**PYTHON EXERCISE**

**1.What is Python?**

**Python:** It is invented by Guido Van Rossum and after named as Monly Python.

* Python is an open-source programming language.
* It is relatively easy to learn.
* It is powerful tool with many modules, that can be imported in to extend its functionality.

**2. What is the purpose of PYTHONPATH, PYTHONSTARTUP, PYTHONCASEOK, PYTHONHOME environment variables?**

**Python Environment Variables:**

* **PYTHONPATH:**
* It has a role similar to PATH. This variable tells the python interpreter where to locate the module files imported into a program.
* It should include the python source library directory and the directories containing python source code.
* **PYTHONSTARTUP:**
* It contains the path of an initialization file containing python source code.
* It is executed every time you start the interpreter.
* It is named as **pythonrc.py** in unix and it contains commands that load and modify the **PYTHONPATH**.
* **PYTHONCASEOK:**
* It is used in windows to instruct python to find the first case insensitive match in an import statement.
* Set this variable to any value to activate it.
* **PYTHONHOME:**
* It is an alternative module search path.
* It is usually embedded in the **PYTHONSTARTUP** or **PYTHONPATH** directories to make switching module libraries easy.

**3. What are the supported data types in Python?**

**Supported Data types in python:**

Python has various standard data types that are used to define the operations possible on them and the storage method for each of them

They are five supported data types in python.

* Numbers
* String
* List
* Tuple
* Dictionary

**4. What is the difference between list and tuples?**

**Difference between list and tuple:**

|  |  |
| --- | --- |
| **LIST** | **TUPLE** |
| List are mutable | Tuples are immutable |
| Implication and iteration take time | Implication and iteration compare to list, it is fast |
| The list is better for performing operations, like append, delete etc | Tuple data type is appropriate for accessing the elements |
| Lists consume more memory | Tuples consume less memory |
| Lists have several built-in methods | Tuples does not have many built-in methods |
| The unexpected changes and errors are more likely to occur | In tuple, it is hard to take place |

**5. How is memory managed in Python?**

* Memory allocation can be defined as allocating a block of space in the computer memory to a program.
* Memory management in Python involves a private heap containing all Python objects and data structures.
* The management of this private heap is ensured internally by the Python memory manager.

**6. Explain Inheritance in Python with an example.**

**Inheritance:**

* It is most important aspect of object-oriented programming which simulates the real-world concepts of inheritance.
* It specifies that the child object acquires all the properties and behaviour of the parent.
* The class is derived class or child class and the one whose properties are acquired is known as "BASE CLASS" or "PARENT CLASS".
* It provides the reusability of the code.
* In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class.

**Example:**

class fruits:

def eat(self):

print("paul is eating apple")

# child class salad will inherit the base class fruits

class salad(fruits):

def chew(self):

print("john loves eating salad")

a=salad()

a.chew()

a.eat()

**Output:** john loves eating salad

Paul is eating apple

**7. Whenever Python exits, why isn’t all the memory de-allocated?**

* Using (del) keyword we can try to remove some particular object.
* In python, there is no guarantee that the object is actually removed from the memory when you use **del(object)**.

**8. What is dictionary in Python?**

**Dictionary:**

* Dictionaries are used to store the data values in **key:value** pairs.
* A dictionary is a collection which is ordered, changeable and do not allow duplicates.
* In dictionary the items can be referred by using the key name.
* In dictionary are there are changeable that can change or add or remove

the items after the dictionary is created.

* It cannot have two items in the same key.

**9. Write a one-liner that will count the number of capital letters in a file. Your code should work even if the file is too big to fit in memory.**

**10. Write a sorting algorithm for a numerical dataset in Python.**

**Code:**

numbers=[256,785,963,159,248,367,945]

numbers.sort()

print(numbers)

**Output:** [159, 248, 256, 367, 785, 945, 963]

**11. How will you reverse a list?**

**Code:**

**veggies=["carrot","potato","beetroot","tomato"]**

**veggies.reverse()**

**print(veggies)**

**Output:** ['tomato', 'beetroot', 'potato', 'carrot']

**12. How will you remove last object from a list?**

**Code:**

fruits=["kiwi","strawberry","banana","orange"]

fruits.remove("orange")

print(fruits)

**Output:** ['kiwi', 'strawberry', 'banana']

**13. What are negative indexes and why are they used?**

**Negative indexes:**

* Use the negative index to start the slice from the end of the string.
* The index value of -1 gives the last element, and -2 gives the second last element of an array.
* The negative indexing starts from where the array ends.

