Review

Function structure (header and body)

Namespace and local variable

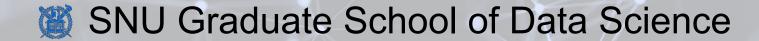
- What happens when you call a function
- Guidelines for writing a new function

Computing Bootcamp

Strings

Lecture 3-1

Hyung-Sin Kim



Programming for Big "Data"

 One of the main goals for this course is for you to handle various types of data more easily

• Yes, you need to be familiar with various types of data and how to represent and handle them

We will see representative data structures provided by Python

Let's start from strings!

String Type

- Recall that Python uses int and float types to represent <u>number</u> values
- Python defines another type, **string (str)**, to represent <u>text</u> values
 - Text is a sequence of characters (letters, digits, and symbols)
- Python recognizes that a value is string if it is surrounded by "or ""
 - Programming Foundations'
 - "Programming Foundations"
- 19 vs. "19"

Built-in Operations on Strings

- len('string'): Number of characters of the string
 - len('Programming for Data Science ^0^/')
 - 33

- 'string' + 'string': Concatenation
 - "I" + "" + "don" + 't' + 'like ' + "COVID" + '-' + '19'
 - I don't like COVID-19

- "string" * num: Repetition
 - "(--)()" * 5
 - "(--)(__)(--)(__)(--)(__)"

Built-in Operations on Strings

- Type changes are possible
 - $int('1') \rightarrow 1$
 - float('-234.2') \implies -234.2
 - $str(5) \rightarrow 5$

- Strings are values, so you can assign a string to a variable
 - my_name = "Hyung-Sin Kim"
 - len(my name) \rightarrow 13
 - my_name * 2 → "Hyung-Sin KimHyung-Sin Kim"
 - my_name + "teaches this course" → "Hyung-Sin Kim teaches this course"

Special Characters in Strings

- I'm studying
 - "I'm studying"
- I said "I'm studying"
 - ???? Need another way
 - "I said \"I\'m studying\\""
- Escape sequence (sequence escaping from Python's usual syntax rules)
 - \': Single quote
 - \": Double quote
 - \\: Backslash
 - \t: Tab
 - \n: Newline
 - \r: Carriage return

Printing

- print $(1+1) \rightarrow 2$
- print("I like this.") \rightarrow I like this.
- print $(1, 2, 3) \rightarrow 123$
- radius = 3
- print("The diameter of the circle is", radius * 2, "m.")
 - → The diameter of the circle is 6 m.
- print("Name\tNationality\nKim\tKorean\nCuller\tAmerican")
 - Name Nationality
 - Kim Korean
 - Culler American

Getting Input from the Keyboard

- a = input()
 - Computer waits for you to type something
 - Whatever you type, Python represents the value as a **string**
 - a = input()
 - 10438482
 - a
 - '10438482'
- Input can get a string argument, which is used to prompt the user for input
 - > name = input("Please enter your name: ")
 - Please enter your name: <u>Hyung-Sin Kim</u>
 - > name
 - 'Hyung-Sin Kim'

Summary

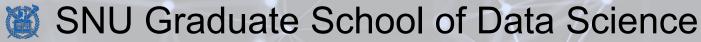
- String values and operations
- Special characters

Print and Input

Control Structures - Boolean Types

Lecture 3-2

Hyung-Sin Kim



Introduction

- Making choices is a fundamental concept of programming
- We do this whenever we want our program to behave differently depending on the data it's working with
 - Ex 1) Depending on whether a user types yes or no
 - Ex 2) Depending on whether the number of students is less than 40 or not
- Today, you will learn
 - Control flow statements which are for making choices
 - These statements involve another Python type called **Boolean** that represents truth and falsehood

Boolean Type

- Type "bool" has only two values, True and False
- Boolean operators: not, and, or
- Boolean variables and operations
 - >>> cold = True
 - >>> windy = False
 - >>> (not cold) and windy
 - False
 - >>> not (cold and windy)
 - True

Relational Operators

- A comparison using a relational operator results in a bool-type value
 - >, <, >=, <=, ==, !=
 - WARNING: == is for equality but = is for assignment
- Examples
 - >>> $22 > 10 \implies True$
 - $>>> 30 > 40 \implies \text{False}$
 - >>> 55 == 55 \longrightarrow True
 - >>> $56 != 56 \implies False$
- Useful function using relational operators
 - def is positive(x: float) -> bool:
 - return x > 0

More Complex Comparisons

- Combining comparisons:
 - Precedence: arithmetic operators / relational operators / Boolean operators
 - >>> True and (3 == 3) and (7 > 3+4) \implies False
 - PRACTICE: Use parentheses whenever you think your expression may not be clear

