Python(3) Cheat Sheet

Instructor: Farbod Parvin

Command Prompt Basics

- Accessing another drive:
  - from "C:" to "D:“ ☞ type ‘`d:`’

```
C:\Users\Farbod>d:

D:\>
```

- Changing the directory:
  - from “C:” to “C:\windows\system32\” ☞ type ‘`cd windows\system32`’

```
C:\>cd windows\system32

C:\Windows\System32>
```

- Going one folder up:
  - from “C:\windows\system32\” to “C:\windows\” ☞ type ‘`cd..`’

```
C:\>cd windows\system32

C:\Windows\System32>
```

Command Prompt Basics (Python)

- Running python interpreter: 👉 type `‘python’`
- Exiting interpreter: 👉 type `‘quit()’`
- Running a python script: 👉 type `‘python my_file.py’`
  - Note: you should be in the file directory!!!

Numbers

- Types:
Name	Type	Example
Integer	`int`	`21  100  -5`
Floating point	`float`	`21.0  42.32  0.12`
Complex	`complex`	`2+3j 100+0j`

- Type Conversion:
  - `2 + 3.0` ☞ returns `5.0`
  - `2 + int(3.0)` ☞ returns `5`
  - `3 / 2` ☞ returns `1.5`
  - `3 // 2` ☞ returns `1`

Operators			
Sum `+`	Subtract `-`	Multiply `*`	Division `/`
Integer division `//`	Mod `%`	Power `**`	

Variables

- Definition: a name attached to a particular object
- Python is a dynamically-typed language (as opposed to a statically-typed language)
- Dynamically-typed: type of variables need **not** be declared or defined in advance
- Give a variable a name that is descriptive enough to make clear what it is being used for
- Naming conventions:
  - uppercase and lowercase letters (A-Z, a-z)
  - digits (0-9)
  - underscore character (_)
  - Snake Case: Words (lower-cased) are separated by underscores
    - Example: number_of_iterations = 20

Strings

- Creation: `my_str = “Hello World”` or `‘Hello World’`
- Printing: `print(my_str)` ☞ returns `“Hello World”`
- Print formatting: ☞ returns `‘inject text: hello and world’`
  - Old: `‘inject text: %s and %s’ % (‘hello’, ‘world’)`
  - New: `‘inject text: {} and {}’.format(‘hello’, ‘world’)`
- Indexing: `my_str[1]` ☞ returns `‘e’` (zero-based index)
  - My_str[-1] ☞ returns ‘d’

	H	e	l	l	o
ind	0	1	2	3	4

- Slicing: my_str[1:4:1] ☞ returns ‘ell’ (grab from index 1 up to, but not including 4, with the step of 1)
- Length: len(my_str) ☞ returns 11
- Concatenation: my_str + ‘ farbod’ ☞ returns “Hello World farbod”
- Repetition: `‘a’*10` ☞ returns `‘aaaaaaaaaa’`

Strings (Indexing)

	H	e	l	l	o
ind	0	1	2	3	4
ind	-5	-4	-3	-2	-1

- `my_str = “Hello”`
- `my_str[1:4]` 👉 “ell” | `my_str[:4]` 👉 “Hell” | `my_str[1:]` 👉 “ello”
- `my_str[:-3]` 👉 “He” | `my_str[-3:]` 👉 “llo”
- Reversing: `my_str[::-1]` 👉 “olleH”

Commenting

- Single line comment (#):
  - `# tozihaat dar yek khat`

- Multi-line comment (‘’’)
  - ```
    ‘’’ tozihaat be sharh e zir ast:
    khate 1
    khate 2 ‘’’
    ```
    ```