mode: Has the >>> prompt

Operators

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Operation | Example | Evaluates to |
| + | Addition | 5+2 | 7 |
| - | Subtraction | 5-2 | 3 |
| \* | Multiplication | 5\*2 | 10 |
| / | Division | 4/2 | 2 |
| \*\* | Power | 3\*\*2 | 9 |

Indention

* Space at beginning of a code line
* Indention in:

1. Python: a **block** of code
2. Other programming languages: readability (no specific meaning)

print() function

* Used to print values to the screen.
* When Python calls the print() function, a value is passed to the function
* **Argument**-value passed to a function call
* Keyword **end**

1. defines something to be printed automatically at the end of the output
2. defaults to the newline ("\n")

Lesson 3: Variables and Data Types

String Concatenation

* operator **+** is used to **concatenate two strings** as the operation.
* “hello”+ “python” 🡪 “hello python”

input() function

* Read string from standard input
* Return **string** type

Variable

* **Containers** for storing data values
* Store numerical or data value into variable temporarily and get the value from the variable when you need later
* Variable created when you assign value to it
* Advantages:

1. Make code more readable
2. Avoid typing a value repeatedly
3. Make codes easier to modify and debug
4. Let programs run faster

Naming rules

* Variable name contains **only letters, numbers, and underscores (\_)**
* Do not start with a number Ex: 8piece
* No spaces
* Case sensitive Ex: AGE, age, Age are 3 different variables
* Do not use reserved keywords   
  ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
* Names should be short and descriptive
* **Camel Case Naming**

1. Makes compound names more readable
2. Ex: myOneMethod is easier to read than myonemethod

Assigning Variables

* Equal (=) operator is used to assign value to a variable
* Multiple assignment: Assign a value to multiple variables in a single statement

1) Assigning single value to multiple variables:

>>> x=y=z=50

2) Assigning multiple values to multiple variables:

>>> a,b,c=5,10,15

Data types

* Integer (integers like 0, -2, 29, etc.)
* Float (floating point numbers like 1.0, 9.902, 1e10, 44e-2, etc.)
* String (textual value like “the”, “1”, “num of line”, etc.)

Different types of functions

|  |  |
| --- | --- |
| Function | Purpose |
| int() | convert a number or string to an integer  return 0 if no arguments are given |
| float() | convert a string or number to a floating point number |
| str() | convert a number to a string |

Lesson 4: Expressions

Expressions:

* consist of values (such as 4) and operators (such as \*)
* always evaluate down to a single value

Importing Modules

* All python programs…

1. can call a basic set of functions called **built-in functions,** including the print(), input()
2. also comes with a set of modules called the **standard library**. Each module contains a related group of functions that can be embedded in your programs.

* Import module with import statement

>>>import. Math

* After you import a module, you can use all functions of that module

math Module

|  |  |
| --- | --- |
| math.pi |  |
| math.ceil(x) | return an integer >= x |
| math.floor(x) | return an integer <= x |
| math.sqrt(x) | return the square root of x |

Lesson 5: Lists

List

* A collection of items in a particular order
* square brackets ([]) indicate a list
* individual elements separated by commas

Index

* You can access any element in a list by telling Python the position, or **index**, of the desired item
* Access element by writing the name of the list followed by the index of the item enclosed in square brackets ([])
* First item’s index=**0**

len() Function

* Return the number of elements in a list
* EX:  
  >>> numList= [0, 1 , 2 , 3 , 4 , 5]

>>>len(numList)  
6

Adding elements to a list

* **append()** method adds the new element to the end of the list.
* Ex:  
  >>> numList = [0, 1, 2, 3, 4, 5]

>>> numList.append(12)

>>> numList

[0, 1, 2, 3, 4, 5, 12]

* add a new element at any position in your list by using the **insert()** method.
* Specify the index of the new element and the value of the new element.
* Ex:  
  >>> numList = [0, 1, 2, 3, 4, 5]

>>> numList.insert(2, 12)

>>> numList

[0, 1, 12, 2, 3, 4, 5]

Removing Elements from a List

* Use **del** statement when you know the position of the item you want to remove from a list
* Ex:

>>> numList = [0, 1, 2, 3, 4, 5]

>>> del numList[2]

>>> numList

[0, 1, 3, 4, 5]

* Use **remove ()** method is you only know the value of the item you want to remove
* Deletes only the first occurrence of the value you specify
* Ex:  
  >>> numList = [1, 5, 4, 2, 3, 0, 3]

>>> numList.remove(3)

>>> numList

[1, 5, 4, 2, 0, 3]

Find the Index of the Element

* **lst.index(value)**-return the first occurrence of the value

Lesson 6: For-loop

* Used to iterate a block of statements several times
* Syntax of for loop:  
  **for** variable **in** someList:  
   statements
* Convert each element to int/float

Lesson 7: For-Range

* Used to generate a sequence of numbers
* Syntax:  
  **for** variable **in range(start,stop,step)** statements

Random Numbers

* Module random contains functions that generate random numbers in a variety of ways
* Function **randint**(a,b) returns integer in range of [a,b]

import random

for i in range(10):

print(random.randint(1, 6))