DOING SOMETHING USEFUL WITH PYTHON

ES 112

Brief Recap: Representing Data in Python

- Expressions
- Variables
- Assignments

Menu for Today!
Doing Something with the Data

- Strings
- I/O and Formatting I/O
- Types
- Conditionals
- Iterations

What is a String

- A cord or a thread usually used to bind, fasten, or tie
- The gut, wire, or nylon cord of a musical instrument
- The gut, wire, or cord of a racket or shooting bow
- A group of objects threaded on a string (a string of fish, a string of pearls)
  - *A series of things arranged in or as if in a line*
  - *A sequence of like items (such as bits, characters, or words)*
  - *A group of business properties scattered geographically a string of newspapers*
  - *The animals and especially horses belonging to or used by one individual*

Reference: Merriam Webster Dictionary

Strings in Python

- A string is a sequence on characters
  - *A character can be a letter, special character, spaces or digits*
- A string is enclosed in quotation marks

```
hello = “hello there”
hi = ‘hello there’
hello == hi
```

- variables can be assigned string values
- Equality checking operators work with string values
- Strings cannot be changed

Strings have Structure

- Strings are inherently ordered
  - *The first character is numbered 0 and so on*
- Indexing: extract a specific character in the string

```
ringOfPower = ‘One ring to rule them all’
ringOfPower[0] : ‘O’
ringOfPower[4] : ‘r’
ringOfPower[26] : IndexError: string index out of range
```

- Slicing: extract substrings of arbitrary length

```
word1 = ringOfPower[0:3]
word2 = ringOfPower[4:8]
```

Note `ringOfPower[3]` *is not included in* `ringOfPower[0:3]`

- `len(ringOfPower)` gives the length of the string

Operations on Strings: Concatenation

- Concatenation is inverse of slicing
  - *Represented by the operator* +
  - Appends the second string at the end of the first one

```
word3 = word1 + word2
```

- Note there are no spaces in `word3`. If we want spaces, we must add them explicitly

```
word3 = word1 + ‘ ‘ + word2
```

Operations on Strings: Comparison

- We can check if two strings are the same using operator ==

```
hi = ‘Good Morning’
greet = ‘Good’
timeOfDay = ‘Morning’
hello = greet + ‘ ‘+ timeOfDay
hi == hello
```

- Strings are implicitly ordered in dictionary order

```
'apple' < 'orange'
'apple' > 'guava’
‘app’ < ‘apple’
```

- Exception: What happens with capitals?

```
‘bat' < ‘apple’
‘Bat' < ‘apple’
```

- Comparison with other types no allowed

```
‘4’ < 2
```

Operations on Strings: Multiplication

- Multiplication of a `string` by an `int` is allowed
  - *Intuitive meaning: repeat the string n time*

```
celebrate = ‘He is a jolly good fellow’
hurrah = celebrate * 2
```

- Type checking string operations: multiplication by an object of type other than an int is not allowed

```
‘a’*’b’
‘a’*3.2
```

Strings cannot be Modified

```
ringOfPower = ‘One ring to rule them all’
ringOfPower[0:3] = Two
```

- We can construct new strings with the changes we want

```
headRing = ringOfPower[0:3]
tailRing = ringOfPower[3:len(ringOfPower)]
ring2 = headRing + tailRing
```

Objects that cannot be changed are called immutable

On Quotation Marks

- Remember “hello” and ‘hello’ are the same
- You can use “” to create a string that has ‘ as a character and vice versa

```
possessiveNoun = “cat’s”
strangeSentence = ‘“I Think,” Said The Sweet Potato.’
```

