3DC

DIGITAL DESIGN & DEVELOPMENT CLUB

PYTHON WORKSHOP 3

DICTIONARIES & FUNCTIONS

RECAP

1. DATA TYPES
2. OPERATORS
3. COMPARATORS
4. FOR LOOPS
5. WHILE LOOPS

CONTENT

1. RECAP (20 minutes)
2. Dictionaries (40 minutes)
3. Break (10 minutes)
4. Functions (50 minutes)

DICTIONARIES (CODE TOGETHER)

1. Consists of **unordered** Key-Value pairs
2. Word –> definition
3. The **key** must be **immutable**
4. Example: Student as a dictionary
    1. Creating a dictionary
    2. Assigning a key to a value
    3. Re-assigning
    4. Deleting
    5. Getting all the keys and getting all the values
    6. Looping through a dictionary
    7. Getting the length of a dictioanry

KEY		VALUE
”BALL”	→	”A round thing”
“SUTD”	→	“An awesome school”
“3DC”	→	“An awesome 5th row”

EXERCISE 1

- Define a dictionary called **released**
- **released = { iPhone = }** released = { "iphone" : 2007, "iphone 3G" : 2008, "iphone 3GS" : 2009, "iphone 4" : 2010, "iphone 4S" : 2011, "iphone 5" : 2012 }

PRINT IT OUT!

1. Which year was iPhone 4 released?
2. How many iPhone models are there from 2007 to 2011?
3. Since iPhone 3G, how long did it take for Apple to release iPhone 4S?

- iPhone 5 was released in 2012, add that in to the dict.
- Remove iPhone 3G from the dict.
- First iPhone's actual release date was 2007, change it

iPhone Model	Year
iPhone	2006
iPhone 3G	2008
iPhone 3GS	2009
iPhone 4	2010
iPhone 4S	2011

EXERCISE 2

- You are interested in the price of computers. After doing some research, you decided to compile the prices of different brands in a dictionary.
- Computers = {}

Computer	Price ($)
Macbook Air	1300
Aspire S3	900
Zenbook	1050
Ideapad	1500
Folio 13	1100

1. Print the price of an ideapad
2. Reassign the price of Folio 13 to 1200
3. You have a budget of $1200. Loop through the dictionary and find print out the models of computers you can afford.
4. **Challenge: Find out the total price of all the computers**

INTRO TO FUNCTION

- Functions are building blocks of complicated programmes.
- Higher degree of reusing code.
- It allows you to perform abstraction.
- Enhances modularity.

GREETINGS

- Print the following for 'Jane", "Marry" and 'Jones"
- Hi <name>
- "Good morning"
- "How have you been"
- "It is a pleasure to meet you"

SOME CONVERSION FUNCTIONS

Code Together

- 1 cm = 0.393701 inches
- Programme a cm_to_inches function
- Programme an inches_to_cm function
- Returning (Void vs Fruitful functions)

Now you try

- 1 kg = 2.20462 pounds
- To convert temperatures in degrees **Celsius to Fahrenheit**, multiply by 1.8 (or 9/5) and add 32.

GREETINGS WITH PARAMETERS

Rewrite the greeting function to accept 2 parameters.

1. A string variable – name.
2. An integer in 0000 hours format. Your greeting should vary from Good Morning (600 to 1159) / Good afternoon (1200 – 1759) / Good Night (1800 to 0559)
    1. You may write 0000 to 0959 as 0 and 959 respectively.

MORE ON VOID VS FRUITFUL FUNCTIONS

- Code Together
- Sort vs Sorted example [1,4,5,2,3,6].sort() vs sorted(1,4,5,2,3,6])

- Now you try
- Create a function that takes in a list of integers and multiply them by 2.
    1. Modify the list, do not return the list
    2. Return, but do not modify the list

CHALLENGE

- Supposed you are a 7-up ice breaker enthusiast. You would like to memorise all the 7 ups below a certain number so that you can "train" for the game.
- Create a function that takes in **an integer** and prints all the numbers from 1 to the **integer (inclusive). For multiples of 7 and numbers containing 7, print "Seven up!" instead.**
- Seven_up(10)
  - 1,2,3,4,5,6,**"Seven up!"** ,8,9,10
- Seven_up(20)
  - 1,2,3,4,5,6,"**Seven up!",** 8, 9, 10, 11, 12, 13, ,"**Seven up!",** , 15 16, ,"**Seven up!",** , 18, 19, 20

MORE ADVANCED CONCEPTS

- Returning booleans
- Key-word arguments
- Default parameters
- Recursive function