

MTA Document- 4

Microsoft Python Exam Preparation

Mr. Yogesh P Murumkar

Bharat Soft Solutions

Machine Learning Data Science and Big Data

Youtube – <https://www.youtube.com/c/yogeshmurumkar>

1

Which of the following is False?

- ☐ A try statement can have one or more except clauses
- ☐ A try statement can have a finally clause without an except clause
- ☐ A try statement can have a finally clause and an except clause
- ☒ A try statement can have one or more finally clauses **(Correct)**

Explanation

Every try statement should be associated with atmost one finally block.i.e we cannot take more than one finally block for the same try.

2

Which type of exception will be raised if we are trying to call a method on the inappropriate object?

☒ IndexError (Incorrect)

☐ TypeError

☐ AttributeError (Correct)

☐ None of these

Explanation

If we are trying to access a method on the object, if the corresponding class does not contain that method, then we will get AttributeError.

eg:

```
l=[10,20,30]
```

```
l.add(30)
```

```
AttributeError: 'list' object has no attribute 'add'
```

Consider the code

```
f=open('abc.txt')  
f.readall()
```

Which exception will be raised?

☒ AttributeError (Correct)

☐ EOFError

☐ SystemError

☐ SyntaxError (Incorrect)

Explanation

If we are trying to access a method on the object, if the corresponding class does not contain that method, then we will get AttributeError.

readall() method is not available for file object.

AttributeError: '_io.TextIOWrapper' object has no attribute 'readall'

Consider the code

```
1 def f1():
2     try:
3         return 1
4     finally:
5         return 2
6
7 x=f1()
8 print(x)
```

What is the result?

☒ 1 (Incorrect)

☐ 2 (Correct)

☐ prints both 1 and 2

☐ Error, because more than one return statement is not allowed

Explanation

finally block return statement has more priority than try block return statement

5

Question 5: **Incorrect**

Which of the following are True?

☐ A try block can have any number of except blocks

☒ A try block should be associated with atmost one finally block (Incorrect)

☐ A try block should be associated with atmost one else block

☐ All the above (Correct)

Explanation

We can take any number of except blocks for the same try.
We can't take more than one finally blocks for the same try.
We can't take more than one else blocks for the same try.

Question 6: **Incorrect**

The base class for all exceptions in python is:

☐ ExceptionBase

☐ BaseException **(Correct)**

☒ Exception **(Incorrect)**

☐ EOFError

Explanation

Every Exception class in python should be child class of BaseException either directly or indirectly.

Which of the following is True about else block?

☐ else block will be executed if there is no exception in try block

☐ Without writing except block we cannot write else block

☒ For the same try we can write atmost one else block **(Incorrect)**

☐ All the above **(Correct)**

Explanation

else block will be executed if there is no exception in try block

Without writing except block we cannot write else block

For the same try we can write atmost one else block.i.e more than one else block we cannot take.

Consider the code:

```
1 try:
2     print('try')
3 except:
4     print('except')
5 else:
6     print('else')
7 finally:
8     print('finally')
```

What is the result?

☐ 1 try
2 except
3 else
4 finally

☒ 1 try
2 else
3 finally (Correct)

☐ 1 try
2 except
3 finally

☐ try
finally

Explanation

except block will be executed if there is an exception in try block.

else block will be executed if there is no exception in try block.

finally block will be executed always whether exception raised or not raised and whether handled or not handled.

Consider the code

```
1 | try:
2 |     print('try')
3 |     print(10/0)
4 | except:
5 |     print('except')
6 | else:
7 |     print('else')
8 | finally:
9 |     print('finally')
```

What is the result?

☐ 1 | try
2 | except
3 | else
4 | finally

☐ 1 | try
2 | else
3 | finally

☒ 1 | try
2 | except
3 | finally (Correct)

☐ 1 | try
2 | finally

Explanation

except block will be executed if there is an exception in try block.

else block will be executed if there is no exception in try block.

finally block will be executed always whether exception raised or not raised and whether handled or not handled.

Consider the code :

```
1 | try:
2 |     print('try')
3 |     print(10/0)
4 |
5 | else:
6 |     print('else')
7 |
8 | except:
9 |     print('except')
10 | finally:
11 |     print('finally')
```

What is Result?

