The Very First Course of Python

Changye Li

[1]Institute of Health Informatics Student Group (IHI-ISG)
University of Minnesota

June 2021

What We Will Do Today

What is IHI-ISG?

- We the official student group of Institute of Health Informatics (IHI)
- We serve as an IHI community liaison to other UMN Student Groups or Affiliates
- Our goal is to foster a community with a shared interest in the field of health informatics
- We encourage the academic and career development of students in the field of health informatics
- We offer networking and connecting students through information exchange on collaboration, workshops, and potential professional opportunities

What is Programming Club?

- We are a subcommittee of ISG
- Our goal is to teach IHI students to programming on the introduction level
- We want IHI students to learn the ability of self-learning with programming with advanced topics
- Students will be able to program on the basic level, and able to learn more advanced topics (i.e., complex syntax, advanced application) on their own

Who am I?

- Changye Li
- IHI Ph.D. student, Data Science and Informatics for Learning Health Systems track
- My research interest is Natural Language Processing (NLP) application on health and medical domain
- B.A. in Statistics, M.S. in Data Science

ISG-UMN
IHI STUDENT GROUP

Python Installment, Version, IDE

- Download Python *at least* **3.6** from Python official website
- **Always start with a virtual environment!**, but not required for this workshop
- You will need this guide to create a virtual environment with terminal
- Chose your IDE, i.e., Visual studio code or PyCharm
  - Your python script extension is `.py`
- Jupyter notebook is also one of my favorite
  - Your jupyter notebook extension is `.ipynb`
- You can also try Python online compiler like this one

Name Your Data Type

Python is an Object-oriented programming language, which means we need to store our data types as object – *object references* or *variables*

```
x = 55414 # int
y = "Hello World" # string
z = x
print(x, y, z)
```

What's the output of code above?

ISG-UMN
IHI STUDENT GROUP

Rules of Naming Variables

- The start character should be a Unicode letter ("a", "b", ... "z", "A", "B", ..., "Z")
- The names of variables are case-sensitive. `var1`, `Var1`, `VAR1` are three different variables
- You can use underscore for your variables. For example `var_1`, but not for the first letter of your variable
- You cannot use Python keywords and predefined variables for your variables
- Make your variables easy to read and understandable. In general, don't name your variables like `var1`, `x`, `y`, `z`

ISG-UMN
IHI STUDENT GROUP

Short Quiz!

Which one is **NOT** a legal variable name in Python?

A `MyVal`

B `_my_val`

C `my-val`

D `myval1`

Python Keywords and Predefined Variables

and	continue	except	global	lambda	pass	while
as	def	False	if	None	raise	with
assert	del	finally	import	return	nonlocal	return
yield	break	elif	for	in	not	True
class	else	from	is	or	try	sum
super	tuple	type	zip			

Table: Python keywords and predefined variables

And many more ↓

```
print(dir(__builtins__))
```

Python Common Built-in Data Types

- Integers (`int`): 1, 2, 3
- Booleans (`bool`): `True`, `False`
- Floats (`float`): 1.0, 2.0, 1.5
- Strings (`str`): "Hello World"
- Range (`range`): generates consecutive numbers with given range
- Tuples (`tuple()` or `(,)`): `(1.0, 2, True, "Hello World")`
- Lists (`list()` or `[]`): `[]`, `[1.0, 2, True, "Hello World"]`
- Sets (`set()`): `{1.0, 2, True, "Hello World"}`
- Dictionaries (`dict()`): `{1:1.0, "2": 2, "value": True, "message":"Hello World"}`

Other Python Built-in Data Types Not Mentioned Here

- complex
- frozenset
- bytes
- bitarray
- memoryview

Immutable and Mutable Variables

Whenever an object is instantiated, it is assigned a unique object ID. If the object can change its state or content, then it is called mutable variable, otherwise it is immutable

- Immutable objects:
  `int, float, boolean, string, unicode, tuple`
- Mutable objects: `list, dict, set`

```
tuple_1 = (1, 2, 3, 4)
tuple_1[0] = 5
print(tuple_1)
message = "Hello World"
message[0] = "X"
print(message)
color = ["red", "blue", "gray"]
color[0] = "yellow"
print(color)
```

