Python Tutorial Notes

Python Tutorial

(1:47) Installing Python & PyCharm

IDE: Integrated development enviorment

(6:42) Setup & Hello World

PyCharm->Configure->Preferences->Theme: Darcula

print("Hello World")

(10:25) Drawing a Shape

print(" /|")  
print(" / |")  
print(" / |")  
print("/\_\_\_|")

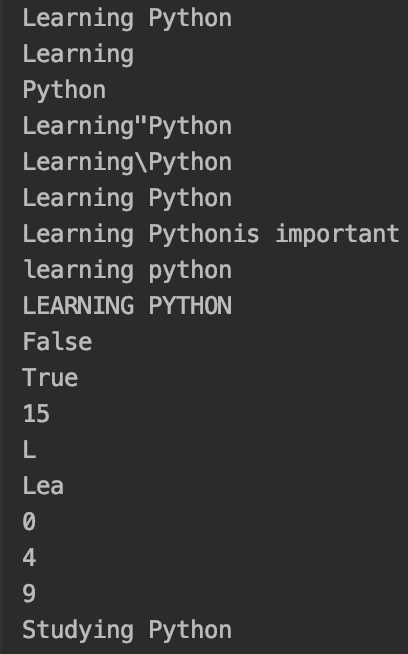
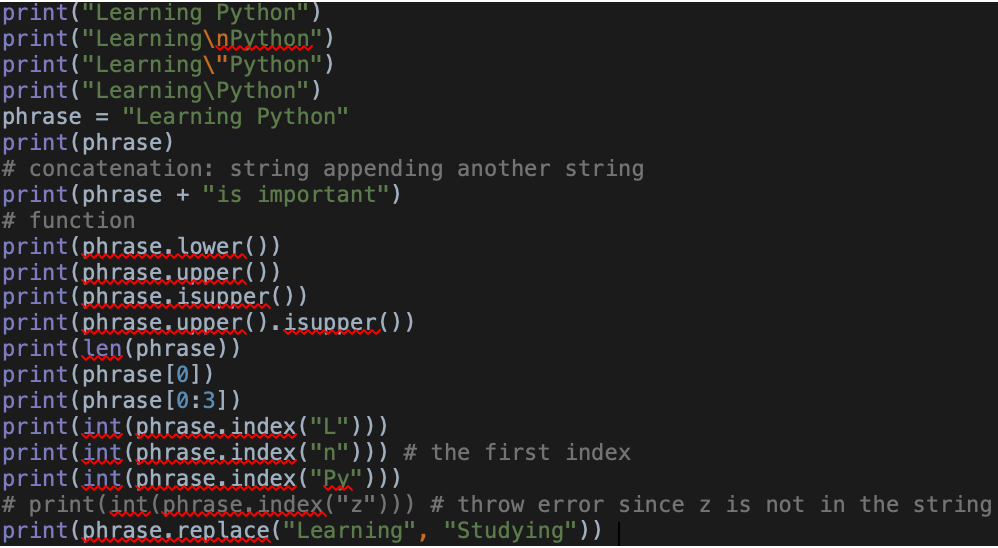
/Users/KevinMa/Desktop/Screen Shot 2020-01-19 at 8.36.48 PM.png play button to rerun the code

The order of the instructions(code) matters a lot.

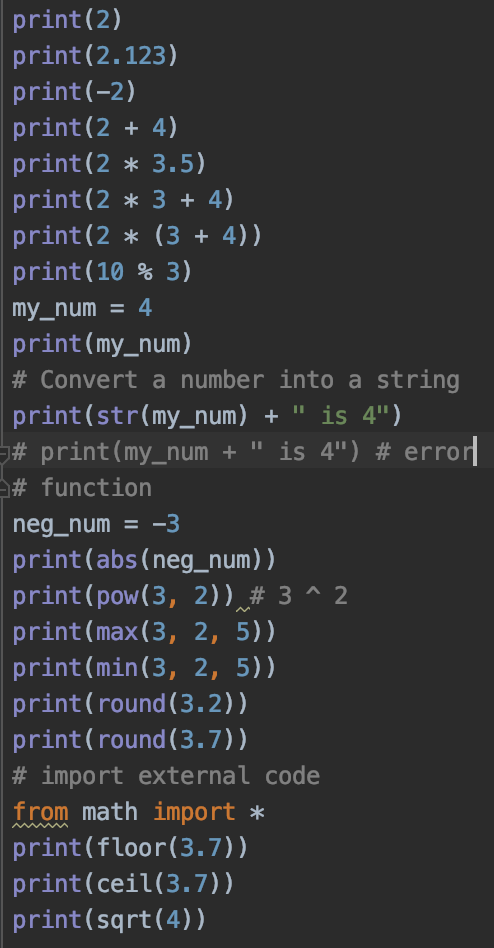
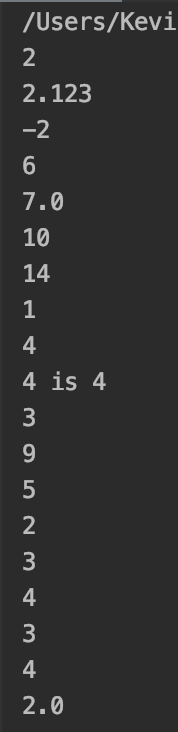
(15:08) Variables & Data Types

