python™

QUICK REVIEW

March 21, 2024

TOPICS COVERED

- Why Python
- Coding Setup
- Variables
  - Random Integer From One to N
- Strings
  - String Slicing
  - String Manipulation
  - Word Count
- Dictionaries
  - Dictionary of Cell Phone Numbers and Their Owners
- Classes and Objects (OOP)
  - Class That Specifies Different Types of Footballs
  - In-Memory Database of Restaurants

WHY PYTHON

Python is the most in-demand programming language, it is easy to use, and has libraries which greatly extend its capabilities.

IN-DEMAND

- Rated most in-demand language by PYPL and TIOBE indexes [1]

EASY TO USE

- High-Level Language
- GPT-4 and LLMs very familiar with Python
- Simple OOP/scripting language

LIBRARIES

- Pandas, NumPy
- OpenCV, Pytorch, Keras, TensorFlow
- Requests, Tkinter

CODING SETUP

Each year around October a new version of Python tends to be released.

The current version (Python 3.12) can be downloaded here:
https://www.python.org/downloads/

When installing on Windows be sure to disable the MAX_PATH length:
https://docs.python.org/3/using/windows.html#removing-the-max-path-limitation

CODING SETUP

Python code can be written with:

- Text editor
  - Sublime Text
  - Notepad++
- Integrated developer environment
  - Visual Studio Code
  - IDLE (official IDE)
  - Eclipse with PyDev (open-source)
  - PyCharm (full-feature)
- Jupyter Notebook

VARIABLES

PEP 8 is the official style guide for Python:
https://peps.python.org/pep-0008/#introduction.

Variables and most things in Python are recommended to be named in lowercase with underscores between words (snake case).

Variables can be declared without needing to state their data type.

```
# this line is a comment and below is an integer variable
car_speed_mph = 42
```

VARIABLES

Python 3.12.1 has 8 different built-in data types with many subtypes. [2]

```
Text Type:          str
Numeric Types:      int, float, complex
Sequence Types:     list, tuple, range
Mapping Type:       dict
Set Types:          set, frozenset
Boolean Type:       bool
Binary Types:       bytes, bytaearray, memoryview
None Type:          NoneType
```

VARIABLES

Example - Random Integer From One to N

```
import random

def main():
    random_int_from_one_to_n(7)

# Shows a random, whole number between 1 and n inclusive
def random_int_from_one_to_n(n):
        if n < 1:
                print('Heyyy, that number is less than 1')
                return
        random_int = random.randrange(1, n+1)
        print(f'Here\'s a random integer from 1 to {n}:  {random_int}')

if __name__ == '__main__':
    main()
```

Output:

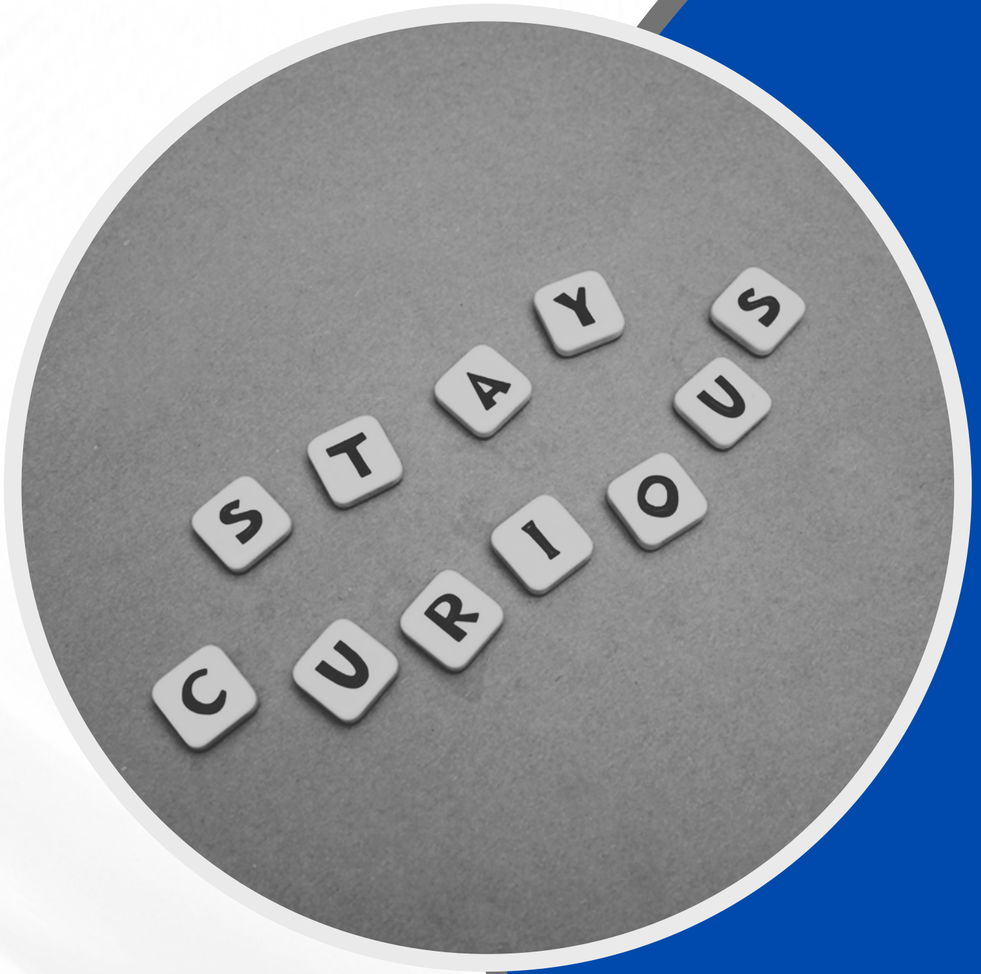
```
Here's a random integer from 1 to 7:  2
```

STRINGS

Python doesn’t have a character data type.

