

GUIDELINES FOR VALUE ADDED COURSE (VAC) 2021-22

1. Evaluation

The value added courses shall carry 100 marks and shall be evaluated through internal Assessments only.

Continuous Assessment (CA)

The CA shall be a combination of a variety of tools such as class test, assignment, seminars, and viva-voce that would be suitable to the course.

Theory Course

The break-up of marks shall be as follows:

t S F V

| Item | Marks | Grading Marks | | |
|---|-------|---------------|--|--|
| Quiz Tests/Class Assignments/Home Assignments/Google form online | 40 | 4 | | |
| test | | | | |
| Seminar/ Class Presentations /Class Performance | 30 | 3 | | |
| Viva-voce | 30 | 3 | | |
| Total | 100 | 10 | | |

| Practical Course | | | | | | |
|--|-----|----|--|--|--|--|
| Item Marks Grading Marks | | | | | | |
| Demonstration of Skills and Viva Voce | 40 | 4 | | | | |
| Assignments& Exercises | 30 | 3 | | | | |
| Lab Performance | 30 | 3 | | | | |
| Total | 100 | 10 | | | | |

Continuous Assessment Tests

- i. Continuous assessments shall be conducted preferably one in the middle and other at the end of the course.
- ii. The duration of the test, the pattern of question paper and the units included shall be decided by the Course Coordinator and prior intimation shall be given to the students.
- iii. The assessment shall be done by the course teacher/Course Coordinator.

iv. No improvement option shall be available for CA. However, if a student could not attend the test for any valid reason, the prerogative of arranging a special test lies with the Course Coordinator in consultation with the Head of the Department.

2. Grading

Evaluation of the performance of the student will be rated as shown in the Table

| Marks | Grade | Grade Point |
|------------------------|-------|--------------------|
| 90 - 100 | О | 10 |
| 75 – 89 | A+ | 9 |
| 65 - 74 | А | 8 |
| 55 - 64 | B+ | 7 |
| 50 - 54 | В | 6 |
| 45 - 49 | C | 5 |
| 40 - 44 | Р | 4 |
| Less than 40 or absent | F | 0 |

The grades and credits obtained in VACs shall not be considered for calculating the GPA and CGPA of the regular course that the student is undergoing. The percentage of marks obtained by a candidate in a course will be indicated in the awarding certificate.

3. Awarding Certificate

On successful completion of the VAC, the student shall be issued a certificate duly signed by the Head of the Department and the Course Coordinator.

Course Coordinator

HoD

| 1 | Assessment Ethical Hacking | |
|-------------|--|---------|
| <u>* r</u> | ndicates required question | |
| 1. | Email * | |
| 2. | Email * | |
| | | |
| | | |
| | | |
| | | |
| 3. | Name * | |
| 4. | Roll No. * | |
| 5. | 1) What built-in list method would you use to remove items from a list? Mark only one oval. | 1 point |
| | delete() method | |
| | | |
| | del(my_list) | |
| | \square non() method | |
| | | |

6. 2) What is the purpose of an if/else statement?

Mark only one oval.

It tells the computer which chunk of code to run if the instructions you coded are incorrect.

It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful.

It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false.

It tells the computer which chunk of code to run if the is enough memory to handle it, and which chunk of code to run if there is not enough memory to handle it.

7. 3) What does the built-in map() function do?

1 point

Mark only one oval.

It creates a path from multiple values in an iterable to a single value.

It applies a function to each item in an iterable and returns the value of that function.

It converts a complex value type into simpler value types.

It creates a mapping between two different elements of different iterables.

8. 4) If you don't explicitly return a value from a function, what happens? 1 point

Mark only one oval.

The function will return a RuntimeError if you don't return a value.

If the return keyword is absent, the function will return None.

If the return keyword is absent, the function will return True.

The function will enter an infinite loop because it won't know when to stop executing its code.

9. 5) When does a for loop stop iterating?

Mark only one oval.

when it encounters an infinite loop

when it encounters an if/else statement that contains a break keyword

when it has assessed each item in the iterable it is working on or a break keyword is encountered

| \frown | | | | <i>c</i> | | | | \sim | |
|----------|------|-----|---------|----------|-----|------|---------|--------|-------|
| .) | when | the | runtime | tor | the | loop | exceeds | O(| n^2) |
| \smile | | | | | | | | - (| ·· _/ |

10. 6) What is key difference between a set and a list?

1 point

1 point

Mark only one oval.

A set is an ordered collection unique items. A list is an unordered collection of non-unique items.

Elements can be retrieved from a list but they cannot be retrieved from a set.

A set is an ordered collection of non-unique items. A list is an unordered collection of unique items.

A set is an unordered collection unique items. A list is an ordered collection of non-unique items.

11. 7) Review the code below. What is the correct syntax for changing the 1 point price to 1.5?

fruit_info = { 'fruit': 'apple', 'count': 2, 'price': 3.5 }

Mark only one oval.

- fruit_info ['price'] = 1.5
- ____ my_list [3.5] = 1.5
- 1.5 = fruit_info ['price]
- ____ my_list['price'] == 1.5

12. 8) You are given a piece of code. Assume m and n are already defined as 1 point some positive integer value. When it completes, how many tuples will my list contain?

mylist = [] for i in range(m): for j in range(n): mylist.append((i,j))

Mark only one oval.

| \bigcirc | m |
|------------|-------|
| \bigcirc | m + n |
| \bigcirc | n |
| \bigcirc | m * n |

13. 9) Assume m, n and p are positive integers. In the following 1 point comprehension, how many times will the function randint be called?

[[randint(1,100) for i in range(m)] for j in range(n)] for k in range(p)]

Mark only one oval.

_____m_n_p

_____ the greater value of (m,n,p)

1 million

(____) m + n + p

14. 10) What is the correct syntax for replacing the string apple in the list with 1 point the string orange?

my_list = [2, 'apple', 3.5]

Mark only one oval.

orange = my_list[1]

____ my_list[1] = 'orange'

____ my_list['orange'] = 1

____ my_list[1] == orange

15. 11) Which comparison of lists and tuples in Python is correct?