Integers

Syntax	Description
`x+y`	adds number `x` and number `y`
`x-y`	subtracts `y` from `x`
`x*y`	multiplies `x` by `y`
`x/y`	divides `x` by `y`; always returns `float`
`x//y`	divides `x` by `y`, returns an `int` that truncates any fractional part of the result
`x%y`	returns the modules of `x/y`
`x**y`	x^y, `pow(x, y)`
`x==y`	returns `True` if two integers are the same
`abs(x)`	returns the absolute value of `x`

Table: Numeric operators and functions

ISG-UMN
IHI STUDENT GROUP

Booleans

```
x = True
y = False
print(x and y)
print(x and True)
```

Booleans (Cont.)

Operation	Results
`x or y`	if x is false, then y, else x
`x and y`	if x is false, then x, else y
`not x`	if x is false, then true, else false
<	strictly less than
<=	less than or equal
>	strictly greater
>=	greater than or equal
==	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

Table: Python Boolean Operations and comparisons

Truth Table: AND

Let's say T for True, F for False

x	y	x and y
F	F	F
F	T	F
T	F	F
T	T	T

Table: Truth Table: AND

Truth Table: OR

x	y	x or y
F	F	F
F	T	T
T	F	F
T	T	T

Table: Truth Table: OR

ISG-UMN
IHI STUDENT GROUP

Truth Table: NOT

x	not X
F	T
T	F

Table: Truth Table: NOT

Floats

- Three types of floating-point value: `float`, `complex`, `decimal.decimal`
- `float` can be assigned with exponential notation: `print(8.9e-4)`

What's the difference between `1` and `1.0`?

Try `print(1 == 1.0)`

Strings

- String is a sequential text data

Strings can be created by single or double quotes, or simply `str()`

```
x = ""
y = str()
z = "Hello World"
```

But what if we want to create a string like this one with **one** string variable:

```
Hello
        World
I'm using Python
```

Python Common String Escapes

Escape	Meaning
\\	backslash (\)
\'	single quote (')
\n	end-of-line terminator, Unix OS
\r	end-of-line terminator, old Mac OS
\r\n	end-of-line terminator, Windows and DOS OS
\b	backspace
\t	tab
\v	vertical tab

Table: Python Common String escapes

Programming quiz: use appropriate escape(s) to generate the above string with **one** variable

String Operations

Let's say `s = "Light ray"`. What if I want to extract individual characters?

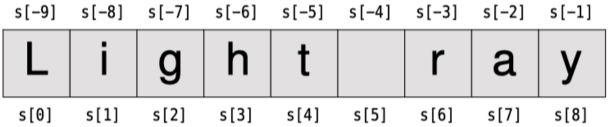

Figure: String index positions

- What if I want to extract `Light`? `s[:5]`
- What if I want to extract `ray`? `s[:-3]`
- What if I want to extract every two characters of `s` (`Lgtry`)? `s[::2]`

Short quiz: what if I want to extract `ght`?

More on Strings

A sting object s

Syntax	Description
`len(s)`	returns the length of s
`s.capitalize()`	returns a copy of s with the first letter capitalized
`s.endswith(x, start, end)`	returns True if s (or during the start:end of s) ends with pattern x
`s.startswith(x, start, end)`	returns True if s (or during the start:end of s) starts with pattern x
`s.format()`	returns a copy of s formatted according to the given arguments
`s.lower()`	returns a copy of s that all characters are lowered

Table: More on String Operations

More on Strings (Cont.)

```
txt1 = "Hello Changye!"
print(txt1)
fname = input("Enter your first name:")
txt2 = "Hello {}!".format(fname.capitalize())
print(txt2)
```

Tuples

- Tuple is a sequential collection data type
- Tuple is ordered sequence data type

```python
# tuple with more than 1 item
t = ("venus", -28, "green", "21", 19.74)
# tuple with 0 item
t = ()
# tuple with 1 item
t = ("venus",)
```