- Use triple quotes ‘’’ to include line breaks in your string

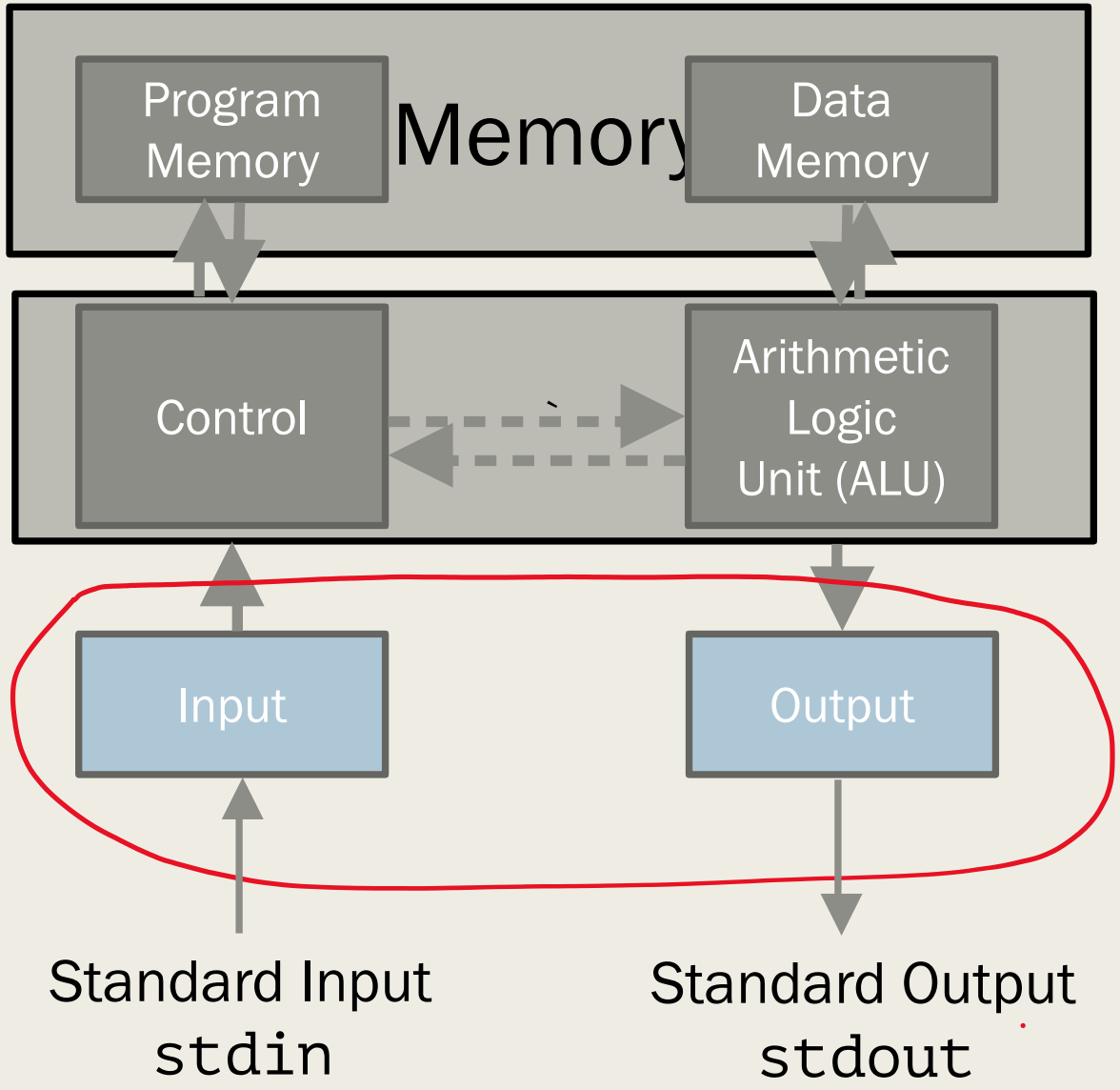
```
veryStrange = ‘’’”I think”, Said The Sweet Potato.
“Therefore, I Yam”’’’
```

- what is the difference between
  - `veryStrange`
  - `print(veryStrange)`

Interacting with the World

Remember:

```
print("Hello world")
```



Basic IO mechanism: `print`

- Output
  - *output stuff to console / stdout*
  - *Key word is* `print`

```
myNumber = 42
print("My favourite number is ", myNumber)
```