Instead, Python makes strings by having each character in the string be a string of length 1.

PEP 8 recommends using either single quotes or double quotes consistently. [3]



STRINGS

Example - String Slicing

```
s = 'poodles are cool'
print(s[0:2])   # prints the first 2 letters - "po"
print(s[:2])    # prints the first 2 letters - "po"
print(s[4:6])   # prints the 5th and 6th letters - "le"
print(s[1:])    # prints all but the first letter - "oodles are cool"
print(s[0:len(s)-1])    # prints all but the last letter - "poodles are coo"
print(s[-3:])   # prints last 3 letters - "ool"
```

STRINGS

Example - String Manipulation

```
pi = 3.14   # could instead use math.pi
s = '    You are my {} in the sky'
s = s.format(pi)
print(s)        # prints '    You are my 3.14 in the sky'

s = s.strip()
print(s)    # prints 'You are my 3.14 in the sky'

s = 'Muy mal'
print(s.replace('u','').replace('l',''))    # prints 'My ma'
print(s.split())    # prints as a list of words
```

STRINGS

Example - Word Count

```
# Count number of times a word appears in a string
def count_times_word_in_string(word, sentence):
        word_count = 0
        sentence_no_punctuation = sentence.strip(',.!?:;()[]{}')
        for current_word in sentence_no_punctuation.split():
                if current_word == word:
                        word_count += 1
        return word_count

print(count_times_word_in_string('car', 'Behold, a car and another car!'))
```

Output:

2

DICTIONARIES

In Python 3.7 and up a dictionary is a collection of data that is ordered, changeable, and can’t contain duplicates.

Dictionaries are a lot like a simple in-memory database with key-value pairs.

Since dictionaries are hashmaps, the time complexity of the average lookup time is O(1).

DICTIONARIES

	Dictionary	Tuple	List	Set
Ordered	✓	✓	✓	✗
Changeable	✓	✗	✓	✗ … but can add / delete
Duplicates Allowed	✗	✓	✓	✗
Indexable	✓	✓	✓	✗
	{}	()	[]	{}

DICTIONARIES

Example - Dictionary of Cell Phone Numbers and Their Owners

```
cell_and_name = {}  # dictionary for cell numbers and their owners

cell_and_name[17773334444] = 'Bubba Cleetus'    # added to dictionary
cell_and_name[19991234567] = 'Norma Jean'       # added to dictionary

for cell, name in cell_and_name.items():
    print(f'Cell # {cell} belongs to {name}')   # shows all entries

del cell_and_name[19991234567]  # deleted Norma Jean from dictionary
```

Output:

```
Cell # 17773334444 belongs to Bubba Cleetus
Cell # 19991234567 belongs to Norma Jean
```

CLASSES AND OBJECTS

OBJECT-ORIENTED PROGRAMMING (OOP)

Class

- CapWords naming convention [4]
- `__init__` constructor
- If class definition is empy, use `pass`

Object

- snake_case naming convention [4]
- `self` refers to current object
- `del` to delete object

CLASSES AND OBJECTS

Example - Class That Specifies Different Types of Footballs

```
class Football:
        def __init__(self, size, color):
                self.size = size
                self.color = color

        def describe_football(self):
                print(f'Your {self.size} football is {self.color}')

my_football = Football('medium', 'blue')
my_football.describe_football()
```

Output:

```
Your medium football is blue
```

CLASSES AND OBJECTS

Example - In-Memory Database of Restaurants

```
class Restaurant:
    def __init__(self, name, address, phone):
        self.name = name
        self. address = address
        self.phone = phone

    def show_info(self):
        print(f'{self.name} is in {self.address}.  Phone: {self.phone}')

identifier_and_info = {}    # dictionary to store list of restaurants
taco_bell = Restaurant('Taco Bell', 'Chicago, IL', 13337774444)
identifier_and_info[hash(taco_bell)] = taco_bell    # hash for unique ID
for identifier, info in identifier_and_info.items():
    info.show_info()
```

Output:

```
Taco Bell is in Chicago, IL.  Phone: 13337774444
```

REFERENCES

[1] https://www.turing.com/blog/in-demand-programming-languages-to-learn/

[2] https://www.w3schools.com/python/python_datatypes.asp

[3]. https://peps.python.org/pep-0008/#string-quotes

[4] https://peps.python.org/pep-0008/#naming-conventions

THANK YOU

Easy, Fun AI Training - https://aieasyfun.com