☐ 1 | try
2 | else
3 | except
4 | finally

☐ 1 | try
2 | else
3 | finally

☐ 1 | try
2 | except
3 | finally

☒ `SyntaxError: invalid syntax` (Correct)

Explanation

in try-except-else-finally the order is important.
After except block only we have to take else block

Consider the code:

```
1 f=open('abc.txt')
2 print(f.read())
3 f.close()
```

We required to add exception handling code to handle FileNotFoundError.

We of the following is appropriate code for this requirement?

```
1 f=None
2 try:
3     f=open('abc.txt')
4 except FileNotFoundError:
5     print('File does not exist')
6 else:
7     print(f.read())
8 finally:
9     if f != None:
10        f.close()
```

(Correct)

```
1 f=None
2 try:
3     f=open('abc.txt')
4 except FileNotFoundError:
5     print('File does not exist')
6 else:
7     print(f.read())
8 finally:
9     if f != None:
10        f.close()
```

☐ 3 f=open('abc.txt')

4 except FileNotFoundError:

5 print('File does not exist')

6 else:

7 print(f.read())

8 finally:

9 if f != None:

10 f.close()

1 f=None

2 try:

3 f=open('abc.txt')

4 else:

5 print(f.read())

6 except FileNotFoundError:

7 print('File does not exist')

8 finally:

9 if f != None:

10 f.close()

11

12

☐ None of these

Explanation

In Python we have FileNotFoundError but not FileNotFoundExpection.
In try-except-else-finally, order is important. We cannot take else block before except block.

Question 12: Skipped

Consider the code

```
1 a=10
2 b=20
3 c='30'
4 result=a+b+c
```

What is the result?

☐ 102030

☐ 3030

☒ TypeError (Correct)

☐ ArithmeticError

Explanation

We cannot apply + operator for 'int' and 'str' arguments. Otherwise we will get
TypeError: unsupported operand type(s) for +: 'int' and 'str'

Consider the code:

```
1 prices=[30.5,'40.5',10.5]
2 total=0
3 for price in prices:
4     total += price
5 print(total)
6
7 While executing this code we are getting the following error
8
9 Traceback (most recent call last):
10   File "test.py", line 4, in <module>
11     total += price
12   TypeError: unsupported operand type(s) for +=: 'float' and 'str'
```

Which of the following code should be used to fix this error?

☐ `total += str(price)`

☐ `total += int(price)`

☒ `total += float(price)` (Correct)

☐ `total = total+price`

Explanation

If the string contains internally float value, then we should use `float()` function to convert into float value. If we are trying to use `int()` function then we will get `ValueError`. Hence the following line is the correct fix for the problem
`total += float(price)`

Consider the code:

```
1 prices=[10, '20', 30, '40']
2 total=0
3 for price in prices:
4     total += price
5 print(total)
6
7 While executing this code we are getting the following error:
8
9 Traceback (most recent call last):
10   File "test.py", line 4, in <module>
11     total += price
12 TypeError: unsupported operand type(s) for +=: 'int' and 'str' total += str(price)
```

By using which of the following code segments we can fix this problem(Choose Two)?

☐ total += str(price)

☒ total += int(price) (Correct)

☒ total += float(price) (Correct)

☐ total = total+price

Explanation

We cannot apply + operator between int and str. Hence We have to type cast str to either int type or float type, then only we can apply + operator.

Consider the code

```
1 | courses={1:'Java',2:'Scala',3:'Python'}
2 | for i in range(1,5):
3 |     print(courses[i])
4 |
5 | While executing this code we are getting the following error
6 |
7 | Traceback (most recent call last):
8 |   File "test.py", line 3, in <module>
9 |     print(courses[i])
10 |  KeyError: 4
```

By using which of the following code segments we can fix this problem ?