Basic IO mechanism: `input`

- Input
  - *Prints the given prompt*
  - *User types in something and hits an enter*
  - *Binds the value to a variable*
  - *Remember that the object you get has type* `string`

```
myNumber = input(“Give me a number”)
myNumber + 1
myInteger = int(myNumber)
myInteger + 1
```

```
TypeError: can only
concatenate str (not "int")
to str
```

Formatted Printing in Python

- Suppose I want to print a floating point number in a form that has space for 6 characters. I want to ensure that 2 digits after the decimal point are printed.
- Python has three methods
  1. *The modulus (%) operator on strings:*

     ```
     myString = ('You owe me Rs % 6.2f' % 72.1)
     ```

  2. *The str.format method*

     ```
     myString = 'You owe me Rs {:6.2f}'.format(72.1)
     ```

  3. The f-strings method

     ```
     amount = 72.1
     myString = f'You owe me Rs {amount:6.2f}'
     ```

I recommend you master f-strings.

Details on internet: https://docs.python.org/3/library/string.html#format-string-syntax

Also other sites such as:

https://realpython.com/python-f-strings/#f-strings-a-new-and-improved-way-to-format-strings-in-python

What does Type Mean?

type [tahyp] SHOW IPA

See synonyms for: type / typed / types / typing on Thesaurus.com

Elementary Level

noun

1. a number of things or persons sharing a particular characteristic, or set of characteristics, that causes them to be regarded as a group, more or less precisely defined or designated; class; category:
   a criminal of the most vicious type.
2. a thing or person regarded as a member of a class or category; kind; sort (usually followed by *of*):
   This is some type of mushroom.
3. *Informal*. a person, regarded as reflecting or typifying a certain line of work, environment, etc.:
   a couple of civil service types.

a **data type** or simply **type** is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data

Why Do We Need Types

- Types
  - *Constrain the values that an object / variable / expression can take*
  - *Define operations that can be done on the data*
  - *Dictates how the data is stored in memory*
- Type errors help us catch mistakes