**Example:**

**Code:**

str=["a","l","p","h","a","b","e","t","s"]

print(str[-3])

print(str[-8:-4])

print(str[-2:])

print(str[:-5])

**Output:** e

['l', 'p', 'h', 'a']

['t', 's']

['a', 'l', 'p', 'h']

**14. Explain split(), sub(), subn() methods of “re” module in Python.**

* The (re) module in python refers to the module Regular Expressions (RE).
* It specifies a set of strings or patterns that matches it.

**spilt( ):** It splits the string according to the occurrences of a character or a pattern. When it finds that pattern, it returns the remaining characters from the string as part of the resulting list.The split method should be imported before using it in the program.

**Syntax:** re.split (pattern, string, maxsplit=0, flags=0)

**sub():** This function stands for the substring in which a certain regular expression pattern is searched in the given string (3rd parameter). When it finds the substring, the pattern is replaced by repl (2nd parameter). The count checks and maintains the number of times this has occurred.

**Syntax:** re.sub (pattern, repl, string, count=0, flags=0)

**subn( ):** This function is similar to sub() in all ways except the way in which it provides the output. It returns a tuple with count of total of all the replacements as well as the new string.

**Syntax:** re.subn (pattern, repl, string, count=0, flags=0)

**15. What is the difference between range & xrange?**

**xrange( ):** It will have real integer objects.

* It returns a list of integers.
* Execution speed is slower.
* Takes more memory as it keeps the entire list of elements in memory.
* All arithmetic operations can be performed as it returns a list.

**range( ):** It has to reconstruct the integer object every time.

* Returns a generator object.
* Returns a generator object.
* Takes less memory as it keeps only one element at a time in memory.
* Operators cannot be performed in xrange( ).

**16. What is a Python module?**

* A Python module is a file containing Python definitions and statements.
* A module can define functions, classes, and variables.
* A module can also include runnable code.
* Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

**17. Name the File-related modules in Python?**

**Get Current Directory (** **getcwd( ) ) :**

* This method returns the current working directory in the form of a string.

**Changing Directory ( chdir( ) ) :**

* The new path that we want to change to must be supplied as a string to this method.
* We can use both forward slash (/) or the backward slash () to separate path elements.

**List Directories and Files:**

* All files and sub directories inside a directory can be known using the **listdir()** method.
* This method takes in a path and returns a list of sub directories and files in that path.
* If no path is specified, it returns from the current working directory.

**Making a New Directory (mkdir( ) ) :**

* This method takes in the path of the new directory.
* If the full path is not specified, the new directory is created in the current working directory.

**Renaming a Directory or a File ( rename( ) ) :**

* The first argument is the old name and the new name must be supplies as the second argument

**Removing a directory ( os.rmdir( ) ) :**

* It is used to remove/delete a directory.
* The parameter passed is the path to that directory.
* It deletes the directory if and only if it is empty, otherwise raises an OS-Error.

**18. Explain the use of with statement?**

* **“with”** statement in Python is used in exception handling to make the code cleaner and much more readable.
* It simplifies the management of common resources like file streams.
* It helps avoiding bugs and leaks by ensuring that a resource is properly released when the code using the resource is completely executed.
* It is popularly used with file streams, as shown above and with Locks, sockets, subprocesses and telnets etc.