character\_name = "Tom"  
character\_age = 50  
is\_male = True

(27.04) Working with Strings



(38:20) Working with Numbers

(48:26) Getting Input from Users

name = input("Enter your name: ")  
age = input("Enter your age: ")  
print("Hello " + name + "! You are " + age)

Result:

Enter your name: Meilin

Enter your age: 18

Hello Meilin! You are 18

(52:38) Building a Basic Calculator

num1 = input("Enter a number: ")  
num2 = input("Enter another number: ")  
result = num1 + num2  
print(result)  
# By default, python convert the input number into a string  
# Convert string into number  
# result = int(num1) + int(num2)

# when input decimal number, use int() be error, so use float()  
result = float(num1) + float(num2)  
print(result)

Result:

Enter a number: 1.2

Enter another number: 33

1.233

34.2

(58.26) Mad Libs Game

# print("Roses are red")  
# print("Violets are blue")  
# print("I love you")  
color = input("Enter a color: ")  
plural\_noun = input("Enter a plural noun: ")  
celebrity = input("Enter a celebrity: ")  
  
print("Roses are " + color)  
print(plural\_noun + " are blue")  
print("I love " + celebrity)

Result:

Enter a color: pink

Enter a plural noun: shoes

Enter a celebrity: him

Roses are pink

shoes are blue

I love him

(1:03:10) Lists

my\_list = ["String", 2, 'v', True]  
print(my\_list)  
  
friends = ["Elle", "Emily", "Eileen"]  
print(friends)  
print(friends[2])  
print(friends[-1])  
print(friends[-2])  
print(friends[0:2])  
print(friends[0:3])  
print(friends[1:])  
friends[1] = "Ami"  
print(friends[1])

Result:

['String', 2, 'v', True]

['Elle', 'Emily', 'Eileen']

Eileen

Eileen

Emily

['Elle', 'Emily']

['Elle', 'Emily', 'Eileen']

['Emily', 'Eileen']

Ami

(1:10:44) List Functions

numbers = [10, 3, 5, 5]  
friends = ["Ami", "Bella", "Clair", "Divina", "Eileen"]  
print(friends)  
friends.extend(numbers)  
print(friends)  
friends.append("Fenty")  
print(friends)  
friends.insert(1, "Gina")  
print(friends)  
friends.remove(3)  
print(friends)  
friends.pop()  
print(friends)  
print(friends.index("Ami"))  
print(friends.count(5))  
numbers.sort()  
print(numbers)  
numbers.reverse()  
print(numbers)  
# friends.sort() # error, int vs. string  
# print(friends)  
friends2 = friends.copy()  
print(friends2)  
friends.clear()  
print(friends)

Result:

['Ami', 'Bella', 'Clair', 'Divina', 'Eileen']

['Ami', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 3, 5, 5]

['Ami', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 3, 5, 5, 'Fenty']

['Ami', 'Gina', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 3, 5, 5, 'Fenty']

['Ami', 'Gina', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 5, 5, 'Fenty']

['Ami', 'Gina', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 5, 5]

0

2

[3, 5, 5, 10]

[10, 5, 5, 3]

['Ami', 'Gina', 'Bella', 'Clair', 'Divina', 'Eileen', 10, 5, 5]

[]

(1:18:57) Tuples

Tuples is immutable, so a tuple cannot be changed and modified, add or remove elements.

coordinates = (1, 2)  
print(coordinates)  
print(coordinates[0])  
# coordinates[0] = 10 # error, tuple cannot be changed

Result:

(1, 2)

1

(1:24:15) Functions

indent sensitive

def say\_hi():  
 # intend sensitive  
 print("Hi, there!")  
print("Top")  
say\_hi()  
print("Bottom")

Result:

Top

Hi, there!