Types tell us how make “meaning” of data

How Does Python Handle Data Types

- Python is dynamically typed

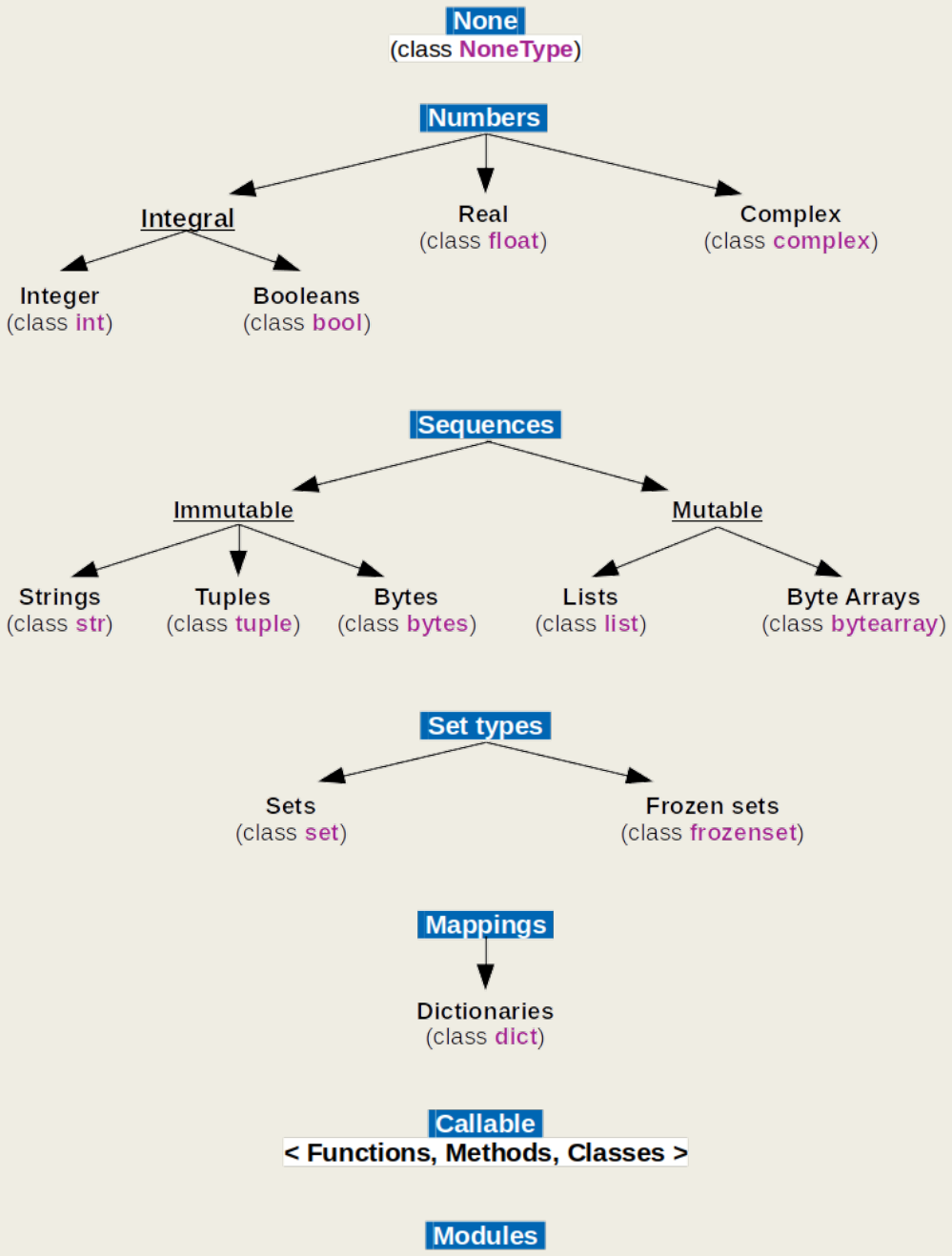
```
>>> 2 > “1”
>>> (2 > 1) and (2 > “1”)
>>> (2 > 1) or (1 > “1”)
```

- Python is strongly typed
  - *Implicit conversions from `int` to `float`*
  - *`Implicit conversion only between few built in data types`*

```
>>> 2.5 + 3
>>> 2 > int(”1”)
>>> True and 1
>>> True and 0
```

Much more on Types later

The Standard Type Hierarchy of Python 3



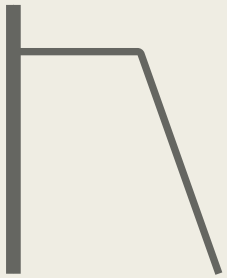
Making Decisions

```
if (condition):
    expression1
    expression2
else:
    expression3
    expression4
expression5
```

- Indentation is important
  - *Try what happens if you indent wrong*
- Else is optional

Ramukaka in Python

```
age = int(input("Please tell me your age :"))
if (age <= 0):
    print("Boss, you need to be born first")
else:
    if (age > 0 and age <= 10):
        print("Let me give you glass of milk")
    else:
        print("Would you like tea or coffee?")
        drink = input()
        if (drink != 'Tea' and drink != 'Coffee'):
            print("I am sorry, I don’t have ", drink)
        else:
            print("Please have some ", drink)
```



Too many levels of indentation makes program difficult to read

Making Multiple Decisions

```
if condition1:
    expression1
    expression2
elif condition2:
    expression3
    expression4
elif condition3:
…
else:
    expressionN
```

Ramukaka once again

```
age = int(input("Please tell me your age :"))
if (age <= 0):
    print("Boss, you need to be born first")
elif (age > 0 and age <= 10):
    print("Let me give you glass of milk")
else:
    print("Would you like tea or coffee?")
    drink = input()
    if (drink != 'Tea' and drink != 'Coffee'):
        print("I am sorry, I dont have ", drink)
    else:
        print("Please have some ", drink)
```