**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **20/05/2020** | **Name:** | **Shilpa S** |
| **Course:** | **Python** | **USN:** | **4AL14EC078** |
| **Topic:** | **Application 1: Build an interactive English dictionary** | **Semester & Section:** | **8th - A** |
| **GitHub Repository:** | **Shilpa-online** |  |  |

**Afternoon Course Details**

**Image of session**

**Report –**

* A list comprehension is an expression that creates a list by iterating over another container.
* A basic list comprehension:
  1. [i\*2 for i in [1, 5, 10]]

Output: [2, 10, 20]

* List comprehension with if condition:
  1. [i\*2 for i in [1, -2, 10] if i>0]

Output: [2, 20]

* List comprehension with an if and else condition:
  1. [i\*2 if i>0 else 0 for i in [1, -2, 10]]

Output: [2, 0, 2

If –Else condition comprehension

obj = ["Even" if i%2==0 else "Odd" for i in range(10)]

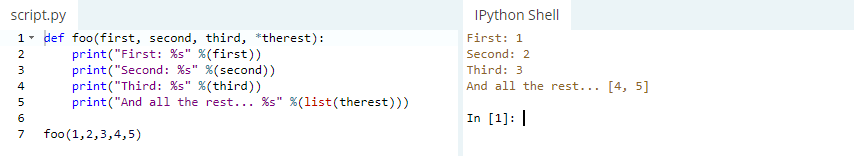
print (obj)

**Output -**['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']

['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']

**More on Functions**

Multiple Function Arguments –



In this section I learned that:

* Functions can have more than one parameter:

1. def volume(a, b, c):
2. return a \* b \* c

* Functions can have default parameters (e.g. coefficient):

1. def converter(feet, coefficient = 3.2808):
2. meters = feet / coefficient
3. return meters
5. print(converter(10))

Output: 3.0480370641306997

Arguments can be passed as non-keyword (positional) arguments (e.g. a) or keyword arguments (e.g. b=2 and c=10):

1. def volume(a, b, c):
2. return a \* b \* c
4. print(volume(1, b=2, c=10))

* An \*args parameter allows the  function to be called with an arbitrary number of non-keyword arguments:

1. def find\_max(\*args):
2. return max(args)
3. print(find\_max(3, 99, 1001, 2, 8))

Output: 1001

* An \*\*kwargs parameter allows the function to be called with an arbitrary number of keyword arguments:

1. def find\_winner(\*\*kwargs):
2. return max(kwargs, key = kwargs.get)
4. print(find\_winner(Andy = 17, Marry = 19, Sim = 45, Kae = 34))

Output: Sim

**File Processing**

In this section I learned that:

* You can **read** an existing file with Python:

1. with open("file.txt")as file:
2. content =file.read()

* You can **create** a new file with Python and **write** some text on it:

1. with open("file.txt","w")as file:
2. content =file.write("Sample text")

* You can **append** text to an existing file without overwriting it:

1. with open("file.txt","a")as file:
2. content =file.write("More sample text")

* You can both **append and read** a file with:

1. with open("file.txt","a+")as file:
2. content =file.write("Even more sample text")
3. file.seek(0)
4. content =file.read()

**Imported Modules**

In this section you learned that:

* **Built-in objects** are all objects that are written inside the Python interpreter in C language.
* **Built-in modules** contain built-ins objects.
* Some built-in objects are not immediately available in the global namespace. They are parts of a built-in module. To use those objects the module needs to be **imported** first. E.g.:
  1. import time
  2. time.sleep(5)
* **A list of all built-in modules** can be printed out with:
  1. import sys
  2. sys.builtin\_module\_names
* **Standard libraries** is a jargon that includes both built-in modules written in C and also modules written in Python.
* **Standard libraries** written in Python reside in the Python installation directory as *.py* files. You can find their directory path with sys.prefix.
* **Packages** are a collection of *.py* modules.
* **Third-party libraries** are packages or modules written by third-party persons (not the Python core development team).
* Third-party libraries can be **installed** from the terminal/command line:

Windows:

pip install pandas or use python -m pip install pandas if that doesn't work.

* Mac and Linux: -pip3 install pandas or use python3 -m pip install pandas if that doesn't work.