Computer Science Intensive Course - MindX

BÀI 1. ÔN TẬP PYTHON

1. Các thành phần cơ bản của Python

1.1. Hello World!

In [ ]:

```
# this line prints "Hello World" to the screen
print("Hello World")
```

```
Hello World
```

In [ ]:

```
# in a Python file, e.g. hello_world.py
if __name__ == '__main__':
    print("Hello World")
```

```
Hello World
```

1.2. Biến và Hằng

Trong Python, biến và hằng được định nghĩa giống nhau

In [ ]:

```
# variable
number = 2
print(number)
```

```
2
```

In [ ]:

```
# constant
PI = 3.1416
PI  # in notebook, we can get the value of a variable just by calling its name
```

Out[ ]:

```
3.1416
```

1.3. Kiểu Dữ Liệu

Python có các kiểu dữ liệu cơ bản như những ngôn ngữ lập trình khác.
Một số kiểu dữ liệu phổ biến:

In [ ]:

```
var_int = 3  # integer
var_float = 123.456  # floating point number / real number
```

```
var_boolean = True  # logic type, can be either True or False

var_string = "hello world"  # string
var_string2 = 'a'  # a character is considered a string
```

Xem kiểu dữ liệu của một biến:

In [ ]:

```
type(var_int)
```

Out[ ]:

```
int
```

Trong Python, ta không cần xác định kiểu dữ liệu khi khởi tạo biến, nhưng mỗi biến vẫn có kiểu dữ liệu riêng. Một trong những ứng dụng của đặc điểm này là ta có thể thay đổi kiểu dữ liệu của một biến đã được tạo.

In [ ]:

```
var_string = False
type(var_string)
```

Out[ ]:

```
bool
```

Tên biến có thể được xem như là một "nhãn dán" trên dữ liệu thực tế được lưu trong bộ nhớ. Trong một số trường hợp, việc thay đổi giá trị của một biến có thể được xem như việc chuyển nhãn dán từ vị trí này sang vị trí khác trong bộ nhớ.

1.4. Phép Toán

Phép Toán Toán Học

In [ ]:

```
num1 = 13
num2 = 6.3
num3 = 6

print("Add:",        (num1 + num2))
print("Substract:", (num1 - num2))
print("Multiply:",  (num1 * num2))
print("Divide:",    (num1 / num2))
print("Divide for whole number:", (num1 // num2))  # python 3
print("Modulo:",    (num1 % num3))
print("Power:",     (num1 ** num2))
```

```
Add: 19.3
Substract: 6.7
Multiply: 81.89999999999999
Divide: 2.0634920634920637
Divide for whole number: 2.0
Modulo: 1
Power: 10419409.804138675
```

Kết hợp phép gán và phép toán:

In [ ]:

```
num4 = 3
print(num4)
num4 += 10
print(num4)
```

```
3
```

```
13
```

Kết hợp các phép toán với nhau:

In [ ]:

```
print(num1 + num2 * num3)
print(((num1 + num2) * num3) / 14)
```

```
50.8
8.271428571428572
```

Sử dụng hàm có sẵn:

In [ ]:

```
print("Max:", max(num1, num2))
print("Min:", min(num1, num2))
print("Absolute value:", abs(-123))
```

```
Max: 13
Min: 6.3
Absolute value: 123
```

Phép Toán So Sánh

In [ ]:

```
print("Bigger:", num1 > num2)
print("Smaller:", num1 < num2)
print("Equal:", num1 == num2)
print("Not equal:", num1 != num2)
print("Bigger or equal:", num1 >= num2)
print("Smaller or equal:", num1 <= num2)
```

```
Bigger: True
Smaller: False
Equal: False
Not equal: True
Bigger or equal: True
Smaller or equal: False
```

Phép Toán Logic

In [ ]:

```
bool1 = 3 < 5  # true
bool2 = 3 > 5  # false

print("AND:", bool1 and bool2)
print("OR:", bool1 or bool2)
print("NOT:", not bool1)
```

```
AND: False
OR: True
NOT: False
```

1.4. Input và Output

Nhập và xuất string:

In [ ]:

```
var_input = input()    # get user input by the "input" function, this function returns a s
tring variable
```

```
print(type(var_input))  # output is just a print function
var_input
```

```
test
<class 'str'>
```

Out[ ]:

```
'test'
```

Nhập và xuất số nguyên:

In [ ]:

```
var_input = int(input())  # convert the input to int
print(type(var_input))
var_input
```

```
123
<class 'int'>
```

Out[ ]:

```
123
```

1.5. Hàm

Hàm là một nhóm các câu lệnh dùng để thực hiện một nhiệm vụ cụ thể nào đó.