Mark only one oval.

Use lists instead of tuples when you have a collection of related but dissimilar objects.

Use tuples instead of lists when you have a common collection of similar objects.

Use tuples instead of lists for functions that need to return multiple values.

Use lists instead of tuples when the position of elements is important.

16. 12) What will this code output to the screen? for i in range(5):

1 point

else:

print(i)

print("Done!")

Mark only one oval.

_____ 1 2 3 4 5 Done!

0 1 2 3 4 5 Done!

0 1 2 3 4 Done!

You will get a syntax error.

1 point

17. 13) Which collection is ordered, changeable, and allows duplicate members?

Mark only one oval.

SET

- 18. 14) What happens if the file is not found in the following Python code?1 point a=False

while not a:

try:

f_n = input("Enter file name")

```
i_f = open(f_n, 'r')
```

except:

print("Input file not found")

Mark only one oval.

| \bigcirc | No | error |
|------------|----|-------|
|------------|----|-------|

Assertion error

Input output error

Name error

19. 15) What will be the output of the following Python code?
1 point Ist = [1, 2, 3]
Ist[3]

Mark only one oval.

- NameError
- ◯ ValueError
- TypeError

| 20. | 16) Identify the type of error in the following Python codes? |
|-----|---|
| | Print("Good Morning") |
| | print("Good night) |

Mark only one oval.

Syntax, Syntax

🔵 Semantic, Syntax

Semantic, Semantic

- Syntax, Semantic
- 21. 17) An exception is _____

Mark only one oval.

an object

a special function

🔵 a standard module

🔵 a module

22. 18) How many keyword arguments can be passed to a function in a single 1 point function call?

Mark only one oval.

🔵 zero

one

- ____ zero or more
- one or more

1 point

23. 19) How many except statements can a try-except block have?

1 point

Mark only one oval.

0

1

____ more than one

- ____ more than zero
- 24. 20) Which of the following will print the pi value defined in math module? 1 point

Mark only one oval.

____ print(pi)

____ print(math.pi)

from math import pi print(pi)

_____ from math import pi print(math.pi)

25. 21) Which operator is used in Python to import modules from packages? 1 point

Mark only one oval.

◯.

____*

(____->

____&

26. 22) How is a function declared in Python?

1 point

Mark only one oval.

def function function_name():

declare function function_name():

def function_name():

declare function_name():

27. 23) Which one of the following is the correct way of calling a function? 1 point

Mark only one oval.

function_name()

- call function_name()
- ret function_name()
- function function_name()
- 28. 24) Choose the correct option with reference to below Python code? 1 point def fn(a):

print(a)

x=90

fn(x)

Mark only one oval.

🔵 x is the formal argument.

_____ a is the actual argument.

 \bigcirc fn(x) is the function signature.

🔵 x is the actual argument.

29. 25) Which one of the following is incorrect?

Mark only one oval.

The variables used inside function are called local variables.

The local variables of a particular function can be used inside other functions, but these cannot be used in global space

The variables used outside function are called global variables

| \square | In order to | change the | value of g | global varia | able inside | function, | keyword | global is |
|-----------|-------------|------------|------------|--------------|-------------|-----------|---------|-----------|
| use | d. | | | | | | | |

1 point

This content is neither created nor endorsed by Google.



1/5/24, 11:48 AM

Assessment

| Email Address | Score | Email |
|---------------------------------------|---------|-----------------------------|
| | | Linan |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| privanialmalbatra4@amail.aa | | privanialmalbatra/@amail.aa |
| priyanjamanotra4@gman.co | 22 / 25 | |
| | 23723 | III |
| | | |
| | | |
| | | |
| | 04/05 | |
| gopairajput 11a@gmail.com | 24 / 25 | gopairajputi i a@gmail.com |
| | | |
| | | |
| | | |
| | | |
| singhal2002y@gmail.com | 9 / 25 | singhal2002y@gmail.com |
| | | |
| | | |
| | | |
| | | |
| siddharthpopli15@gmail.com | 24 / 25 | Siddharthpopli15@gmail.con |
| | | |
| | | |
| | | |
| | | |
| rohittanwar1287@gmail.com | 20 / 25 | rohittanwar1287@gmail.com |
| | | |
| | | |
| | | |
| | | |
| vrindasuneia2004@gmail.com | 18 / 25 | vrindasuneia2004@gmail.com |
| | |] |
| | | |
| | | |
| | | |
| praijwaldwivedi01@gmail.com | 9 / 25 | praijwaldwivedi01@gmail.com |
| prajjivalavi v čalo i tegritali česti | 0720 | |
| | | |
| | | |
| | | |
| vivekabarma9491@gmail.com | 19 / 25 | 9750770049 |
| viveksnarmao4o i@gmail.com | 10725 | 0750779940 |
| | | |
| | | |
| | | |
| kesnavkumar691999@gmail.c | | |
| om | 23 / 25 | 9205490649 |

| thakurharsh236@gmail.com | 21 / 25 | 8938810595 |
|---------------------------------|---------|---------------------------------|
| | | |
| deepakaiml12@gmail.com | 17 / 25 | 9671248852 |
| | | |
| gibrailzaidi@gmil.com | 17 / 25 | 8126560284 |
| | | |
| gazabthakur07@gmail.com | 17 / 25 | 8958280167 |
| adityamishra.am71@gmail.co m | 23 / 25 | 9835319964 |
| bhinderfazal@gmail.com | 23 / 25 | 90454097615 |
| anujmishra7069@gmail.com | 23 / 25 | 8521407255 |
| adarshraiput1914@gmail.com | 17 / 25 | adarshraiput1914@gmail.com |
| | | |
| bhardwajsagar5498@gmail.co m | 11 / 25 | bhardwajsagar5498@gmail.co m |
| | | |
| choudharyanas19@gmail.com | 17 / 25 | 8920295476 |

