**IHUOMA FAVOUR AGBARU**

**THEORY ASSESSMENT QUESTIONS**

1. What is Python and what are its main features?

**Answer:** Python is a high level , object oriented, interactive and script programming language. It’s main features include:

* Easy to read, python codes are clearer and more defined.
* Easy to learn, python’s syntax is easy to understand and learn.
* Python has a broad standard library across platforms.
* Python provides interfaces to commercial databases.
* Python programs are scalable.
* Python has support for interactive mode, where the program can be tested.

1. Discuss the difference between Python 2 and Python 3.

**Answer:**

|  |  |  |
| --- | --- | --- |
| **S/N** | **PYTHON 3** | **PYTHON 2** |
| 1 | Easy to learn | Difficult to learn |
| 2 | Widely accepted by python developers | Widely accepted by DevOps . |
| 3 | Value of variable never changes. | Value of global variable changes when used inside a for loop. |
| 4 | Range () functions is used to perform iterations. | Xrange () functions is used to perform iterations. |
| 5 | Exceptions are enclosed in parenthesis | Exceptions are enclosed in notations. |
| 6 | Python 3 is not backward compatible to python 2. | Python 2 can be ported to python 3 with a lot of efforts. |
| 7 | Strings are stored as ASCII by default. | Strings are stored as UNICODE. |

1. What is PEP 8?.

**Answer:** PEP 8 is a document that provides guidelines on how a python programmer should write his codes.

1. In computing / computer science what is a program?

**Answer:** A program is a set of instruction (which may be a single line code or a million-line codes) written in any programming language to be executed by the computer.

1. In computing / computer science what is a process?

**Answer:** This is a program in execution.

6. In computing / computer science what is cache?

**Answer:** Cache memory is a high -speed memory used to reduce the average time to access data from the Main memory.

7. In computing / computer science what is a thread and what do we mean by

multithreading?

**Answer:** A thread is flow of execution through the lightweight process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

Multithreading is when two programs run con currently and execute without any issues on the same system.

8. In computing / computer science what is concurrency and parallelism and what

are the differences?

**Answer:** Concurrency is when more than one task overlaps each other in execution. Parallelism is splitting of tasks into subtasks so they can be processed simultaneously.

**DIFFERENCES BETWEEN CONCURRENCY AND PARALLELISM.**

|  |  |  |
| --- | --- | --- |
| **S/N** | **CONCURRENCY** | **PARALLELISM** |
| 1 | This is done using a single processing unit. | This cannot be done by using a single processing unit. |
| 2 | Debugging is hard in Concurrency | Debugging is hard yet simpler than concurrency. |
| 3 | It is a non-deterministic control flow approach | This is a deterministic control flow approach. |
| 4 | This increases the amount of work finished time. | Parallelism increases the throughput and computational speed of the system. |

9. What is GIL in Python and how does it work?

**Answer:** Global Interpreter Lock (GIL) is a type of process lock which is used by python whenever it deals with processes. GIL is simple and it provides a performance increase to single-threaded programs as only one lock needs to be managed.

10. What do these software development principles mean: DRY, KISS, BDUF

**Answer: DRY**  this means Don’t Repeat Yourself: Code reuse is always better than code duplications, be pragmatic with your codes as not to duplicate same codes that are already written. **KISS**  this means Keep it simple, stupid Writing a complex code while a simple efficient code is what is needed for the problem is really not needed. Sometimes it also isn’t about the new technology but the right ones. **BDUF**  this means big design upfront, take out time to think out all processes involved before implementation. This makes the job a little less tasking as the thinking process would have made.

11. What is a Garbage Collector in Python and how does it work?

A garbage collector deletes or removes variables that have referenced to zero and isn’t used anymore. The way garbage collector works is by first importing it with the syntax *import gc, when gc.collect()* is called it can detect the circular reference, destroy the objects, and reclaim the memory.

12. How is memory managed in Python?

**Answer:** Memory is managed in python with the python memory manager that handles the components which deal with various dynamic storage management aspects, like sharing, segmentation, pre-allocation or caching.

13. What is a Python module?

**Answer:** A python module is a file consisting of Python code, it defines functions, classes and variables.

14. What is docstring in Python?

**Answer:** Docstring provides a convenient way of associating documentation with Python modules, functions, classes, and methods, it describes what an exact function does and how. Docstring is written with single triple quotes’’’ or double triple quotes “”” and ends with a full stop.