Bottom

def say\_bye(name):  
 print("Bye, " + name)  
say\_bye("Tom")  
say\_bye("Jerry")

Result:

Bye, Tom

Bye, Jerry

def profile(name, age):  
 print("Name is " + name + ", age is " + str(age))  
profile("Tom", 12)

Result:

Name is Tom, age is 12

(1:34:10) Return Statement

Print cube of a number:

def cube1(num):  
 num \* num \* num

print(cube1(3))

Result:

None

Return statement: get information back from a function

def cube1(num):  
 return num \* num \* num

print(cube1(3))

Result:

27

def cube2(num):  
 return num \* num \* num

result = cube2(4)  
print(result)

Result:

64

(1:40:06) If Statements

is\_male = True  
is\_tall = True  
  
if is\_male:  
 print("You are a male")  
else:  
 print("You are NOT a male")  
  
if is\_male or is\_tall:  
 print("You are a male or tall or both")  
else:  
 print("You neither male nor tall")  
  
if is\_male and is\_tall:  
 print("You are a tall male")  
else:  
 print("You either not male or not tall or both")  
  
if is\_male and is\_tall:  
 print("You are a tall male")  
elif is\_male and not (is\_tall):  
 print("You are a short male")  
elif not(is\_male) and is\_tall:  
 print("You are NOT a male but are tall")  
else:  
 print("You are NOT a male and NOT tall ")

Result:

You are a male

You are a male or tall or both

You are a tall male

You are a tall male

(1:54:07) If Statements & Comparisons

def max\_num(num1, num2, num3):  
 if num1 >= num2 and num1 >= num3:  
 return num1  
 elif num2 >= num1 and num2 >= num3:  
 return num2  
 else:  
 return num3  
  
print(max\_num(4, 2, 9))

Return:

9

(2:00:38) Building a Better Calculator

num1 = float(input("Enter first number: "))  
operator = input("Enter operator: ")  
num2 = float(input("Enter second number: "))  
  
if operator == "+":  
 print(num1 + num2)  
elif operator == "-":  
 print(num1 - num2)  
elif operator == "\*":  
 print(num1 \* num2)  
elif operator == "/":  
 print(num1 / num2)  
else:  
 print("Invalid operator")

Result:

Enter first number: 5

Enter operator: ()

Enter second number: 4

Invalid operator

(2:07:16) Dictionaries

Key has to be unique.

dateConversions = {  
 "Mon": "Monday",  
 "Tue": "Tuesday",  
 "Wed": "Wednesday",  
 "Thu": "Thursday",  
 "Fri": "Friday",  
 "Sat": "Saturday",  
 "Sun": "Sunday"  
}  
print(dateConversions["Sat"])  
print(dateConversions["Mon"])  
print(dateConversions.get("Fri"))  
# default value  
print(dateConversions.get("Abd", "Not a valid key"))

Result:

Saturday

Monday

Friday

Not a valid key

(2:14:14) While Loop

i = 1  
while i <= 5:  
 print(i)  
 i += 1  
 # i++ # error

print("Loop End!")

Result:

1

2

3

4

5

Loop End!

(2:20:02) Building a Guessing Game

secret\_word = "morning"  
guess = ""  
guess\_count = 0  
guess\_limit = 3  
out\_of\_guesses = False  
while guess != secret\_word and not(out\_of\_guesses):  
 if guess\_count < guess\_limit:  
 guess = input("Enter guess: ")  
 guess\_count += 1  
 else:  
 out\_of\_guesses = True;  
  
if out\_of\_guesses:  
 print("Out of guesses! You lose!")  
else:  
 print("You win!")

Result:

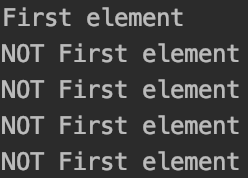
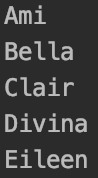
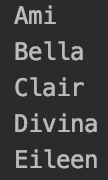
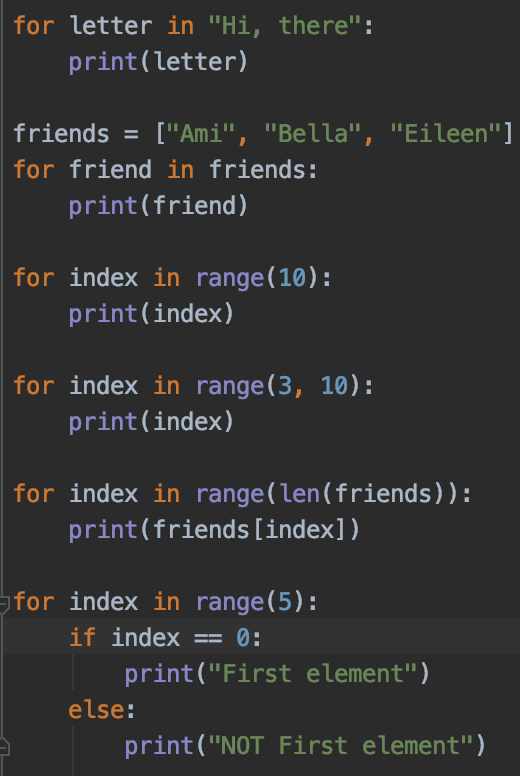
Enter guess: good

Enter guess: night

Enter guess: well

Out of guesses! You lose!