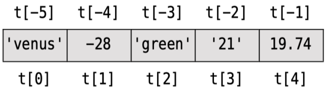

Figure: Tuple index positions

Lists

- List is a mutable, ordered sequence
- List uses the same index/sliding syntax as strings and tuples
- `l = []`
- `m = list()`

More on List

Syntax	Description
`l.append(x)`	appends item `x` to the end of list
`l.extend(m)`	appends all of iterable `m` 's items to the end of list `m`. Equivalent to `l += m`
`l[::-1]`	returns the reversed of `l`. Equivalent to `l.reverse()`
`l[1:]`	returns `l` with first item removed
`l[:-1]`	returns `l` with last item removed
`min(l)`	returns the minimal value of `l` if all items are digital numbers
`max(l)`	returns the max value of `l` if all items are digital numbers
`len(l)`	returns the length of `l`
`sum(l)`	returns the sum of all items for `l`, if all items are digital numbers

ISG-UMN IHI STUDENT GROUP

Table: List Operations

List Comprehension

I want to iterate a list to do some operations. What should I do? For example, for a list with all integers, I want to add 1 to every item.

```
l = [1, 2, 3, 4]
# the "old fashion" way
new_l = []
# use for loop to iterate list
for item in l:
    new_l.append(item+1)
print(new_l)
# the Python way
l = [item+1 for item in l]
print(l)
```

List Comprehension Short Quizzes

What if I want to add 1 to **odd** items only?
What if I want to keep **even** items only from `l`?

```
[expression for item in iterable if condition]
[expression for item in iterable]
```

Sets

- Set is un-ordered collections that each item in set is hashable
- The value does not change during its lifetime, therefore both the items and their hash values are unique
- We cannot access items inside a set with indexes
- `s = {7, "veil", 0, -29, ("x", 11) [1, "hello"], 913}`
- `s = set()`

What will a set, `s = {7, "veil", 0, -29, ("x", 11) [1, "hello"], 913, "veil", 7}` look like?

Set Operations

Syntax	Description
`s.add(x)`	appends item `x` to the set
`s.clear()`	remove all items from the set
`s.difference(t)`	returns a new set that has every item that is in set `s` that is not in set `t`. Equivalent to `s-t`
`s.intersection(t)`	returns a new set that has each item that is in both set `s` and set `t`
`s.remove(x)`	remove item `x` from the set
`s.issubset(t)`	returns True if set `s` is equal to or a subset of set `t`. Equivalent to `s <= t`

Table: Set Operations

Set comprehension also works, similar syntax to list comprehension

Dictionaries

- Dictionary is an un-ordered, mutable collection data types with key-value pairs.
- Keys are hashable (unique), but not for values
- `d = dict()`
- `d = {}`
- `d = {"name": "changye", "major": "health informatics", "id": 1234}`

ISG-UMN
IHI STUDENT GROUP

Dictionary Operations

Syntax	Description
`d.get(k)`	return the value associated with `k`
`d.items()`	returns the view of key-value pairs
`d.keys()`	returns the view of all keys of `d`
`d.values()`	returns the view of all values of `d`

Table: Dictionary Operations

The “view” is a read-only iterable object
Dictionary comprehension also works, similar syntax to list comprehension

Data Type Conversion, Or Type Casting

- Implicit type conversion: Python automatically convert data types to others
- Explicit type conversion: we need to convert data types manually

Type Casting (Cont.)

```
x = 10
y = 10.1
print("x is type {}".format(type(x))
print("y is type {}".format(type(x))
x = x+y
print("x is type {}".format(type(x))
s = "10010"
c = int(s, 2)
print("{} is {} after converting to int base 2".format(s,
↪  c))
e = float(s)
print("{} is {} after converting to float".format(s, e))
c = tuple(s)
print("{} is {} after converting to tuple".format(s, c))
l = list(s)
print("{} is {} after converting to list".format(s, l))
```

How to Check Data Type

What do we need to know before casting data type conversions?