| hs9650612804@gmail.com | 21 / 25 | 9650612804 |
|----------------------------------|---------|---------------------------|
| | | |
| faizannasim59@gmail.com | 20 / 25 | 7532053343 |
| | | |
| md.amberkhan9631@gmail.co m | 15 / 25 | 8700753024 |
| | | |
| Srivastavayushbth@gmail.co m | 17 / 25 | 6205646148 |
| | | |
| gaganshuyadav16@gmail.co m | 21 / 25 | 9868043151 |
| | | |
| faridianzar786@gmail.com | 20 / 25 | 7895829079 |
| | | |
| arunpandey916162@gmail.co m | 17 / 25 | 8756295300 |
| | | |
| anshumansoni4546@gmail.co m | 17 / 25 | 7237031490 |
| | | |
| himanshusharma1454@gmail. com | 17 / 25 | 6393381416 |
| | | |
| vermavishesh345@gmail.com | 16 / 25 | vermavishesh345@gmail.com |

| kamranahmadmd463@gmail.c om | 20 / 25 | kamranahmadmd463@gmail.c om |
|---------------------------------|---------|---------------------------------|
| | | |
| viraz.rr@gmail.com | 16 / 25 | viraz.rr@gmail.com |
| | | |
| manishking241@gmail.com | 18 / 25 | 9177893190 |
| | | |
| shivamsinghcse19@gmail.co m | 20 / 25 | shivamsinghcse19@gmail.co m |
| | | |
| himanshurautela87@gmail.co m | 22 / 25 | himanshurautela87@gmail.co m |
| | | |
| rawatsumit1290@gmail.com | 23 / 25 | rawatsumit1290@gmail.com |
| | | |
| hs185008@gmail.com | 21 / 25 | hs185008@gmail.com |
| | | |
| me.souravk03@gmail.com | 7 / 25 | me.souravk03@gmail.com |
| | | |
| arcuscode@gmail.com | 15 / 25 | arcuscode@gmail.com |
| | | |
| naveensangwanhas@gmail.co m | 15 / 25 | naveensangwanhas@gmail.co m |

| vaibhavjain98111@gmail.com | 15 / 25 | vaibhavjain98111@gmail.com |
|------------------------------------|---------|------------------------------------|
| | | |
| fagunchauhan4@gmail.com | 16 / 25 | fagunchauhan4@gmail.com |
| | | |
| devmohansharma177013@gm ail.com | 19 / 25 | devmohansharma177013@gm ail.com |
| | | |
| sidra.tabassum2004@gmail.co m | 12 / 25 | sidra.tab1804@gmail.com |
| | | |
| jhanvikhanna2@gmail.com | 13 / 25 | jhanvikhanna2@gmail.com |
| | | |
| aviralrastogi2004@gmail.com | 13 / 25 | aviralrastogi2004@gmail.com |
| | | |
| aanchalmangla73@gmail.com | 19 / 25 | aanchalmangla73@gmail.com |
| | | |
| rajpriya012005@gmail.com | 20 / 25 | rajpriya012005@gmail.com |
| | | |
| tcccvv@gmail.con | 6 / 25 | Yfgb@ynmq |
| | | |
| summitparmar@gmail.com | 23 / 25 | summitparmar@gmail.com |

| PARTHG5082@GMAIL.COM | 22 / 25 | parthg5082@gmail.com |
|---------------------------|----------|---------------------------|
| - | | |
| | | |
| | | |
| mittalmanya10@gmail.com | 17 / 25 | mittalmanya10@gmail.com |
| | | |
| | | |
| mrukaiya13@gmail.com | 24 / 25 | mrukaiya13@gmail.com |
| | | |
| | | |
| | | |
| sagarparasher04@gmail.com | 23 / 25 | sagarparasher04@gmail.com |
| | | |
| asmarthmadaan72@gmail.as | | aamarthmadaan70@amail.aa |
| m | 23 / 25 | m |
| | | |
| | | |
| | o., / o- | |
| yuvsharma464@gmail.com | 24 / 25 | yuvsharma464@gmail.com |
| | | |
| | | |
| ajeetsingh4656@gmail.com | 24 / 25 | ajeetsingh4656@gmail.com |
| | | |
| | | |
| | | |
| shreymitra376@gamil.com | 20 / 25 | shreymitra376@gmail.com |
| | | |
| | | |
| nsnikitasoni17@gmail.com | 17 / 25 | nsnikitasoni17@gmail.com |
| | | |
| | | |
| | | |
| namitjoshi2004@gmail.com | 24 / 25 | namitjoshi2004@gmail.com |

| mankirat.matharu@gmail.com | 10 / 25 | mankirat.matharu@gmail.com |
|----------------------------|---------|----------------------------|

| Namo | Roll No. | 1) What built in list mathed |
|--------------------|-------------|------------------------------|
| Name | NOILINO. | 1) What built-in list method |
| | | would you use to remove |
| | | items from a list? |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| Privanial Malbotra | 13 | non() method |
| | 13 | |
| | | |
| | | |
| | | |
| | | |
| Gopal Kumar | 02721302022 | .pop() method |
| | | |
| | | |
| | | |
| | | |
| Vetherth singhel | 04221202022 | non() mothed |
| ratharth singhai | 04321302022 | .pop() method |
| | | |
| | | |
| | | |
| | | |
| Siddharth popli | 53 | .pop() method |
| · · | | |
| | | |
| | | |
| | | |
| | 00404000000 | |
| Rohit Tanwar | 00121302022 | .pop() method |
| | | |
| | | |
| | | |
| | | |
| Vrinda Suneia | 03521302022 | .pop() method |
| ······ | | ·F -F () ········· |
| | | |
| | | |
| | | |
| | | |
| Prajjwal Dwivedi | 2.10132E+12 | pop(my_list) |
| | | |
| | | |
| | | |
| | | |
| vivek | 2200041 | non(my_list) |
| | 2200041 | |
| | | |
| | | |
| | | |
| | | |
| Keshav Kumar | 2200040 | .pop() method |