(2:32:44) For Loop



(2:41:20) Exponent Functions

print(2 \*\* 3) # 2 ^ 3  
  
def raise\_to\_power(base\_num, pow\_num):  
 result = 1  
 for index in range(pow\_num):  
 result \*= base\_num  
 return result  
print(raise\_to\_power(2, 4))

Result:

8

16

(2:47:12) 2D Lists & Nested Loops

number\_grid = [  
 [1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9],  
 [0]  
]  
print(number\_grid[1][1])  
  
for row in number\_grid:  
 for col in row:  
 print(col)

Result:

5

1

2

3

4

5

6

7

8

9

0

(2:52:40) Build a Translator

vowels -> g

dog -> dgg

cat -> cgt

def translate(phrase):  
 translation = ""  
 for letter in phrase:  
 if letter in "AEIOUaeiou":  
 translation = translation + "g"  
 else:  
 translation = translation + letter  
 return translation  
  
print(translate(input("Enter a phrase: ")))

Result:

Enter a phrase: apple

gpplg

def translate(phrase):  
 translation = ""  
 for letter in phrase:  
  
 # if letter in "AEIOUaeiou":  
  
 if letter.lower() in "aeiou":  
 if letter.isupper():  
 translation = translation + "G"  
 else:  
 translation = translation + "g"  
 else:  
 translation = translation + letter  
 return translation  
  
print(translate(input("Enter a phrase: ")))

Result:

Enter a phrase: On

Gn

(3:00:18) Comments

# Comments  
'''  
multiple  
line  
comments  
'''  
print("Comments are fun")

(3:04:18) Try Except

try:  
 value = 10 / 0 # ZeroDivisionError  
 number = int(input("Enter a number: "))  
 print(number)  
except ZeroDivisionError:  
 print("Divided by zero")  
except ValueError:  
 print("Invalid Input")

Result:

Divided by zero

As a variable

except ZeroDivisionError as err:  
 print(err)

(3:12:40) Reading Files

open("employees.txt", "r")  
open("employees.txt", "w")  
open("employees.txt", "a")  
open("employees.txt", "r+")

# r -> read only  
# w -> write -> add / change info in the file  
# a -> append info at the end of the file, but not change the existing info  
# r+ -> read and write  
# open file, must close file

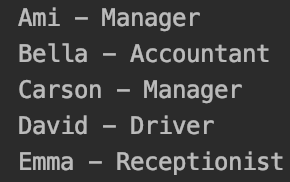
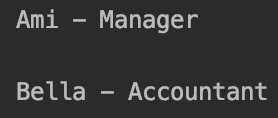
employee\_file = open("employees.txt", "r")  
  
print(employee\_file.readable()) # True  
print(employee\_file.read()) # All info from the file  
print(employee\_file.readline()) # The first line in the file  
print(employee\_file.readline()) # The second line in the file  
print(employee\_file.readlines()) # Print lines as an array

print(employee\_file.readlines()[2]) # Print line with index

for employee in employee\_file.readlines(): # Print each line  
 print(employee)

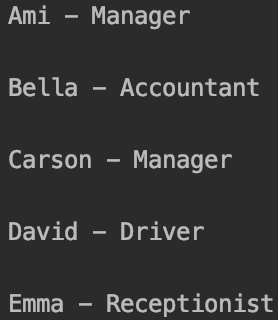
employee\_file.close()

Result:





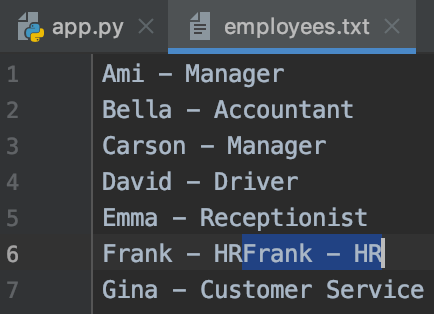


(3:21:26) Writing to Files

employee\_file = open("employees.txt", "a")  
employee\_file.write("Frank - HR")  
employee\_file.write("\nGina - Customer Service")  
employee\_file.close()

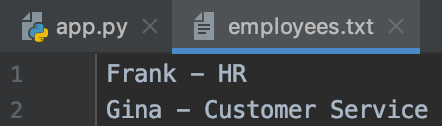
appending may cause trouble, no space, no new line,

\n -> new line



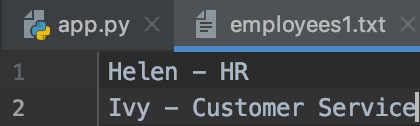
employee\_file = open("employees.txt", "w")

with "w", overwrite the entire file.