- Data type of the variable before conversion
- Desired data type

What if I don't know the data type before conversion?

```
x = 5
# suppose I forget it
# is it int?
if isinstance(x, int):
    print("yes it is an integer")
```

ISG-UMN
IHI STUDENT GROUP

Conditional Loops

```
if boolean_expression_1:
    suite_1
# if you have more than one condition
elif boolean_expression_2:
    suite_2
# you can have zero or more elif causes
...
# else cause is optional
else:
    else_suite
# if you only have two conditions
expression_1 if boolean_expression else expression_2
```

Conditional Loop (Cont.)

Let's say we have 3 students entering different academic programs: 1. Mark, PhD; 2. Mary, Master; 3. Ben, Bachelor. We would like to write a simple script to greet students with their belonging programs (`Welcome StudentName to our ProgramLevel program!`). Assume we only have 3 students, and each student has a unique name. The names are **NOT** case sensitive.

```python
fname = input("Enter your name:")
# write your script below
# hint: remember what we just learned on string operations
```

IHI STUDENT GROUP

while Loop

We can execute a set of statements as long as a condition is true using while loop

```
while boolean_suite:
    while_suite
# else cause is optional
else:
    else_suite
# example
i = 1
while i < 6:
    print(i)
    i += 1
```

ISG-UMN
IHI STUDENT GROUP

for Loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string)

```
for expression in iterable:
    for_suite
# examples
fruits = ["apple", "peach", "pear"]
s = "Hello World"
for item in fruits:
    print(item)
for item in s:
    print(item)
# what if I want the value and its associated index?
for index, value in fruits:
    print(index, value)
```

ISG-UMN
IHI STUDENT GROUP

Leave Loop: `break`, `continue`, `pass`

I've found what I looked for from a loop, then what should I do next?

- Continue to run your script
- "Stop" the loop using `break`, `continue`, `pass`
  - `break`: terminates the current loop and resume execution at the next statement
  - `continue`: "ignore" all remaining statements in the current iteration and move the control to the top of the loop
  - `pass`: something is required, but you don't have anything else to execute

Leave Loop (Cont.)

```
for letter in "python":
    if letter == "h":
        break
    print("current letter is {}".format(letter))
print("======================")
for letter in "python":
    if letter == "h":
        continue
    print("current letter is {}".format(letter))
print("======================")
for letter in "python":
    if letter == "h":
        pass
    print("current letter is {}".format(letter))
```

Quizzes on Various Topics

- Given a list `l = list(range(11))`, find the **odd** numbers and save it to a new list, and print the new list
- What's the output of the following snippets? Don't copy and paste and run them on your laptop!

```
i = 1
while i<=10:
    if(i%2==0):
        print(i)
    i+=1
```

Quizzes on Various Topics

What is the correct syntax to output Hello World in Python?

- `echo 'Hello World'`
- `print('Hello World')`
- `print('hello world')`
- `echo('Hello World')`

Quizzes on Various Topics

What is the correct syntax to comment in Python?

- `/* this is a comment */`
- `## this is a comment`
- `// this is a comment`

What is the correct file extension for Python script?

- .py
- .pyc
- .pt

Quizzes on Various Topics

What is the correct syntax to output the type of a variable or object in Python?

- `print(type(obj))`
- `print(typeof(obj))`
- `print(typeOf(obj))`
- `print(typeof obj)`

ISG-UMN
IHI STUDENT GROUP

Short Assignments

- Given an array of integers *nums* and an integer *target*, return indices of the two numbers such that they add up to *target*. For example, a list of `nums = [2,7,11,15], target = 9`, your script should return `nums = [0, 1]`. How about `nums = [3,2,4], target = 5`?
- Given an integer, return the integer that every single digit is reversed. For example, with input of 321, your script should return 123

Write Your Own Functions

- A function is a block of code which only runs when it is called. You can pass data, known as parameters or arguments, into a function
- When you need to repeatedly use a piece of code more than 3 times, make it as function

```
def my_function(fname):
    print("Hello {}!".format(fname))
my_function("Changye")
# parameter can have default values
def my_function(fname="Changye"):
    print("Hello {}!".format(fname))
my_function()
```