| Harsh gour | 2.10132E+12 | .pop() method |
|----------------|-------------|---------------|
| | | |
| DEEPAK | 2.10132E+12 | .pop() method |
| | | ······ |
| | | |
| Gibrail Zaidi | 2.10132E+12 | .pop() method |
| | | |
| Gazab Bhati | 2.10132E+12 | .pop() method |
| | | |
| Aditya Mishra | 2.10132E+12 | .pop() method |
| | | |
| Fazal Singh | 2.10132E+12 | .pop() method |
| | | |
| ANUJ MISHRA | 2.10132E+12 | .pop() method |
| | | |
| | | |
| Adarsh Kumar | 2.10132E+12 | pop(my_list) |
| | | |
| sagar sharma | 2200122 | del(my_list) |
| | | |
| Anas choudhary | 2200094 | .pop() method |

| Harsh Sharma | 2.10132E+12 | .pop() method |
|-----------------------|-------------|------------------|
| | | |
| | | |
| faizan nasim | 2200067 | .delete() method |
| | | |
| | 0.404005.40 | |
| Md Amber Khan | 2.10132E+12 | .pop() method |
| | | |
| Ayush kumar Srivastav | 2.10132E+12 | .pop() method |
| | | |
| | | |
| Gaganshu yadav | 2.10132E+12 | .pop() method |
| | | |
| | | |
| Anzar Hashmat | 2200095 | .pop() method |
| | | |
| Arun Pandey | 2.10132E+12 | .pop() method |
| | | |
| | | |
| Anshuman Soni | 2.10132E+12 | pop(my_list) |
| | | |
| | | |
| himanshu sharma | 2200071 | .pop() method |
| | | |
| Vishesh Verma | 2.10132E+12 | .pop() method |

| Kamran Ahmad | 2.19132E+12 | .pop() method |
|------------------------|-------------|------------------|
| | | |
| Viraz Anand Gupta | 2.10132E+12 | .pop() method |
| | | |
| manish kumar singh | 2.10132E+12 | pop(my_list) |
| | | n en () m eth ed |
| snivam singn | 2.10132E+12 | .pop() method |
| Himonohu Singh Poutolo | 2 101225+12 | per() method |
| | 2.10132E+12 | .pop() method |
| Sumit rawat | 2.10132E+12 | .pop() method |
| | | |
| Himanshu singh | 2.10132E+12 | .pop() method |
| | | |
| sourav kumar | 2.10132E+12 | pop(my_list) |
| Nav Verma | 01217002022 | .pop() method |
| | | |
| Naveen Sanawan | 02/17002022 | non() method |
| naveen oangwall | 02711002022 | .pop() memou |

| | | - |
|-------------------|---------------|---------------------|
| | | |
| VAIBHAV JAIN | 00617002022 | pop(my_list) |
| | | |
| | | |
| Fagun | 41 | .pop() method |
| | | |
| Doy Mohan Sharma | 01017002022 | non() mothed |
| Dev Monari Sharma | 01017002022 | |
| | | |
| Sidra Tabassum | 04617002022 | .delete() method |
| | | |
| | | |
| Jhanvi Khanna | 35617002022 | del(my_list) |
| | | |
| | 0.10.17000000 | |
| Aviral Rastogi | 01817002022 | .pop() method |
| | | |
| Aanchal | 04717002022 | .pop() method |
| | | |
| | | |
| Priya kumari | 00217002022 | del(my_list) |
| | | |
| D# | - | delete () and the d |
| | <u>F</u> | .delete() method |
| | | |
| Shankar | 3 | pop(my_list) |

| Parth Goyal | 00421302022 | pop(my_list) |
|----------------|-------------|------------------|
| | | |
| Manya Mittal | 35321302022 | .pop() method |
| | | |
| Rukaiya | 38 | .pop() method |
| | | |
| Sagar parasher | 22 | .pop() method |
| | | |
| Samarth nadaan | 01421302022 | .delete() method |
| | | |
| Yuv Sharma | 32 | .pop() method |
| | | |
| Ajeet Singh | 02121302022 | .pop() method |
| | | |
| Shrey | 17 | pop(my_list) |
| | | |
| Nikita Soni | 02321302022 | pop(my_list) |
| | | |
| Namit Joshi | 8 | .pop() method |

| Mankirat Singh | 06 | .delete() method | |
|----------------|----|------------------|--|

| 2) What is the purpose of an if/else statement? | 3) What does the built-in map() function do? | 4) If you don't explicitly return a value from a function, what happens? |
|---|---|--|
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It converts a complex value type into simpler value types. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful. | It applies a function to each item in an iterable and returns the value of that function. | The function will enter an infinite loop because it won't know when to stop executing its code. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |

| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
|---|---|--|
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will return a RuntimeError if you don't return a value. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| chunk of code to run if the is enough memory to handle it, and which chunk of code to run if there is not enough memory to handle it. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return True. |

| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return True. |
|---|---|--|
| It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will return a RuntimeError if you don't return a value. |
| It tells the computer which chunk of code to run if the instructions you coded are incorrect. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will return a RuntimeError if you don't return a value. |
| chunk of code to run if the is enough memory to handle it, and which chunk of code to run if there is not enough memory to handle it. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return True. |
| It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful. | It converts a complex value type into simpler value types. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will return a RuntimeError if you don't return a value. |

| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
|---|---|--|
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will return a RuntimeError if you don't return a value. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful. | It creates a path from multiple values in an iterable to a single value. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It runs one chunk of code if all the imports were successful, and another chunk of code if the imports were not successful. | It converts a complex value type into simpler value types. | The function will enter an infinite loop because it won't know when to stop executing its code. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a path from multiple values in an iterable to a single value. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | | If the return keyword is absent, the function will return None. |

| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a path from multiple values in an iterable to a single value. | If the return keyword is absent, the function will return None. |
|---|---|--|
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It converts a complex value type into simpler value types. | If the return keyword is absent, the function will return None. |
| | | The function will enter an infinite loop because it won't know when to stop executing its code. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a mapping between two different elements of different iterables. | The function will return a RuntimeError if you don't return a value. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | The function will enter an infinite loop because it won't know when to stop executing its code. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It converts a complex value type into simpler value types. | If the return keyword is absent, the function will return True. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |

| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It converts a complex value type into simpler value types. | If the return keyword is absent, the function will return None. |
|---|---|---|
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It creates a path from multiple values in an iterable to a single value. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |
| It executes one chunk of code if a condition is true, but a different chunk of code if the condition is false. | It applies a function to each item in an iterable and returns the value of that function. | If the return keyword is absent, the function will return None. |

| It executes one chunk of code | | |
|--------------------------------|--------------------------------|----------------------------------|
| if a condition is true, but a | | If the return keyword is |
| different chunk of code if the | It converts a complex value | absent, the function will return |
| condition is false. | type into simpler value types. | True. |

| 5) When does a for loop stop | 6) What is key difference | Review the code below. |
|--------------------------------|---------------------------------------|--|
| iterating? | between a set and a list? | What is the correct syntax for |
| ů, s | | changing the price to 1.5? |
| | | |
| | | fruit_info = { 'fruit': 'apple' |
| | | |
| when it has assessed each | A set is an upordered | |
| item in the iterable it is | A set is an unordered | |
| | collection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| | | |
| when it has assessed each | A set is an unordered | |
| item in the iterable it is | collection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| | | |
| | A set is an ordered collection | |
| when it encounters an if/else | of non-unique items. A list is | |
| statement that contains a | an unordered collection of | |
| break keyword | | fruit info ['price'] = 1.5 |
| | | |
| when it has assessed each | A set is an unordered | |
| itom in the iterable it is | collection unique items. A list | |
| | conection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| | | |
| when it has assessed each | A set is an unordered | |
| item in the iterable it is | collection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| | | |
| when it has assessed each | A set is an unordered | |
| item in the iterable it is | collection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| | | |
| | A set is an ordered collection | |
| | unique items. A list is an | |
| when it encounters an infinite | unordered collection of non- | |
| loop | unique items | fruit_info['price'] = 1.5 |
| P | | |
| when it has assessed each | A set is an unordered | |
| item in the iterable it is | collection unique items Δ list | |
| working on or a break | is an ordered collection of non | |
| working on or a bleak | | fruit info ['nrioc'] = 1.5 |
| keyword is encountered | | inut_inio [price] = 1.5 |
| when it has account each | A set is an upordorod | |
| item in the iterable it is | alloction unique iteme A list | |
| | conection unique items. A list | |
| working on or a break | is an ordered collection of non- | |
| keyword is encountered | unique items. | [truit_info ['price'] = 1.5 |

| when it has assessed each item in the iterable it is working on or a break | A set is an unordered collection unique items. A list is an ordered collection of non- | |
|--|---|----------------------------|
| keyword is encountered | unique items | fruit_info['price'] = 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an infinite loop | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an infinite loop | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | my_list['price'] == 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
|--|---|----------------------------|
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | my_list['price'] == 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | my_list['price'] == 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |

| when it has assessed each item in the iterable it is working on or a break | A set is an unordered collection unique items. A list is an ordered collection of non- | fmuit info ['price'] = 1.5 |
|--|---|----------------------------|
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an ordered collection unique items. A list is an unordered collection of non- unique items. | my_list['price'] == 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |

| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
|--|---|----------------------------|
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an ordered collection unique items. A list is an unordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an ordered collection unique items. A list is an unordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when the runtime for the loop exceeds O(n^2) | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | my_list['price'] == 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |

| | | Ĩ |
|--|---|----------------------------|
| when it encounters an if/else statement that contains a break keyword | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it encounters an infinite loop | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an ordered collection of non-unique items. A list is an unordered collection of unique items. | fruit_info ['price'] = 1.5 |
| when it has assessed each item in the iterable it is working on or a break keyword is encountered | A set is an unordered collection unique items. A list is an ordered collection of non- unique items. | fruit_info ['price'] = 1.5 |

| when it has assessed each | A set is an ordered collection | |
|----------------------------|--------------------------------|----------------------------|
| item in the iterable it is | unique items. A list is an | |
| working on or a break | unordered collection of non- | |
| keyword is encountered | unique items. | fruit_info ['price'] = 1.5 |

| 8) You are given a piece of | 9) Assume m, n and p are | 10) What is the correct |
|--|------------------------------|-----------------------------------|
| code. Assume m and n are | positive integers. In the | syntax for replacing the string |
| already defined as some | following comprehension, how | apple in the list with the string |
| positive integer value. When | many times will the function | orange? |
| it completes, how many tuples | randint be called? | 5 |
| ······································ | | |
| | | |
| | | |
| | | |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | m _ n _ p | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | m + n + p | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | | my_list[1] = 'orange' |
| | <u></u> P | |
| | | |
| | | |
| | | |
| | | |
| m ^ n | m_n_p | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | m + n + p | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | the greater value of (m,n,p) | my_list[1] = 'orange' |
| | | |
| | | |
| | | |
| | | |
| m * n | m n p | my_list[1] = 'orange' |
| | ···· _ ·· _ ٢ | |
| | | |
| | | |
| | | |
| m * n | m n n | my list[1] = 'orange' |
| | | |

| m * n | | my_list[1] = 'orange' |
|-------|--------------|-----------------------|
| | | |
| m * n | m + n + p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | mu list[4] - lovensel |
| | <u>m_n_p</u> | my_list[1] = orange |
| m * n | m n p | my_list[1] = 'orange' |
| | | /_ (/) |
| m * n | | my_list[1] = 'orange' |
| | | |
| m * n | m n p | mv_list[1] = 'orange' |