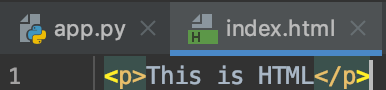


User open, "w" to create a new file

employee\_file = open("employees1.txt", "w")  
employee\_file.write("Frank - HR")  
employee\_file.write("\nGina - Customer Service")  
employee\_file.close()



employee\_file = open("index.html", "w")  
employee\_file.write("<p>This is HTML</p>")  
employee\_file.close()



(3:28:14) Modules and Pip

<https://docs.python.org/3/py-modindex.html>

Third party modules

<https://python-docx.readthedocs.io/en/latest/>

<https://python-docx.readthedocs.io/en/latest/user/install.html>

pip install python-docx

pip is used to install python modules.

import useful\_tools  
print(useful\_tools.roll\_dice(10))  
  
import docx

(3:43:55) Classes & Objects

Student.py

class Student:  
  
 def \_\_init\_\_(self, name, major, gpa, is\_male):  
 self.name = name  
 self.major = major  
 self.gpa = gpa  
 self.is\_male = is\_male

app.py

from Student import Student

# from Student file import Student class  
  
student1 = Student("Ami", "Business", 3.2, False)  
student2 = Student("Bill", "Computer", 3.9, True)  
  
print(student1)  
print(student1.major)  
print(student2.name)

Result:

Business

Bill

(3:57:37) Building a Multiple Choice Quiz

Question.py

class Question:  
  
 def \_\_init\_\_(self, prompt, answer):  
 self.prompt = prompt  
 self.answer = answer

app.py

from Question import Question  
question\_prompts = [  
 "What color are apples?\n(a) Red/Green\n(b) Purple\n(c) Orange\n\n",  
 "What color are bananas?\n(a) Teal\n(b) Magenta\n(c) Yellow\n\n",  
 "What color are strawberries?\n(a) Yellow\n(b) Red\n(c) Blue\n\n"  
]  
  
questions = [  
 Question(question\_prompts[0], "a"),  
 Question(question\_prompts[1], "c"),  
 Question(question\_prompts[2], "b"),  
]  
  
def run\_test(questions):  
 score = 0  
 for question in questions:  
 answer = input(question.prompt)  
 if answer == question.answer:  
 score += 1  
 print("You got " + str(score) + "/" + str(len(questions)) + " correct")  
  
run\_test(questions)

(4:08:29) Object Functions

Student.py

class Student:  
  
 def \_\_init\_\_(self, name, major, gpa, is\_male):  
 self.name = name  
 self.major = major  
 self.gpa = gpa  
 self.is\_male = is\_male  
  
 def on\_honor\_roll(self):  
 if self.gpa >= 3.5:  
 return True  
 else:  
 return False

app.py

from Student import Student  
  
student1 = Student("Ami", "Business", 3.2, False)  
student2 = Student("Bill", "Computer", 3.9, True)  
  
print(student1.on\_honor\_roll())  
print(student2.on\_honor\_roll())

Result:

False

True

(4:12:36) Inheritance

Farm.py

class Farm:  
 def plant\_apple(self):  
 print("The farm plants apple")  
 def plant\_banana(self):  
 print("The farm plants banana")  
 def plant\_grape(self):  
 print("The farm plants grape")

TropicalFarm.py

from Farm import Farm  
  
class TropicalFarm(Farm):  
   
 def plant\_grape(self):  
 print("The farm plants tropical grape")  
  
 def plant\_mango(self):  
 print("The farm plants tropical mango")

app.py

from Farm import Farm  
from TropicalFarm import TropicalFarm  
  
myFarm = Farm()  
myFarm.plant\_apple()  
myFarm.plant\_grape()  
  
myTropicalFarm = TropicalFarm()  
myTropicalFarm.plant\_grape()  
myTropicalFarm.plant\_mango()

Result:

The farm plants apple

The farm plants grape

The farm plants tropical grape

The farm plants tropical mango

(4:20:44) Python Interpreter

terminal

<https://www.youtube.com/watch?v=rfscVS0vtbw>