☐

```
1 | courses={1:'Java',2:'Scala',3:'Python'}
2 | for i in range(1,5):
3 |     if i in courses:
4 |         print(courses[i])
```

☒

```
1 | courses={1:'Java',2:'Scala',3:'Python'}
2 | for i in courses:
3 |     print(courses[i])
```

 (Incorrect)

☐

```
1 | courses={1:'Java',2:'Scala',3:'Python'}
2 | for i in range(1,4):
3 |     print(courses[i])
```

☐ All of these (Correct)

Explanation

In the above code key 4 is not available in the dictionary. Hence we are getting KeyError. If the key is within the range from 1 to 3, then the problem will be fixed.

Consider the code

```
1 def area(b,w):  
2     return B*w  
3     print(area(10,20))
```

What is the result?

☒ NameError will be raised at runtime (Correct)

☐ AttributeError will be raised at runtime (Incorrect)

☐ IndentationError will be raised at runtime

☐ 200

Explanation

If the variable is not available, still if we are trying to access then we will get NameError. In the above code variable 'B' is not available.

17

Consider the following code:

```
1 def get_score(total=0,valid=0):  
2     result=int(valid)/int(total)  
3     return result
```

For which of the function calls we will get Error?

☐ score=get_score(40,4)

☒ score=get_score('40','4') (Incorrect)

☐ score=get_score(40)

☐ score=get_score(0,10) (Correct)

Explanation

We are performing division by zero and hence we will get ZeroDivisionError: division by zero

Consider the code:

```
1 data=[]
2 def get_data():
3     for i in range(1,5):
4         marks=input('Enter Marks:')
5         data.append(marks)
6
7 def get_avg():
8     sum=0
9     for mark in data:
10        sum += mark
11    return sum/len(data)
12 get_data()
13 print(get_avg())
```

For the input: 10,20,30,40 what is the result?

☐ 25

☒ 25.0 (Incorrect)

☐ NameError is thrown at runtime

☐ TypeError is thrown at runtime (Correct)

Explanation

All marks are available in string form and hence we cannot apply + operator between int and str `sum += mark` `TypeError: unsupported operand type(s) for +=: 'int' and 'str'`

Consider the code:

```
1 a=18
2 b=0
3 try:
4     print(a/b)
```

Which of the following except block print the name of exception which is raised,i.e exception class name?

☐ `except ZeroDivisionError as e:`
`print('Exception Type:',e.__class__.__name__)` (Correct)

☒ `except ZeroDivisionError as e:`
`print('Exception Type:',type(e).__name__)` (Correct)

☐ `except ZeroDivisionError as e:`
`print('Exception Type:',e)`

☐ All of these

Explanation

We can get class name from the exception object as follows

`e.__class__.__name__` or

`type(e).__name__`

If we use just `print(e)` then we will get description of the exception like: 'division by zero'

Which of the following is valid way of creating our own custom exception?

- ☐

```
1 class MyException:
2     pass
```
- ☒

```
1 class MyException():
2     pass
```

 (Incorrect)
- ☐

```
1 class MyException(Exception):
2     pass
```

 (Correct)
- ☐ It is not possible to define custom exceptions in python

Explanation

To define custom exceptions, compulsory we have to create child class for BaseException either directly or indirectly

Consider the code

```
1 x=int(input('Enter First Number:'))
2 y=int(input('Enter Second Number:'))
3 try:
4     print(x/y)
```

Which of the following is valid except block that handles both ZeroDivisionError and ValueError

☐

```
1 except(ZeroDivisionError,ValueError) from e:
2     print(e)
```

☒

```
1 except(ZeroDivisionError,ValueError) as e:
2     print(e)
```

 (Correct)

☐

```
1 except(ZeroDivisionError | ValueError) as e:
2     print(e)
```

☐

```
1 except(ZeroDivisionError, ValueError as e):
2     print(e)
```

 (Incorrect)

Explanation

The following is the valid syntax:

```
except(ZeroDivisionError,ValueError) as e:
    print(e)
```

We should use as keyword only but not from keyword. The variable e must be outside of parenthesis because it is common for both exceptions.

Consider the following code.

```
1 import os
2 def get_data(filename,node):
3     if os.path.isfile(filename):
4         with open(filename, 'r') as file:
5             return file.readline()
6     else:
7         return None
```

Which of the following are valid about this code?