| m * n | m_n_p | my_list[1] = 'orange' |
|-------|------------------------------|-----------------------|
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | 1 million | my_list[1] = 'orange' |
| | | |
| m + n | the greater value of (m,n,p) | my_list[1] = 'orange' |
| | | |
| m * n | m + n + p | mv_list[1] = 'orange' |

| m * n | m_n_p | my_list[1] = 'orange' |
|-------|------------------------------|-----------------------|
| | | |
| m * n | m + n + p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m + n | the greater value of (m,n,p) | orange = my_list[1] |
| | | |
| m | m_n_p | my_list[1] = 'orange' |
| | | |
| m | m n p | my_list[1] = 'orange' |

| m * n | m + n + p | my_list[1] = 'orange' |
|-------|------------------------------|-----------------------|
| | | |
| m + n | m + n + p | my_list[1] = 'orange' |
| | | |
| m + n | m_n_p | my_list[1] = 'orange' |
| | | |
| m + n | m_n_p | my_list[1] = 'orange' |
| | | |
| m + n | the greater value of (m,n,p) | my_list[1] = 'orange' |
| | | |
| m * n | m + n + p | my_list[1] = 'orange' |
| | | |
| m + n | m + n + p | my_list[1] = 'orange' |
| | | |
| m * n | m_n_p | my_list[1] = 'orange' |
| | | |
| m | the greater value of (m,n,p) | my_list[1] == orange |
| | | |
| m * n | m n p | my list[1] = 'orange' |

| m * n | m_n_p | my_list[1] = 'orange' |
|-------|-----------|-----------------------|
| m * n | m + n + p | my_list[1] = 'orange' |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m n n | my list[1] = 'orongo' |
| | <u> </u> | ny_nst[1] – orange |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m_n_p | my_list[1] = 'orange' |
| m * n | m n p | mv_list[1] = 'oranœ' |

| <u>m * n</u> | my_list[1] = 'orange' |
|--------------|-----------------------|

| 11) Which comparison of lists | 12) What will this code output | 13) Which collection is |
|--|--------------------------------|---------------------------|
| and tuples in Pythen is | to the screen? | ordered changeable and |
| | | |
| correct? | for I in range(5): | allows duplicate members? |
| | print(i) | |
| | else: | |
| | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values | 0 1 2 3 4 Donel | UST |
| | | |
| | | |
| Line lists instead of tuples | | |
| Use lists instead of tuples | | |
| when you have a collection of | Vou will got a overtox orror | ert. |
| | Fou will get a syntax error. | |
| | | |
| l la structura in stand of lists for | | |
| Use tupies instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use lists instead of tuples | | |
| when the position of elements | | |
| is important. | 0 1 2 3 4 5 Done! | LIST |
| · · | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values | 0 1 2 3 4 Done! | LIST |
| ······································ | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values | 0 1 2 3 4 Done! | LIST |
| | | |

| line tout a brack of the first from | | |
|---|-------------------|------|
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| · · | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| line tomber in the sile of the feature | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lies tuming instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lies tuming instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lies turies instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Leo tunlos instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 5 Done! | LIST |
| | | |
| I lee tunles instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |

| l las tumbre instant of lists for | | |
|-----------------------------------|-----------------|------|
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lise tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |

| Use tuples instead of lists for | | |
|----------------------------------|-----------------|-------|
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lies tuples instead of lists for | | |
| functions that need to return | | |
| multiple values | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | LISI |
| | | |
| Lies tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 1 2 3 4 5 Done! | TUPLE |
| | | |
| | | |
| | | |
| | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| | | |
| | | |
| | 0 1 2 3 4 Done! | LIST |

| I to a line to the and a fitter to a | | |
|---|------------------------------|------------|
| Use lists instead of tuples | | |
| is important | 0 1 2 3 4 Donel | UST |
| | | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lise tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | LIST |
| , | | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | TUPLE |
| | | |
| Use lists instead of tuples | | |
| when you have a collection of | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | You will get a syntax error | |
| | Tou will get a syntax error. | |
| | | |
| Use tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| Lies tuples instead of lists | | |
| when you have a common | | |
| collection of similar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 5 Done! | DICTIONARY |
| | | |
| I lse tuples instead of lists for | | |
| functions that need to return | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |

| Use tuples instead of lists for | | |
|---------------------------------|-----------------|------|
| multiple values. | 0 1 2 3 4 Done! | LIST |
| · · | | |
| | | |
| Use lists instead of tuples | | |
| related but dissimilar objects. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| · · | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values. | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values | 0 1 2 3 4 Done! | LIST |
| | | |
| | | |
| Use tuples instead of lists for | | |
| runctions that need to return | 0 1 2 3 4 Donel | LIST |
| การแม่วย งิสเนอง. | | |
| | | |
| Use tuples instead of lists for | | |
| multiple values | 0 1 2 3 4 Done! | LIST |
| | | |

| 0 1 2 3 4 5 Donal | |
|-------------------|------------|
| U I Z 3 4 5 DONE! | DICTIONARY |

| 14) What happens if the file is not found in the following Python code? a=False | 15) What will be the output of the following Python code? Ist = [1, 2, 3] Ist[3] | 16) Identify the type of error in the following Python codes? Print("Good Morning") print ("Good night) |
|--|---|---|
| | | |
| Input output error | IndexError | Semantic, Syntax |
| | | |
| No error | IndexError | Semantic, Syntax |
| Input output error | IndexError | Syntax, Syntax |
| | | |
| No error | IndexError | Semantic, Syntax |
| No error | IndexError | Syntax, Syntax |
| | | |
| Input output error | IndexError | Syntax, Syntax |
| | | |
| No error | ValueError | Semantic, Syntax |
| Input output error | IndexError | Syntax, Syntax |
| | | |
| No error | IndexError | Semantic, Syntax |

