STMC coding team Training

Lesson 2: Variables, data types

Tsai Yun Chen

February 29, 2024

Goal today

- Formally introduce the concept of variable
- Introduce concept of data type: `int`, `char`, `str` and `boolean`
- Perform basic arithmetics and complete some simple exercise

A small recap

So What we have learnt so far?

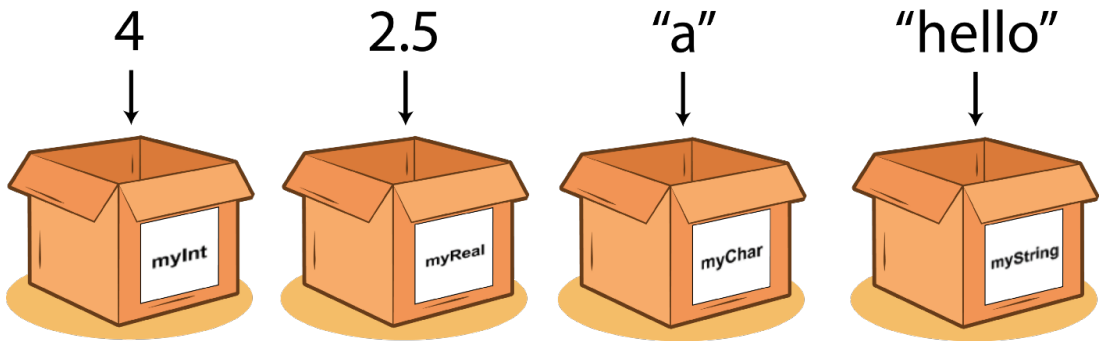
- Asking a computer to output something – `print`
- Entering something to computer – `input`
- Storing something in computer – `xxx=`

Input, process and output

- Almost all program consist of 3 parts: Recieving inputs, processing inputs, and returning outputs
- For example, in a computer game, the game continuously recieve inputs like mouse clicks, keyboard typing, process them to perform actions on the character, and finally displaying the updated screen to you
- Today we will like to illustrate concepts surrounding these things using an example

Variables - Your handy storage box

- To store data, a computer uses something called a **variable**
- A variable is a ”storage box” with a name
- It stores data temporarily so that the value inside can be retrieved for further processing



Source: https://stevenpcurtis.medium.com/what-is-a-variable-3447ac1331b9

Three parts of a variable

- A variable always consist of 3 things:
  1. **Name** for us to refer later in the program
  2. **Value** that contain what is stored
  3. **Data type** that indicate what *kinds* of value is stored
- Name and value is trivial, let's look at data types

What is data types?

- Values in computer are always stored as 0s and 1s
- The same sequence of bits can represent different things according to the way we decode it!
- Different data type care stored and manipulate differently by computer!
- So important to let the computer knows what **data types** they are dealing with

Data types: Integer

- **Integers** ($\mathbb{Z}$) are *numbers without decimal points and fraction component*
- Examples of integers $\{\cdots, -4, -3, -2, -1, 0, +1, +2, +3, +4, \cdots\}$
- Examples of integral data: *age*, *no. of apples*, *no. of people*

Data types: Float

- **Floating point numbers** are *numbers with decimal points*
- Examples of floating point numbers: 2.5, 0.0, -3.1, 100.173
- Examples of float data: *temperature*, *volume*, *mass*, *height*

Data types: Character

- **Character** is *a single alphabet/symbol*
- Usually quoted by **single quotes (’)**
- Examples of characters numbers: `'a','F','@','0','+'`
- Examples of character data: *letter grades*,*biological gender*,*T/F*

Data types: String

- **String** is *a sequence of one or more character*
- Usually quoted by **double quotes (“”)**
- Examples of string: `"hello","ymchan@gmail.com","1+2=3","kim_979"`
- Examples of string data: *name*, *email*, *username*, *password*