- Short-circuit evaluation
 - Python draws a conclusion fast if it is obvious, without evaluating further
 - >>> (2 > 3) and (5 > 7) and (5 == 5)
 - >>> (3+5 == 8) or (1/0)
 - >>> (0 > 1) and duguwe384ihoslslsjlkjsdlfijoijeroijhpojfdslkmglkl_sls930kgk

ASCII Code and Comparing Strings

- Character encoding standard
 - Represented by integers
 - $>>> \operatorname{ord}(\text{``A''}) \implies 65$

- Comparing strings (dictionary ordering)
 - >>> "A" < "B" → True
 - >>> "A" < "a" ->> True
 - >>> "abc" < "abd" → True
 - >>> "abdaaa" < "abc" → False
 - >>> "abc" < "abcd" → True

DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç
2	•	33	!	65	Α	97	а	129	ü
3	•	34	"	66	В	98	b	130	è
4	•	35	#	67	C	99	c	131	â
5	•	36	\$	68	D	100	d	132	ä
6	*	37	%	69	E	101	e	133	à
7	•	38	&	70	F	102	f	134	å
8	•	39	•	71	G	103	g	135	ç
9	0	40	(72	Н	104	h	136	ê
10	O	41)	73	1	105	i	137	ë
11	3	42	*	74	J	106	j	138	è
12	\$	43	+	75	K	107	k	139	ï
13	2	44	,	76	L	108	1	140	î
14	. 1	45	-	77	M	109	m	141	ì
15	₩	46		78	N	110	n	142	Ä
16	>	47	1	79	Ο	111	o	143	Å
17	◀	48	0	80	Р	112	р	144	È
18	‡	49	1	81	Q	113	q	145	æ
19	!!	50	2	82	R	114	r	146	Æ
20	¶	51	3	83	S	115	s	147	ô
21	§	52	4	84	Т	116	t	148	ö
22	_	53	5	85	U	117	u	149	ò
23	1	54	6	86	V	118	v	150	û
24	<u></u>	55	7	87	w	119	t	151	ù
25	\downarrow	56	8	88	х	120	х	152	ÿ
26	\rightarrow	57	9	89	Υ	121	у	153	Ö
27	←	58	:	90	Z	122	z	154	Ü
28	L	59	;	91]	123	{	155	ø
29	\leftrightarrow	60	<	92	١	124	1	156	£
30	A	61	=	93]	125	}	157	Ø
31	▼	62	>	94	^	126	~	158	×
		63	?	95	_	127	Δ	159	f

A String inside Another String

- "in" operator (case sensitive)
 - >>> "Sep" in "09 Sep 2020" → True
 - >>> "Jan" in "09 Sep 2020" → False
 - >>> "" in "abc" ->> True

- An example
 - >>> birthday = input("Enter your birthday in the format DD MTH YYYY:")
 - Enter your birthday in the format DD MTH YYYY: <u>01 Jan 2000</u>
 - >>> "Jan" in birthday → True

Summary

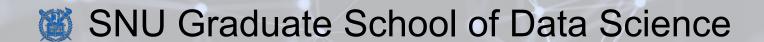
- Boolean type
- Relational operators and comparisons

• "in" operator

Control Structures - If/Else

Lecture 3-2

Hyung-Sin Kim



If Statement

- if statement lets you change how your program behaves based on a condition
 - if <<condition>>:
 - <<blook>>
 - The block must be indented (similar to function body)
 - The block is executed **only when** the condition is True
- More complex forms
 - if <<condition1>>:
 - <<blook 1>>
 - elif <<condition2>>:
 - <<blook2>>
 - else:
 - <<bl><<<bl>block3>></bl>

If Statement

- Example (Let's do it together)
 - >>> time = input("Enter the current time in the format HH:MM: ")
 - Enter the current time in the format HH:MM: 11:15
 - >>> if time < "11:00":
 - ... print("Before Programming Foundations")
 - >>> elif time < "12:15":
 - ... print("During Programming Foundations")
 - >>> else:
 - ... print("After Programming Foundations")

Nested If Statements

- **if** statements in another if statement
 - if <<condition1>>:
 - <<blook 1>>
 - if <<condition1-1>>:
 - <<bl/>block1-1>>
 - elif << condition 1-2>>:
 - <<bl/>block1-2>>
 - else:
 - <
block1-3>>
 - else:
 - <<blook2>>

Nested If Statements

• Example (assume that you have two variables **age** and **bmi**)

```
• young = (age < 45)
 slim = (bmi < 22.0)
 if young:
       if slim:
            risk = "low"
       else:
            risk = "medium"
  else:
       if slim:
            risk = "medium"
       else:
            risk = "high"
```

Summary

• "if" and "else" statements

Nested if statements

Thanks!