**19. Explain all the file processing modes supported by Python?**

* **r** - open a file for reading.
* **w** - Open a file for writing. If file already exists its data will be cleared before opening. Otherwise, new file will be created.
* **x** - open for exclusive creation, failing if the file already exists.
* **a** - open for writing, appending to the end of the file if it exists.
* **b** - binary mode
* **t** - text mode
* **+r** - Open a file for updating (reading and writing)
* **r+:**  To read and write data into the file. The previous data in the file will not be deleted.
* **w+:** To write and read data. It will override existing data.
* **a+:** To append and read data from the file. It won’t override existing data.

**20. How many kinds of sequences are supported by Python? What are they?**

Python supports six different types of sequences.

* Strings
* Lists
* Tuples
* Byte sequences
* Byte arrays
* Range objects.

**21. How do you perform pattern matching in Python? Explain**

* It is process of checking a specific sequence character, tokens, data exists among the given data.
* Regular programming languages make use of regular expressions **(**[**regex**](https://www.educative.io/edpresso/how-to-use-regex-in-python)**)** for pattern matching.
* Pattern matching is used to determine whether source files of high-level languages are syntactically correct.
* It is also used to find and replace a matching pattern in a text or code with another text or code.
* Any application that supports search functionality uses pattern matching in one way or another.

**Code:** import re

a="guru99,education is fun"

b= re.findall(r"^\w+",a)

print(b)

**Output:** 'guru99'

**22. How to display the contents of text file in reverse order?**

**23. Which of the following is an invalid statement?**

**a) abc = 1,000,000**

**b) a b c = 1000 2000 3000**

**c) a,b,c = 1000, 2000, 3000**

**d) a\_b\_c = 1,000,000**

**Answer: b**

**Code:** abc = 1,000,000

#a b c = 1000 2000 3000

a,b,c = 1000, 2000, 3000

a\_b\_c = 1,000,000

print(abc)

#print(a b c)

print(a,b,c)

print(a\_b\_c)

**Output:** (1,0,0)

(1000,2000,3000)

(1,0,0)

**24. What is the output of the following?**

**try: if '1' != 1:**

**raise**

**a) some Error has occured**

**b) some Error has not occured**

**c) invalid code**

**d) none of the above**

**Answer: c**

**25. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?**

**Answer: 25**

**Code: list1=[2,33,222,14,25]**

**print(list1[-1])**

**Output: 25**

**26. To open a file c:\scores.txt for writing?**

**Code:** fileptr=open("scores.txt","r")

if fileptr:

print("file is opened successfully")

**Output:** file is opened successfully

**27. Is Python object oriented? what is object oriented programming?**

* Yes, python is object-oriented programming language.
* Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code.
* Data in the form of fields and code in the form of procedures.

**28. What is multithreading? Give an example.**

* A thread is a unit of execution within a process.
* **Multithreading** refers to concurrently executing multiple threads by rapidly switching the control of the CPU between threads called **context switching**.

**Code:**

**# importing the threading module**

import threading

**# importing the time module**

import time

**# Function to print "Hello", however, the function sleeps**

**# for 2 seconds at the 11th iteration**

def print\_hello():

for i in range(5):

if i == 10:

time.sleep(2)

print("Hello")

**# Function to print numbers till a given number**

def print\_numbers(num):

for i in range(num+1):

print(i)

**# Creating the threads. Target is set to the name of the**

**# function that neeeds to be executed inside the thread and**

**# args are the arguments to be supplied to the function that**

**# needs to be executed.**

print("Greetings from the main thread.")

thread1 = threading.Thread(target = print\_hello, args = ())

thread2 = threading.Thread(target = print\_numbers, args = (5,))

**# Starting the two threads**

thread1.start()

thread2.start()

print("It's the main thread again!")

**Output:**

Greetings from the main thread.

Hello

Hello

Hello

Hello

Hello

0

1

2

3

4

5

It's the main thread again!

**29. Does Python supports interfaces like in Java? Discuss.**

* No, python does not have any equivalent of interfaces.
* An interface in python is defined using python class and is a subclass of interface.
* Interface which is the parent interface for all interfaces.
* The implementations will be done by the classes which will inherit the interface.
* Interfaces in Python are a little different from other languages like Java or C# or C++