15. What is pickling and unpickling in Python? Example usage.

**Answer:** picking is the conversion of objects into byte streams and textual representation. Unpickling is the deconversion of data to an object.

**Example.** *'''  
This is to pickle a file. it created a school.txt file that cannot be read.  
'''*import pickle  
school = {  
 "Faculty": ["Sciences", "Engineering", "Law", "Agriculture", "Management Science"],  
 "department": ["Computer Science", "Civil Engineering", "Law", "Soil Science", "Mass Communication"]  
 }  
  
school\_file = open('school.txt', 'wb')  
pickle.dump(school, school\_file)  
school\_file.close()

''' This is to unpickle file. '''  
import pickle  
  
school\_file = open("school.txt", "rb")  
school = pickle.load(school\_file)  
school\_file.close()  
print(school)

16. What are the tools that help to find bugs or perform static analysis?

**Answer:** Pychecker and pylint: Pychecker warns about the complexity and style of the bug. Pylint checks if the module meets the coding standard.

17. How are arguments passed in Python by value or by reference? Give an example.

**Answer:** All **arguments** in the Python are **passed by reference**. This means if you change what a parameter refers to within a function, the change also reflects back in the calling function.

**Example:**

name={'Favour':29, 'Queeneth':35,'Victory':32,}  
def test(name):  
 new={'Stanley':38,'Gift':35}  
 name.update(new)  
 print("The first function",name)  
 return  
test(name)  
print("The Outer function:",name)

18. What are Dictionary and List comprehensions in Python? Provide examples.

**Answer:** List is a collection of data that are ordered, changeable and allows duplicate values. Dictionary is a collection of data values in key that is ordered, changeable and does not allow duplicate values.

**Examples:** *'''  
This is to show that list allows duplicate values  
'''*names = ["Favour", "Daniela", "Favour", "Faith", "Daniela"]  
  
print(names)

*'''  
This dictionary shows duplicate values is not allowed  
'''*food = {  
 "type": "Rice",  
 "class": "carbohydrate",  
 "kind": "brown",  
 "kind": "white"  
}  
print(food)

19. What is namespace in Python?

**Answer:** This is defined as a collection of names and details of objects referenced by the names**.**

20. What is pass in Python?

**Answer:** Pass is written when the programmer doesn’t know what to write anymore, at the line of pass the program executes but shows nothing. Pass can also be tagged as a placeholder**.**

21. What is unit test in Python?

**Answer:** Unit test is a test that checks if all specific parts of the functions behave correctly.

22. In Python what is slicing?

**Answer:** Slicing meansobtaining a sub-string from the given string, it can be done from start to end and is also a fast/quick way to access data from a list.

23. What is a negative index in Python?

**Answer:** Negative index means accessing the list from the end/last. [::-1]

24. How can the ternary operators be used in python? Give an example.

**Answer:** These are operators that evaluate a program based on conditional statement of true/false.

**Example:**

*'''  
A program to describe a ternary operator  
'''*a= 50  
b= 60  
if a > b:  
 print(True)  
else:  
 print(False)

25. What does this mean: \*args, \*\*kwargs? And why would we use it?

**Answer:** \*args this means non- keyword argument, It allows us to pass arguments more than the argument already defined. \*\*kwargs means keyword argument and it is used as a dictionary that maps each keyword to the value that we pass alongside it.

26. How are range and xrange different from one another?

**Answer:** Range is used to perform iterations function in python 3 version while xrange is used to iterate function in version 2 of python.

27. What is Flask and what can we use it for?

**Answer:** Flask is a web framework, used for developing web applications.

28. What are clustered and non-clustered index in a relational database?

**Answer:** Clustered indexalters the way data(rows) are stored on disk and non- clustered index structure exists as a separate first- class object in the database and does not alter rows in the table.

29. What is a ‘deadlock’ a relational database?

**Answer:** This is a condition that occurs when two or more database are waiting for each other and none is willing to give up the lock so the other process can be completed.

30. What is a ‘livelock’ a relational database?

**Answer:** This is a process where a request for exclusive lock is not allowed repeatedly because a series of overlapping shared locks keeps on interfering each other.