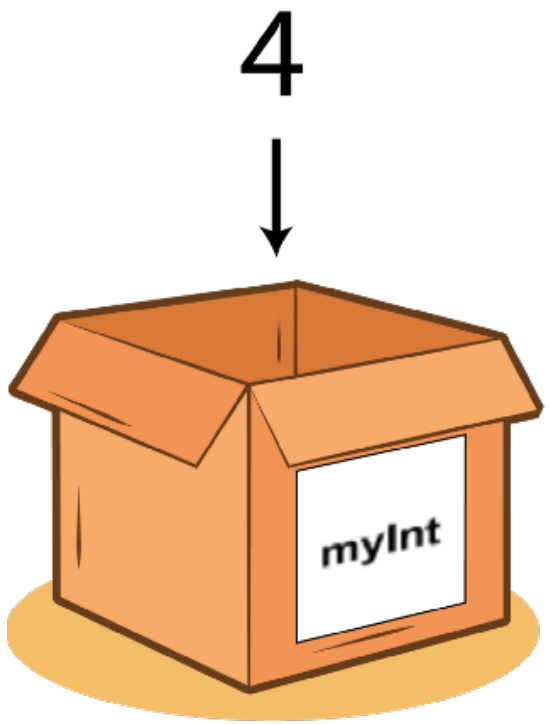
Data types: Boolean

- **Boolean** is a variable either `true` or `false`
- Usually denote a state, or some flag
- Examples of boolean data: *is_married*, *is_empty*, *is_opened*, *is_running*

Case study I

- Variable name is **myInt**
- The value stored is **4**
- The data are numbers *without decimal points*, so the data type is **integer**
- Python Code:

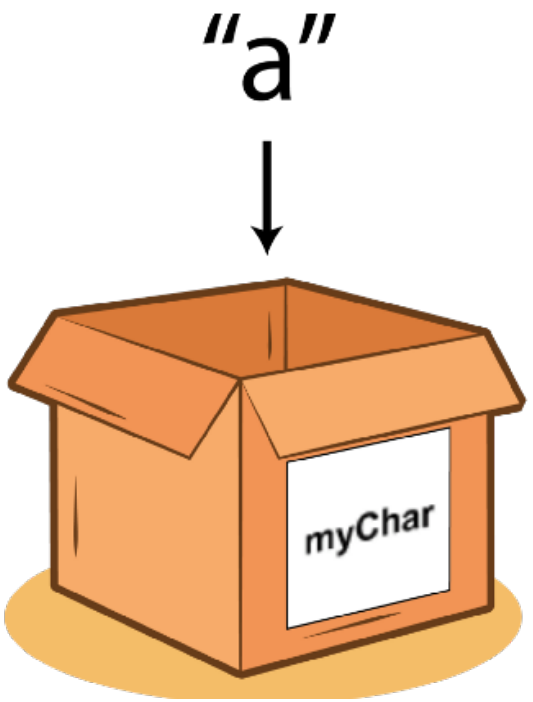
```
1    myInt=4
```



Case study II

- Variable name is **myChar**
- The value stored is **”a”**
- The data is a *single character*; The data type is **character**
- Python Code:

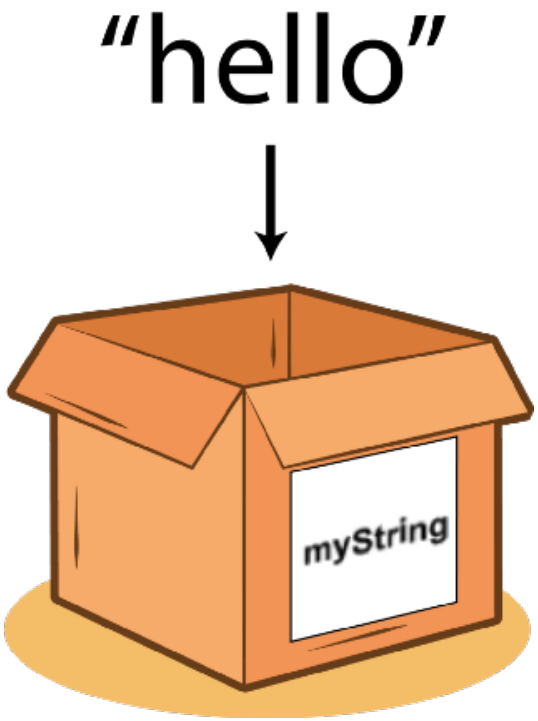
```
1    myChar='a'
```



Case study III

- Variable name is **myString**
- The value stored is **”hello”**
- The data is a *sequence of characters*; The data type is **string**
- Python Code:

```
1    myString="hello"
```



Checking data type

- We can also look at the data type of things using `type()` function

```
>>> type(1)
<class 'int'>

>>> type(1.2)
<class 'float'>

>>> type('12')
<class 'str'>

>>> type(True)
<class 'bool'>
```

Checking data type

- Similarly for variables

```
>>> myName = "John"
>>> type(myName)
<class 'str'>

>>> temp = 30.2
>>> type(temp)
<class 'float'>
```

Checking data type

- We can also use the `is` keyword to check if an object belongs to a type

```
>>> age = 20
>>> type(age)
<class 'int'>
>>> type(age) is int
True

>>> myName  = "rs132"
>>> type(myName) is str
True
```

Operator

- We can also manipulate the value stored in the variable using various operators. For example, if the varaible is `int` or `float`:

```
>>> 1.0 + 2.0       # Addition
3.0
>>> 3 - 10          # Subtraction
-7
>>> 3 * 5           # Multiplication
15
>>> 4/3             # Division
1.3333333333333333
>>> 4**5            # Exponent (4**5 = 4*4*4*4*4)
1024
>>> 17//3           # Integer division, only applicable to Int
5
>>> 17%3            # Modulo/Remainder, only applicable to Int
2
```

Doing operations with variables

- Similarly for string, we have the + and * operations defined
- Note that some operations will results in error

```
>>> "hello" + "bye"          # Concatenate two string
'hellobye'
>>> "hello"*3                # Concatenate string 3 times
'hellohellohello'

>>> "hello" * "bye"          # Make no sense
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cant multiply sequence by non-int of type 'str'

>>> "hello" - "bye"          # Make no sense 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Saving results

- Finally, we can **assign** our computational results back to string
- This is done using the = operator

```
>>> i = 5
>>> i + 9          # Won't change i
14
>>> i
5
>>> i = i + 9     # i = xxx Set i to be xxx
>>> i
14
```

A simple example

- Let's see a simple example which perform the temperature conversion,

```
tempF = input("Enter temperature in (F): ")
tempC = 5.0*(tempF - 32.0)/9.0
print("Temperature in (C): ",tempC)
```

- Let's see how it works out... Ops! An error!

```
>>> python3 temp.py
Enter temperature in (F): 80
Traceback (most recent call last):
  File "temp.py", line 2, in <module>
    tempC = 5.0*(tempF-32.0)/9.0
TypeError: unsupported operand type(s) for -: 'str' and 'float'
```

What is the error?

- What is the error? Let's look at the **error message** more closely:

```
    File "temp.py", line 2, in <module>
    tempC = 5.0*(tempF-32.0)/9.0
TypeError: unsupported operand type(s) for -: 'str' and 'float'
```

- According to the message, we are using the subtraction operand (−) to subtract `str` and `float`!
- But why? Isn't our input 80 a float?

What is the error?

- Is it really so? In debugging, it's always good to check our assumptions because sometimes it can be wrong!
- Let's check by printing the type of `tempF`

```
1 tempF = input("Enter temperature in (F): ")
2 print('Type of tempF: ',type(tempF))
3 #tempC = 5.0*(tempF - 32.0)/9.0
4 #print("Temperature in (C): ",tempC)
```

What is the error?

- Let's see what it returns now:

```
>>> python3 temp.py
Enter temperature in (F): 80
Type of tempF:  <class 'str'>     <---- Ops! tempF is a string!!!
```

What is the error?

- Turns out, the problem is the way `input` handles input. According to the documentation:

input([*prompt*])

If the *prompt* argument is present, it is written to standard output without a trailing newline. The function then reads a line from input, converts it to a string (stripping a trailing newline), and returns that. When EOF is read, `EOFError` is raised. Example:

```
>>> s = input('--> ')
--> Monty Python's Flying Circus
>>> s
"Monty Python's Flying Circus"
```

If the `readline` module was loaded, then `input()` will use it to provide elaborate line editing and history features.

Raises an auditing event `builtins.input` with argument `prompt` before reading input

Raises an auditing event `builtins.input/result` with the result after successfully reading input.

What is the error?

- So basically, `input` will convert everything we input, regardless of it's original form, into string!
- To battle this, we need to *manually* turn `str` back to `Int`
- This is done by wrapping `Int()` in front of `input`, i.e. from:

```
input("Enter temperature in (F): ")
```

to

```
float(input("Enter temperature in (F): "))
```

Final code

- The final code is then:

```
tempF = float(input("Enter temperature in (F): "))
print('Type of tempF: ',type(tempF))
tempC = 5.0*(tempF - 32.0)/9.0
print("Temperature in (C): ",tempC)
```

- and finally we can run!

```
>>> python3 temp.py
Enter temperature in (F): 80
Type of tempF:  <class 'float'>
Temperature in (C):  26.666666666666668
```

Some comment on type casting

- The trick we used to solve the problem is actually called **type casting**
- Type casting converts data type from one kinds to another
- In general, to *cast* a value / variable to `<data-type>`, do:

```
<data-type>(<value/variable>)
```

- For example:

```
float("1.2")      # Converts string "1.2" to float
int("12")         # Converts string "12" to int
str(1.2)          # Converts float 1.2 to string
```

Let's try an exercise

- Write a program that takes in the circumference of a circle and output the radius of the circle
- Recall that the circumference of a circle is calculated as $2\pi r$
- Now you got the circumference, how can you get back the radius?

Answer

```
    circum=float(input('Enter the circumference: '))
    PI=3.14159265354
    r=circum/(2*PI)
    print("The radius is:",r)
```

Some more exercise

- Say now you have two variables `a` and `b`, you want to swap their value
- but notice that the following does not work

```
1        a=b
2        b=a
```

- Why?

Swapping numbers

- The reason behind is that, after we evaluate the first line, `a` stored the value of `b`
- Then when evaluating the second line, `b=a` simply put back value of `b` to `b` itself
- However a common trick is that we can have another variable to temporarily storing the value of `a` to avoid forgetting it
- Now try to complete the code.

Answer

```
1    a=int(input('Give me the first number: '))
2    b=int(input('Give me the second number: '))
3    temp=a
4    a=b
5    b=temp
6    print("Now the number is swapped, a=",a,",b=",b)
```

- The above trick works for any kind of variable
- However we have an extra variable declared, which might be problematic when the variable size is really large
- Can we do it without having an extra variable?

Yes, by the power of Math

- Unsurprisingly, yes we can do it, below is the code

```
a=int(input('Give me the first number: '))
b=int(input('Give me the second number: '))
a=a+b
b=a-b
a=a-b
print("Now the number is swapped, a=",a,",b=",b)
```

Finally, Homework

- Homework 1 is posted on the course website, namely the HW1.ipynb
- it contains 3 problems, sorted in ascending order of difficulty
- You can choose to form group of size 1-4 people to work on the homework together
- submit the homework by uploading the file to here, inside the folder of `HW1 submission`
- remember to include all your group member's name in the document
- deadline: before next lesson, i.e. 9/3
- the solution will be disclosed one week after the deadline, i.e. 16/3