Python PEP 8 - Python Enhancement Proposals: Write Your Code Easy to Read and Review

- Detailed plan here
- Install pylint under your environment for suggestions
- Use 4 spaces or 1 tab per indentation level, but **NOT** the mixture of spaces and tabs
- Limit all lines to a maximum of 79 characters.
- Avoid trailing whitespace anywhere

Python PEP 8: Naming Styles

Type	Naming Convention	Examples
Function	Use lowercase word(s). Separate words by underscores to improve readability	`function`, `my_function`
Variable	Use lowercase single letter, word(s). Separate words with underscores	`v`, `var`, `my_variable`
Class	Start each word with a capital letter. **Don't** separate words with underscores	`Model`, `MyClass`
Method	Use lowercase word(s). Separate words with underscore	`class_method`, `method`
Constant	Use an uppercase single letter, word(s). Separate words with underscores	`CONSTANT`, `MY_CONSTANT`

Table: Python PEP 8 Naming Styles

Comment Your Code, Write Documentation if Necessary

Take the reversed integer as an example

```
def reverse_int(input_int):
    # one-line docstring
    """Return the reversed integer with a given input
    ↪  integer."""
    # multiple lines of docstring
    """
    This is a reST style.

    :param input_int: the given integer to be reversed
    :returns: the reversed of given integer
    :raises keyError: raises an exception, we will talk
↪   about it very soon
    """
```

Comment Your Code, Write Documentation if Necessary (Cont.)

```python
def reverse_int(input_int):
    """
    This is an example of Google style.

    Args:
        input_int: description of this parameter

    Returns:
        This is a description of what is returned.

    Raises:
        KeyError: Raises an exception.
    """
```

Comment Your Code, Write Documentation if Necessary (Cont.)

```
def reverse_int(input_int):
    """
    This is sphinx style

    :param input_int: [ParamDescription], defaults to
↪   [DefaultParamVal]
    :type input_int: int

    :raises [ErrorType]: [ErrorDescription]

    :return: [ReturnDescription]
    :rtype: [ReturnType]
    """
```

More Programming Assignments

- Write a function to find if the input integer is palindrome. That's being said, if the input integer can be read the same backward as forward. For example, with input of 121, your function should return `True`, but with input of -121, it should return `False`

Python Anonymous function

`lambda arguments: expression` is an anonymous function that is same as a regular python function but can be defined without a name

- reduce number of lines of codes compared to `def`
- function which is needed temporarily
- call it immediately at the end of definition

```
squares = lambda x: x*x
print("using lambda to get squares: {}".format(squares(2))
def squares(x):
"""
Ignore docstring for now
"""
    return x*x
print("using def to get squares: {}".format(squares(2))
```

filter(), map(), reduce()

```
# given a list, return the even numbers from the list
mylist = [2,3,4,5,6,7,8,9,10]
list_new  = list(filter(lambda x : (x%2==0), mylist))
print(list_new)
list_new  = list(map(lambda x : x%2, mylist))
print(list_new)
sum = reduce((lambda x,y: x+y), list1)
print(sum)
```

Quiz and Short Assignments

What is the output of the following code?

```
words = ["bay", "cat", "boy", "fan"]
b_words = list(filter(lambda word: word.startswith("b"),
↪  words))
print(b_words)
words = ["mary", "had", "a", "little", "lamb"]
biggest_word = reduce(lambda x, y: x if len(x) > len(y)
↪  else y, words)
print(biggest_word)
```

Calculate the sum of `l=[1, 4 ,5, 2]`. `sum() is not allowed`

Error Handling

Python indicates errors and exceptional conditions by raising exceptions

```
print(6/0)
```

ISG-UMN
IHI STUDENT GROUP

Error Handling (Cont.)

Sometime it is not your fault, but sometime it is!

```
try:
    try_suite
except exception_group1 as variable1:
    except_suite1
...
finally:
    finally_suite
```

Error Handling Example

```
lst = [4, 5, 0, 1]
def list_find(lst, target):
    try:
        index = lst.index(target)
    except ValueError:
        index = -1
    return index
list_find(lst, 2)
```