Mục đích:

- **Chia nhỏ một chương trình lớn thành nhiều phần**
- **Dễ hiểu, dễ quản lý, chỉnh sửa code**
- **Dễ dàng làm việc với người khác**
- **Sử dụng lại những đoạn code giống nhau**

In [ ]:

```
# define function
def say_hi(name, title="Mr."):
    print("Hi " + title + " " + name + "!")


# call the function
say_hi("Henry")
say_hi("Jane", title="Ms.")
```

```
Hi Mr. Henry!
Hi Ms. Jane!
```

In [ ]:

```
def add(num1, num2):
    return num1 + num2

add(1, 2)
```

Out[ ]:

```
3
```

1.6. Module

Code Python có thể được đóng gói thành module và được sử dụng bởi các file Python khác.
Các module có sẵn còn được gọi là thư viện.

In [ ]:

```
import math  # import math library


# use the math library
print("sin:", math.sin(math.pi/2))
print("cos:", math.cos(math.pi))
print("factorial:", math.factorial(5))
```

```
sin: 1.0
cos: -1.0
factorial: 120
```

2. Cấu Trúc Điều Khiển

2.1. Cấu Trúc Điều Kiện

Cấu trúc If hoàn chỉnh:

In [ ]:

```
num = 2
if num == 2:
    print("This number is two")
else:
    print("This number is not two")  # the indentation must match that of the if block
```

```
This number is two
```

Cấu trúc If không có Else:

In [ ]:

```
num = 3
if num == 2:
    print("This number is two")
    # put other statements here
```

Cấu trúc If lồng nhau:

In [ ]:

```
num = 1
if num < 4:
    if num == 3:
        print("This number is three")
    else:
        print("This is a weird number")
```

```
This is a weird number
```

In [ ]:

```
num = 1
if num == 2:
    print("This number is two")
elif num == 3:
    print("This number is three")
else:
    print("This is a weird number")
```

```
This is a weird number
```

2.2. Cấu Trúc Lặp

Vòng Lặp For

Sử dụng khi số lần lặp được xác định trước / lặp trong một tập hợp hữu hạn số phần tử.
Trong Python, vòng lặp For được sử dụng trên list, string, set, dictionary và các loại tập hợp.

In [ ]:

```
for i in range(2, 10):  # try with range(2, 10), range(10, 2, -1)
    print(i, end=" ")
```

```
2 3 4 5 6 7 8 9
```

In [ ]:

```
for c in "loop":
    print(c, end=" ")
```

```
l o o p
```

In [ ]:

```
grades = {"Henry": 2, "Jane": 8, "Andy": 9}
for name in grades:
    print(grades[name], end=' ')
```

```
2 8 9
```

Lặp với index và value:

In [ ]:

```
name_arr = ["Henry", "Jane", "Andy"]
for i, name in enumerate(name_arr):
    print("Person {}: {}".format(i, name))
```

```
Person 0: Henry
Person 1: Jane
Person 2: Andy
```

Sử dụng lệnh break để dừng vòng lặp:

In [ ]:

```
for i in range(10):
    print(i, end=' ')
    if i > 5: break
```

```
0 1 2 3 4 5 6
```

Sử dụng lệnh continue để bỏ qua một lần lặp:

In [ ]:

```
for i in range(10):
    if i == 5: continue
    print(i, end=' ')
```

```
0 1 2 3 4 6 7 8 9
```

Vòng Lặp While

Sử dụng khi không xác định trước số lần lặp.
Chú ý: Cần tránh các trường hợp lặp vô hạn bằng cách đảm bảo điều kiện dừng.

In [ ]:

```
i = 0
```

```
while i < 10:
    print(i, end=' ')
    i += 1  # the variables connecting with the stop condition should be changed
```

```
0 1 2 3 4 5 6 7 8 9
```

Sử dụng lệnh break để dừng vòng lặp:

In [ ]:

```
i = 0
while i < 10:
    print(i, end=' ')
    i += 1
    if i > 5: break
```

```
0 1 2 3 4 5
```

Sử dụng lệnh continue để bỏ qua một lần lặp:

In [ ]:

```
i = 0
while i < 10:
    if i == 5:
        i += 1
        continue
    print(i, end=' ')
    i += 1
```

```
0 1 2 3 4 6 7 8 9
```

2.3. Xử Lý Exception

Cấu trúc Try... Except...:

In [ ]:

```
try:
    print(something)
except:
    print("Exception occured")
```

```
Exception occured
```