| Input output error | IndexError | Semantic, Syntax |
|--------------------|------------|------------------|
| Input output error | IndexError | Semantic Syntax |
| | | |
| Input output error | IndexError | Syntax, Semantic |
| | | |
| Input output error | IndexError | Syntax, Semantic |
| No error | IndexError | Semantic Syntax |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| No error | IndexError | Semantic, Syntax |
| No error | IndexError | Semantic, Syntax |
| | | |
| Input output error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |

| IndexError | Semantic, Syntax |
|--------------|--|
| | |
| | |
| | |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Comentia Syntax |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| | |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Syntax, Semantic |
| | |
| | |
| | |
| Index France | Comentia Cunter |
| Indexerior | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Semantic, Syntax |
| | |
| | |
| | |
| IndexError | Semantic, Svntax |
| | IndexError IndexError IndexError IndexError IndexError IndexError IndexError IndexError |

| No error | IndexError | Semantic, Syntax |
|--------------------|-------------|------------------|
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| Input output orror | IndovError | Comentia Syntax |
| input output error | | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | Indox Error | Comentia Cunter |
| input output error | | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Svntax, Svntax |
| | | |
| | | |
| | | |
| No error | IndexError | Syntax, Syntax |

| Assertion error | IndexError | Semantic Syntax |
|--------------------|-------------|---------------------|
| | | |
| | | |
| | | |
| | | |
| | IndexError | Semantic Syntax |
| | | |
| | | |
| | | |
| | | |
| No orror | IndoxError | Somantia Syntax |
| | Indexentor | |
| | | |
| | | |
| | | |
| | IndoxError | Somantia Syntax |
| | Indexentor | |
| | | |
| | | |
| | | |
| Input output orror | Value | Compantia Syntax |
| | | Semantic, Syntax |
| | | |
| | | |
| | | |
| Acception owner | Index Error | Compantia Compantia |
| Assention error | Indexertor | Semantic, Semantic |
| | | |
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | | |
| | | |
| Input output error | IndexError | Semantic, Syntax |
| | | |
| | | |
| | | |
| | | |
| Assertion error | ValueError | Semantic, Semantic |
| | | |
| | | |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |

| No error | IndexError | Semantic, Syntax |
|----------|------------|------------------|
| | | |
| | | |
| No error | IndexError | Syntax, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |
| | | |
| | | |
| No error | ValueError | Semantic, Syntax |
| | | |
| | | |
| No error | IndexError | Semantic, Syntax |

| Name error | IndexError | Syntax, Syntax |
|------------|------------|----------------|

| 17) An exception is | 18) How many keyword | 19) How many except |
|---------------------|-------------------------------|-----------------------------|
| | arguments can be passed to a | statements can a try-except |
| | function in a single function | block have? |
| | call? | |
| | | |
| | | |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| | | |
| a module | one or more | more than one |
| | | |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| | | |
| | zero or more | more than one |
| | | |
| | | |
| | | |
| an object | zoro or moro | more than one |
| | | |
| | | |
| | | |
| | | |
| a special function | one | more than one |
| | | |
| | | |
| | | |
| | | |
| an object | one or more | more than one |
| | | |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |

| an object | zero or more | more than zero |
|-----------|--------------|----------------|
| | | |
| an object | one or more | more than one |
| an object | one or more | more than one |
| | | |
| en ekiest | | |
| an object | one or more | more than one |
| an object | zero or more | more than zero |
| | | |
| an object | zero or more | more than zero |
| an abient | | |
| an object | zero or more | more than zero |
| | | |
| an object | zero | 1 |
| | | |
| an object | zero | 1 |
| on chiest | ono or more | more then end |
| an object | | |

| an object | zero or more | more than zero |
|-----------|--------------|----------------|
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| an object | zero | 1 |
| | | |
| | | |
| | | |
| an object | zero | 1 |
| | | |
| | | |
| | | |
| an object | one | more than one |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| an object | one or more | 1 |
| | | |
| | | |
| | | |
| an object | zero or more | 1 |
| | | |
| | | |
| an abiaat | Joro or moro | |
| | | 1 |
| | | |
| | | |
| an object | zero | 4 |
| | 2010 | |

| an object | zero | 1 |
|--------------------|--------------|----------------|
| | | |
| | | |
| an object | zero | 1 |
| | | |
| | | |
| an obiect | zero | more than one |
| | | |
| | | |
| an abiant | | |
| an object | one | more than one |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| an object | zero or more | more than one |
| | | |
| | | |
| a special function | zoro or moro | 1 |
| | | |
| | | |
| | | |
| a standard module | zero or more | 1 |
| | | |
| | | |
| a special function | zero or more | 1 |

| on object | | more then zero |
|-------------------------|--------------|----------------|
| | | more than zero |
| | | |
| a an a cial from sticos | | |
| | | |
| | | |
| | | |
| an object | zero or more | more than one |
| | | |
| a standard module | | mare then and |
| | | more than one |
| | | |
| | | |
| | one | I |
| | | |
| a special function | zero or more | 1 |
| | | 1 |
| | | |
| an object | zero | more than zero |
| | | |
| | | |
| an object | zero or more | more than one |
| | | |
| | | |
| a standard module | zero or more | more than one |
| | | |
| | | |
| an object | zero or more | more than zero |
| | - | |

| an object | zero or more | more than zero |
|--------------------|--------------|----------------|
| | | |
| | | |
| | | |
| an object | zero or more | more than one |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| an abiaat | zoro or moro | more then zero |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |
| | | |
| | | |
| a special function | one | more than one |
| | | |
| | | |
| | | |
| a module | zero or more | 1 |
| | | |
| | | |
| an object | zero or more | more than zero |
| | | |

| a special function | one or more | more than one |
|--------------------|-------------|---------------|