**2. Python string methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| capitalize() | Converts first character to upper case. | assessment = "a sincerely stressful period." y = assessment.capitalize() print (y) |
| casefold() | Converts string to lower case. | assessment = "a sincerely stressful period." y = assessment.casefold() print (y) |
| center() | Returns a centered string. | assessment = "stressful" y = assessment.center(30) print (y) |
| count() | Counts a number of times a specified value occurs | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.count("day") print (y) |
| endswith() | Returns true if a string endswith a particular value. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.endswith("hectic") print (y) |
| find() | Searches for a particular value and returns position found. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.find("hectic") print (y) |
| format() | Formats a specific value in a string. | assessment = "a stressful day, {wow} i never want a day as hectic as today" print(assessment.format(wow= "sigh")) |
| index() | Looks through the string for a specific value and returns the position it is found. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.index("never") print (y) |
| isalnum() | Output is TRUE if all characters in string is alphanumeric. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.isalnum() print (y) |
| isalpha() | Output is TRUE if all characters in string is alphabet. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.isalpha() print (y) |
| isdigit() | Output is TRUE if all characters in string are digits. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.isdigit() print (y) |
| islower() | Output is TRUE if all characters in string is lower case. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.islower() print (y) |
| isnumeric() | Output is TRUE if all characters in string is numeric. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.isnumeric() print (y) |
| isspace() | Returns True if there are only whitespace character | assessment = "CFG DEGREE" y = assessment.isspace() print (y) |
| istitle() | Output is TRUE if string follows the TITLE rule. | assessment = "a stressful day, i never want a day as hectic as today" y = assessment.istitle() print (y) |
| isupper() | Returns True if all case is UPPER case | assessment = "CFG DEGREE" y = assessment.isupper() print (y) |
| join() | Converts an iterable element into a string. | assessment = "CFG DEGREE" y = "Nano" z = y.join(assessment) print (z) |
| lower() | Converts a string to a lower case. | assessment = "CFG DEGREE" y = assessment.lower() print (y) |
| lstrip() | Outputs a left trimmed version of the string. | assessment = " CFG DEGREE " y = assessment.lstrip() print (y) |
| replace() | Replaces an old value with a new one. | assessment = "CFG DEGREE" y = assessment.replace('CFG', 'CODE FIRST GIRLS') print (y) |
| rstrip() | Outputs a right trimmed version of the string. | assessment = "CFG DEGREE " y = assessment.rstrip() print (y) |
| rspilt() | This splits the string at the designated separator and returns a list. | assessment = "CFG DEGREE" y = assessment.rsplit(",") print (y) |
| split() | Splits the string at the designated separator and returns a list. | assessment = "CFG DEGREE" y = assessment.split(",") print (y) |
| splitlines() | Splits the string at line breaks and returns a string | assessment = "CFG DEGREE\nA program I Love" y = assessment.splitlines() print (y) |
| startswith() | This returns True if it starts with a specific value. | assessment = "CFG DEGREE" y = assessment.startswith("C") print (y) |
| strip() | Returns a trimmed version of the string | assessment = " CFG DEGREE " y = assessment.strip() print (y) |
| swapcase() | This swaps cases, an upper case becomes a lower case. | assessment = "CFG DEGREE" y = assessment.swapcase() print (y) |
| title() | Converts the first character of each string to an upper case. | assessment = "CFG DEGREE" y = assessment.title() print (y) |
| upper() | Converts a string to an upper case. | assessment = "cfg degree" y = assessment.upper() print (y) |

**DESCRIPTION EXAMPLE**

**3. Python list methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| append() | Adds an element at the end of the list | names=['Favour', 'Joanna', 'Ruth'] names.append('Joy') print(names) |
| clear() | Removes all values/elements from list. | names=['Favour', 'Joanna', 'Ruth'] names.clear() print(names) |
| copy() | Makes a copy of the list and returns it. | names=['Favour', 'Joanna', 'Ruth'] names.copy() print(names) |
| count() | Counts a number of times a specific value occurs | names =['Favour', 'Joanna', 'Ruth'] ab = names.count("Favour") print(ab) |
| extend() | Add an element of a list to the end of a current list | names =['Favour', 'Joanna', 'Ruth'] surname =['Grace', 'John', ' Victory' ] names.extend(surname) print(names) |
| index() | Returns the index of an element with a specific value | names =['Favour', 'Joanna', 'Ruth'] y= names.index('Ruth') print(y) |
| insert() | Adds an element in a specific position | names =['Favour', 'Joanna', 'Ruth'] names.insert(0, 'Jonah') print(names) |
| pop() | Removes an element in a specific position. | names =['Favour', 'Joanna', 'Ruth'] names.pop(1) print(names) |
| remove() | Removes an item in a specified value | names =['Favour', 'Joanna', 'Ruth'] names.remove('Joanna') print(names) |
| reverse() | Reverses the order of the list. | names =['Favour', 'Joanna', 'Ruth'] names.reverse() print(names) |
| sort() | Sorts the list, either in ascending or descending order. | names =['Favour', 'Zaiah', 'Ruth'] names.sort() print(names) |