In [ ]:

```
try:
    print(something)
except NameError:  # catch a specific exception
    print("NameError exception")
except:
    print("Other exception")
```

```
NameError exception
```

Try... Except... Finally...:

In [ ]:

```
f = None
try:
    f = open("demofile.txt", "r")
    print(f.readline())  # print a line from file
except:
    print("File not found")
finally:
```

```
    if f != None:
        f.close()
```

```
File not found
```

3. Lập Trình Hướng Đối Tượng (OOP)

Python là một ngôn ngữ lập trình hướng đối tượng (OOP - Object Oriented Paradigm)
Lập trình hướng đối tượng cho phép lập trình viên tạo ra các đối tượng trừu tượng nhằm làm cho code đơn giản, dễ đọc, dễ bảo trì.

3.1. Lớp và Đối Tượng (Class & Object)

In [ ]:

```
# class definition
class Person:

  race = "human"  # class attribute

  def __init__(self, name, age):  # constructor
    self.name = name              # instance attribute
    self.age = age

  def say_hi(self):               # method
    print("Hello my name is " + self.name)

  def tell_the_day(self, day):
    print("Today is " + day)
```

In [ ]:

```
# initialize object
john = Person("John", 36)

# call object methods
john.say_hi()
john.tell_the_day("Monday")
```

```
Hello my name is John
Today is Monday
```

3.2. Thuộc Tính (Attribute)

Thuộc Tính Của Lớp và Thuộc Tính Của Đối Tượng

In [ ]:

```
# class attribute
Person.race
```

Out[ ]:

```
'human'
```

In [ ]:

```
# initialize object
john = Person("John", 36)

# get object attributes
print("Age: " + str(john.age))

# set object attributes
john.age = 10
```

```
print("Age: " + str(john.age))
```

```
Age: 36
Age: 10
```

Thuộc Tính Public và Private

- **Public attribute: thuộc tính có thể truy vấn tự do**
- **Private attribute: thuộc tính chỉ được truy vấn bên trong class**

In [ ]:

```
class Person:

  race = "human"

  def __init__(self, name, age):
    self.__name = name  # private attribute
    self.age = age      # public attribute

  def say_hi(self):
    print("Hello my name is " + self.__name)


john = Person("John", 20)
print(john.age)

john.say_hi()
```

```
20
Hello my name is John
```

In [ ]:

```
print(john.name)  # cannot access private attributes
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-37-83b40878d933> in <module>
----> 1 print(john.name)  # cannot access private attributes

AttributeError: 'Person' object has no attribute 'name'
```

3.3. Phương Thức (Method)

- **Constructor: Là phương thức đặc biệt, được gọi khi một đối tượng được khởi tạo**
- **Public method: Phương thức có thể được gọi tự do**
- **Private method: Phương thức chỉ được gọi bên trong class**

In [ ]:

```
class Person:

  race = "human"

  def __init__(self, name, age):           # constructor
    self.__name = name
    self.__age = age

  def __get_yob(self, current_year=2021):  # private method
    return current_year - self.__age

  def say_hi(self):                        # public method
    print("Hello my name is {}. I was born in {}".format(self.__name, self.__get_yob()))


john = Person("John", 20)
```

```
john.say_hi()
```

```
Hello my name is John. I was born in 2001
```

In [ ]:

```
john.__get_yob()  # cannot call private method
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-35-9a864d42c44f> in <module>
----> 1 john.__get_yob()  # cannot call private method

AttributeError: 'Person' object has no attribute '__get_yob'
```

3.4. Tính Kế Thừa (Inheritance)

Là khả năng tạo class mới sử dụng các thành phần của class đã có và tiếp tục triển khai chi tiết hơn.
Tính kế thừa được dùng để tận dụng các phần code đã có.

In [ ]:

```
# parent class
class Person:

  def __init__(self, name, age):
    self._name = name  # protected attribute
    self._age = age    # protected attribute

  def say_hi(self):
    print("Hello my name is " + self._name + ". I am " + str(self._age) + " years old.")


# child class
class Engineer(Person):  # inherit "Person" class

  def __init__(self, name, age, field):  # this overrides the parent's constructor
    Person.__init__(self, name, age)     # re-use the parent's constructor
    self.__field = field

  def say_field(self):
    print("I work in the " + self.__field + " industry.")



john = Engineer("John", 36, "Software")  # call child class's constructor
john.say_hi()     # parent class's method
john.say_field()  # child class's method
```

```
Hello my name is John. I am 36 years old.
I work in the Software industry.
```

4. Các Cấu Trúc Dữ Liệu Sẵn Có

4.1. Cấu Trúc Dữ Liệu Có Thứ Tự

List (Array)