| 20) Which of the follo | owing | 21) Which operator is used in | 22) How is a function |
|----------------------------|------------|-------------------------------|-----------------------|
| will print the pi value of | defined | Python to import modules | declared in Python? |
| in math module? | | from packages? | - |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| frame weath improve the | n rint(ni) | | deffunction nemo(). |
| nom main import pr | print(pr) | | |
| | | | |
| | | | |
| | | | |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| | | | |
| | | | |
| | | | |
| print(math.pi) | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| from math import ni | print(pi) | | def function name(): |
| nom maar import pr | print(pi) | • | |
| | | | |
| | | | |
| | | | |
| | | | |
| print(math.pi) | | | def function_name(): |
| | | | |
| | | | |
| | | | |
| | | | |
| from math import pi | print(pi) | | def function name(): |
| | 1 (17 | | |
| | | | |
| | | | |
| | | | |
| from moth import ni | print(pi) | * | def function name(): |
| nom maur import pr | print(pr) | | |
| | | | |
| | | | |
| | | | |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| | | | |
| | | | |
| | | | |
| from math import pi | print(pi) | | def function_name(): |

| from math import pi | print(pi) | | def function_name(): |
|---------------------|-----------|----|----------------------|
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| print(math.pi) | | -> | def function_name(): |
| | | | |
| print(math.pi) | | -> | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| print(pi) | | | def function_name(): |
| | | | |
| print(pi) | | -> | |
| | | | |
| from math import pi | print(pi) | | def function_name(): |

| from math import pi | print(pi) | def function_name(): |
|---------------------|-----------|----------------------|
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| print(pi) | | def function_name(): |
| | | |
| print(pi) | | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| print(math.pi) | | def function_name(): |
| print(pi) | | | def function_name(): |
|---------------------------------------|-----------|---|----------------------|
| | | | |
| print(pi) | | | def function_name(): |
| | | | |
| print(pi) | | | def function_name(): |
| | | | |
| print(pi) | | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | • | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | * | def function_name(): |
| | | | |
| from math import pi print(math.pi) | | * | def function_name(): |
| | | | |
| from math import pi print(math.pi) | | * | def function_name(): |

| from math import pi print(math.pi) | | | def function_name(): |
|---------------------------------------|-----------|----|----------------------|
| | | | |
| from math import pi print(math.pi) | | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi print(math.pi) | | -> | def function_name(): |
| from moth import ni | | | |
| print(math.pi) | | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |
| | | | declare function |
| from math import pi | print(pi) | * | function_name(): |
| | | | |
| from math import pi | print(pi) | | def function_name(): |

| from math import pi | print(pi) | def function_name(): |
|---------------------|-----------|----------------------|
| | | |
| print(math.pi) | | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |
| | | |
| from math import pi | print(pi) | def function_name(): |

| from math import pi | print(pi) . | def function_name(): |
|---------------------|-------------|----------------------|

| 23) Which one of the | 24) Choose the correct option | 25) Which one of the |
|---------------------------------|-------------------------------|------------------------------|
| following is the correct way of | with reference to below | following is incorrect? |
| calling a function? | Python code? | Ũ |
| 5 | def fn(a): | |
| | | |
| | | The least veriables of a |
| | | |
| | | particular function can be |
| | | used inside other functions, |
| | | but these cannot be used in |
| function_name() | x is the actual argument. | global space |
| | | The local variables of a |
| | | particular function can be |
| | | used inside other functions, |
| | | but these cannot be used in |
| function_name() | x is the actual argument. | global space |
| | | |
| | | In order to change the value |
| | | of global variable inside |
| | | function keyword global is |
| function name() | x is the formal argument | used |
| | k is the formal argument. | The least veriables of a |
| | | nerticular function con he |
| | | particular function can be |
| | | used inside other functions, |
| | | but these cannot be used in |
| function_name() | x is the actual argument. | global space |
| | | The local variables of a |
| | | particular function can be |
| | | used inside other functions, |
| | | but these cannot be used in |
| function_name() | x is the actual argument. | global space |
| | | |
| | | In order to change the value |
| | | of global variable inside |
| | | function keyword global is |
| function_name() | x is the actual argument | used |
| | | The local variables of a |
| | | narticular function can be |
| | | used inside other functions |
| | | but these connet he wood in |
| not firm at an anomal () | | put triese cannot de used in |
| ret function_name() | a is the actual argument. | giobal space |
| | | The local variables of a |
| | | particular function can be |
| | | used inside other functions, |
| | | but these cannot be used in |
| function_name() | x is the actual argument. | global space |
| | | The local variables of a |
| | | particular function can be |
| | | used inside other functions. |
| | | but these cannot be used in |
| function name() | x is the actual argument. | global space |
| — V | | |

| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
|-----------------|---------------------------|---|
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |

| function name() | v is the actual argument | The local variables of a particular function can be used inside other functions, but these cannot be used in |
|---------------------|----------------------------------|---|
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| ret function_name() | fn(x) is the function signature. | In order to change the value of global variable inside function, keyword global is used. |
| function name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | fn(x) is the function signature. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| ret function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |

| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
|---------------------|---------------------------|---|
| ret function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | a is the actual argument. | The variables used outside function are called global variables |
| function_name() | a is the actual argument. | The variables used outside function are called global variables |

| | | In order to change the value of global variable inside function, keyword global is |
|--------------------------|----------------------------------|---|
| function_name() | x is the formal argument. | used. |
| function_name() | x is the actual argument. | The variables used inside function are called local variables. |
| function_name() | fn(x) is the function signature. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function function_name() | x is the actual argument. | In order to change the value of global variable inside function, keyword global is used. |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| call function_name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |

| | - | |
|--------------------------|---------------------------|---|
| function name() | x is the actual argument | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function function_name() | x is the formal argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual aroument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |
| function_name() | | In order to change the value of global variable inside function, keyword global is used. |
| function_name() | x is the actual argument. | The local variables of a particular function can be used inside other functions, but these cannot be used in global space |

| | | The variables used inside |
|-----------------|---------------------------|---------------------------|
| | | function are called local |
| function_name() | x is the formal argument. | variables. |