**4. Python tuple methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| count() | Counts a number of times a specific value occurs in a tuple. | departments = ('Science', 'Law', 'Engineering', ' Management Sciences') school= departments.count('Engineering') print(school) |
| index() | Returns the index of an element with a specific value in a tuple. | departments = ('Science', 'Law', 'Engineering', ' Management Sciences') school= departments.index('Law') print(school |

**5. Python dictionary methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| clear() | Removes all elements from dictionary. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } religion.clear() print(religion) |
| copy() | Makes a copy of the dictionary and returns it. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } religion.copy() print(religion) |
| fromkeys() | Returns a dictionary with a specific keys and value. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= dict.fromkeys(religion) print(y) |
| get() | Returns value of a specific key | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.get("Church") print(y) |
| items() | Outputs a list containing a tuple for each key value pair. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.items() print(y) |
| keys() | Outputs a list of dictionary keys. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.keys() print(y) |
| pop() | Removes the specified value and returns the dictionary | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } religion.pop("Traditionalist") print(religion) |
| popitem() | Removes the last inserted key value pair. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.popitem() print(y) |
| setdefault() | This method returns the value of the item with the specific key. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.setdefault("Church") print(y) |
| update() | It inserts a specific item into the dictionary. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.update({"Temple": "Buddhist"}) print(religion) |
| values() | Returns a list of all values in the dictionary. | religion = {  "Church": "Christian",  "Mosque": "Muslim",  "Traditionalist": "Pagans" } y= religion.values() print(y) |

**6. Python set methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| add() | Adds an element the set. | vegetables= {"Spinach", "Kale", "Oranges"} vegetables.add("Celery") print(vegetables) |
| clear() | Removes all element from the set. | vegetables= {"Spinach", "Kale", "Oranges"} vegetables.clear() print(vegetables) |
| copy() | Returns copy of the set. | vegetables= {"Spinach", "Kale", "Oranges"} vegetables.copy() print(vegetables) |
| difference() | Returns a set containing the difference between two or more sets. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Oranges"} food = fruits.difference(vegetables) print(food) |
| intersection() | Returns a set that contains similarity between two or more sets. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Oranges"} food= fruits.intersection(vegetables) print(food) |
| issubset() | Returns True if all values in SET1 is present in SET 2 | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Oranges", "Apple", "Watermelon"} food = fruits.issubset(vegetables) print(food) |
| issuperset() | This returns True if all values in the specific set exists in the original set. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Oranges", "Apple", "Watermelon"} food = vegetables.issuperset(fruits) print(food) |
| pop() | Removes an element from the set. | vegetables= {"Spinach", "Kale"} vegetables.pop() print(vegetables) |
| remove() | Removes the specific element in the set. | vegetables= {"Spinach", "Kale"} vegetables.remove("Spinach") print(vegetables) |
| symmetric-difference() | Returns a set with the proportional differences between two or more sets. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Watermelon"} food= fruits.symmetric\_difference(vegetables) print(food) |
| union() | Joins the elements in SET1 and SET2 then returns a set containing induplicate elements. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Oranges", "Apple", "Watermelon"} food = vegetables.union(fruits) print(food) |
| update() | Updates the current set with another set and confirms if there is duplicate of elements in the current set. | fruits ={"Apple", "Watermelon", "Oranges"} vegetables= {"Spinach", "Kale", "Watermelon"} fruits.update(vegetables) print(fruits) |

**7. Python file methods:**

**Describe each method and provide an example**

|  |  |  |
| --- | --- | --- |
| **METHOD** | **DESCRIPTION** | **EXAMPLE** |
| read() | Returns the number of bytes from the file. | name = open("school.txt", "r") print(name.read()) |
| readline() | Returns just one line of the file. | name = open("school.txt", "r") print(name.readline()) |
| readlines() | Outputs a list that contains each line in the file as a list item | name = open("school.txt", "r") print(name.readlines()) |
| write() | Writes a specific text to the file. | name = open("school.txt", "w") name.write("I love going to school") |
| writelines() | It appends/add to the file with intended words/text that you wish to add. | name = open("school.txt", "a") name.writelines("I love going to school," "even though it is stressful.") |