In [ ]:

```
arr = ["one", "two", "three"]
arr[1]
```

Out[ ]:

```
'two'
```

In [ ]:

```
# list comprehension
arr = [x**2 for x in range(10)]
arr
```

Out[ ]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Tuple

Tuple có cấu trúc giống List, tuy nhiên các giá trị bên trong Tuple không thể thay đổi

In [ ]:

```
tup = ("one", "two", "three")
print(tup[1])
```

```
two
```

String

Trong Python, String chứa các kí tự Unicode

In [ ]:

```
string = "test_string"
string[:3]  # slicing
```

Out[ ]:

```
'tes'
```

4.2. Cấu Trúc Dữ Liệu Không Thứ Tự

Set

In [2]:

```
vowels = {"a", "e", "i", "o", "u"}
squares = {x * x for x in range(10)}

print("a" in vowels)
vowels
```

```
True
```

Out[2]:

```
{'a', 'e', 'i', 'o', 'u'}
```

Dictionary

In [ ]:

```
phonebook = {"bob": 7387, "alice": 3719, "jack": 7052}

print(phonebook["alice"])
print("alice" in phonebook)
```

```
3719
True
```

5. Thực Hành

Bài 1. Ví Dụ

Input: Một số thực *a*.
Output: Giá trị gấp đôi của *a*.
Ví dụ:

- **Input: a = 5**
- **Output: result = 10**

In [ ]:

```
# get input
print("Input the number a: ", end='')
num = float(input())

# process input
num *= 2

# print output
print("Result:", num)
```

```
Input the number a: 2
Result: 4.0
```

Bài 2. Hình Tròn

Input: Một số thực *r* là bán kính hình tròn.
Output: Chu vi và diện tích của hình tròn.
Biết:

$$\pi = 3.1416$$
$$P = 2\pi r$$
$$S = \pi r^2$$

Ví dụ:

- **Input: r = 3**
- **Output:**
  - **Perimeter: 18.8496**
  - **Area: 28.2744**

In [ ]:

```
# define the PI constant
PI = 3.1416

# get input
print("Input the circle's radius: ", end='')
r = float(input())

# process input
perimeter = 2*PI*r
area = PI * (r**2)

# print output
print("Perimeter:", perimeter)
print("Area:", area)
```

```
Input the circle's radius: 3
Perimeter: 18.8496
Area: 28.2744
```

Bài 3. Tổng Các Số Chẵn

Input: Một số nguyên *n > 0* và *n* số nguyên dương.
Output: Tổng các số nguyên dương *chẵn* được nhập.
Ví dụ:

- **Input: n = 4, arr = [5, 4, 1, 12]**
- **Output: result = 16**
- **Giải thích: Trong 4 số nguyên được nhập, có 4 và 12 là số chẵn. Do đó, tổng tìm được là 4+12=16**

In [ ]:

```
# get n
print("Input the number n: ", end='')
n = int(input())

result = 0  # init sum
for i in range(n):

    # get next number
    print("Input number {}: ".format(i+1), end='')
    num = int(input())

    # check even
    if num % 2 == 0:
        result += num

# print output
print("Result:", result)

# ANOTHER METHOD: Save input to list, then process
```

```
Input the number n: 4
Input number 1: 1
Input number 2: 2
Input number 3: 3
Input number 4: 4
Result: 6
```

Bài 4. Hình Chữ Nhật

Input: Hai số thực *a, b > 0* là độ dài hai cạnh kề nhau của một hình chữ nhật.
Output: Chu vi và diện tích của hình chữ nhật.
Yêu cầu: Tạo class *Rectangle* có các phương thức *calculate_perimeter()* và *calculate_area()*
Ví dụ:

- **Input: a = 4, b = 12**
- **Output:**
  - **Perimeter: 32**
  - **Area: 48**

In [ ]:

```
# class definition
class Rectangle:

    def __init__(self, edge1, edge2):
        self.__edge1 = edge1
        self.__edge2 = edge2

    def calculate_perimeter(self):
        return (self.__edge1 + self.__edge2) * 2

    def calculate_area(self):
        return self.__edge1 * self.__edge2


# get input
```

```
edge1 = float(input())
edge2 = float(input())

# init object
rectangle = Rectangle(edge1, edge2)

# print result
print("Rectangle perimeter: ", rectangle.calculate_perimeter())
print("Rectangle area: ", rectangle.calculate_area())
```

```
1
2
Rectangle perimeter:  6.0
Rectangle area:  2.0
```

In [ ]:

```
```