

STMC coding team Training

Lesson 2: Variables, data types

Tsai Yun Chen

November 2, 2022



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=`



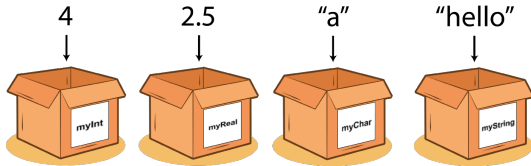
Input, process and output

- 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*

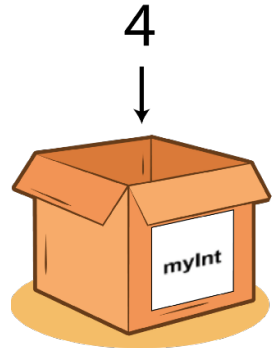


Case study I

- 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
```

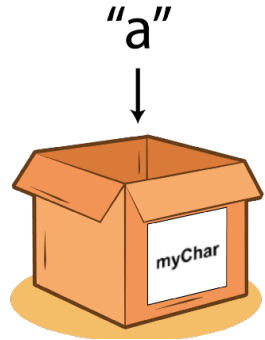


Case study II

- 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'
```

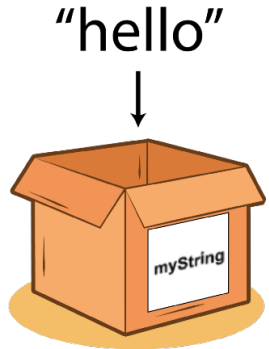


Case study III

- 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

```
1  >>> type(1)
2  <class 'int'>
3
4  >>> type(1.2)
5  <class 'float'>
6
7  >>> type('12')
8  <class 'str'>
9
10 >>> type(True)
11 <class 'bool'>
```



Checking data type

- Similarly for variables

```
1 >>> myName = "John"  
2 >>> type(myName)  
3 <class 'str'>  
4  
5 >>> temp = 30.2  
6 >>> type(temp)  
7 <class 'float'>
```



Checking data type

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

```
1 >>> age = 20
2 >>> type(age)
3 <class 'int'>
4 >>> type(age) is int
5 True
6
7 >>> myName = "rs132"
8 >>> type(myName) is str
9 True
```



Operator

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

```
1 >>> 1.0 + 2.0      # Addition
2 3.0
3 >>> 3 - 10         # Subtraction
4 -7
5 >>> 3 * 5          # Multiplication
6 15
7 >>> 4/3            # Division
8 1.3333333333333333
9 >>> 4**5           # Exponent (4**5 = 4*4*4*4*4)
10 1024
11 >>> 17//3          # Integer division, only applicable to Int
12 5
13 >>> 17%3          # Modulo/Remainder, only applicable to Int
14 2
```



Doing operations with variables

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

```
1 >>> "hello" + "bye"           # Concatenate two string
2 'hellobye'
3 >>> "hello"*3                 # Concatenate string 3 times
4 'hellohellohello'
5
6 >>> "hello" * "bye"           # Make no sense
7 Traceback (most recent call last):
8   File "<stdin>", line 1, in <module>
9   TypeError: cant multiply sequence by non-int of type 'str'
10
11 >>> "hello" - "bye"           # Make no sense 2
12 Traceback (most recent call last):
13   File "<stdin>", line 1, in <module>
14   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

```
1 >>> i = 5
2 >>> i + 9          # Won't change i
3 14
4 >>> i
5 5
6 >>> i = i + 9      # i = xxx Set i to be xxx
7 >>> i
8 14
```



A simple example

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

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

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

```
1 >>> python3 temp.py
2 Enter temperature in (F): 80
3 Traceback (most recent call last):
4   File "temp.py", line 2, in <module>
5     tempC = 5.0*(tempF-32.0)/9.0
6 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:

```
1 File "temp.py", line 2, in <module>
2   tempC = 5.0*(tempF-32.0)/9.0
3 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:

```
1 >>> python3 temp.py
2 Enter temperature in (F): 80
3 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:

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

to

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



Final code

- The final code is then:

```
1 tempF = float(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)
```

- and finally we can run!

```
1 >>> python3 temp.py
2 Enter temperature in (F): 80
3 Type of tempF: <class 'float'>
4 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:

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

- For example:

```
1 float("1.2")    # Converts string "1.2" to float
2 int("12")        # Converts string "12" to int
3 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

```
1  circum=float(input('Enter the circumference: '))  
2  PI=3.14159265354  
3  r=circum/(2*PI)  
4  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

```
1 a=int(input('Give me the first number: '))
2 b=int(input('Give me the second number: '))
3 a=a+b
4 b=a-b
5 a=a-b
6 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
- no submission is needed, the homework only serves as way for you to practice
- answer will be released two weeks later, on 29/10
- Though no submission is required, you are still encouraged to finish the homework to practice what you have learnt, so that you don't get lost in the coming lesson.