☐ This function returns the first line of the file if it is available (Correct)

☒ This function returns None if the file does not exist (Correct)

☐ This function returns total data present in the file

☐ This function returns last line of the file

Explanation

This function returns None if the file does not exist. If the file exists, then the function must return the first line. `os.path.isfile(filename)` can be used to check whether the given file exists or not. If it exists returns True otherwise returns False.

We are writing a Python program for the following requirements:

Each line of the file must be read and printed

if the blank line encountered, it must be ignored

When all lines have been read, the file must be closed.

Consider the code:

```
1 inventory=open('inventory.txt','r')
2 eof=False
3 while eof == False:
4     line=inventory.readline()
5     if XXX:
6         if YYY:
7             print(line,end='')
8
9     else:
10        print('End of file')
11        eof=True
12        inventory.close()
```

Which of the following changes are required to perform to meet the requirements

☒

```
1 XXX should be replaced with
2 line != ''
3 YYY should be replaced with
4 line!= '\n'
```

 (Correct)

☐

```
1 XXX should be replaced with
2 line!= '\n'
3 YYY should be replaced with
4 line != ''
```

```
1 XXX should be replaced with
2 line != ''
3 YYY should be replaced with
4 line != ''
```

```
1 XXX should be replaced with
2 line!= '\n'
3 YYY should be replaced with
4 line!= '\n'
```

Explanation

\n represents blank line and if end of file then readline() method returns empty string. Hence
XXX should be replaced with
line != ""
YYY should be replaced with
line!= '\n'

24

You develop a python application for your school.

You need to read and write data to a text file. If the file does not exist, it must be created. If the file has the content, the content must be removed.

Which code we have to use?

- ☐ `open('abc.txt', 'r')`
- ☐ `open('abc.txt', 'r+')`
- ☒ `open('abc.txt', 'w+')` (Correct)
- ☐ `open('abc.txt', 'w')`

Explanation

In the case of 'r+', if file exist it will be opened in read and write mode and if file is not available, it doesn't create a new file.

25

We are creating a function that reads a data file and prints each line of that file. Consider the following code:

```
1 import os
2 def read_file(file):
3     line=None
4     if os.path.isfile(file):
5         data=open(file,'r')
6     while line != '':
7         line=data.readline()
8     print(line)
```

The code attempts to read the file even if the file does not exist.

You need to correct the code. which lines having indentation problems?

☐ First 3 Lines inside function

☒ Last 3 Lines inside function (Correct)

☐ Last 2 Lines inside function

☐ There is no indentation problem

Explanation

The Last 3 lines having indentation problem and the correct code is:

```
import os
def read_file(file):
line=None
if os.path.isfile(file):
data=open(file,'r')
while line != '':
line=data.readline()
print(line)
```

Consider the code:

```
1 import sys
2 try:
3     file_in=open('in.txt','r')
4     file_out=open('out.txt','w+')
5 except IOError:
6     print('cannot open',file_name)
7 else:
8     i=1
9     for line in file_in:
10         print(line.rstrip())
11         file_out.write(str(i)+":"+line)
12         i=i+1
13     file_in.close()
14     file_out.close()
```

Assume that in.txt file is available but out.txt file does not exist.

Which of the following is true about this code?

☒ This program will copy data from in.txt to out.txt (Correct)

☐ The code runs,but generates logical error

☐ The code will generates a runtime error

☐ The code will generates a syntax error

Explanation

It is valid code and it is reading total data from the in.txt and writing to out.txt.

Consider the file abc.txt:

abc.txt:

```
1 | Durga:10
2 | Ravi:20
3 | Shiva:30
4 | Pavan:40
```

Consider the python code which is present in the same location of the file

test.py:

```
1 | values=0
2 | try:
3 |     f=open('abc.txt','r')
4 |     content=f.readlines()
5 |     for line in content:
6 |         values+=float(line.split(':')[1])
7 |     f.close()
8 | except Exception:
9 |     print('Unable to open the file')
10 | print(values)
```

What is the result?

☐ Unable to open the file

☐ 100

☒ 100.0 (Correct)

☐ 10.0

Explanation

The above program reads all the numbers present in every line of file and converting to float value and then adding that value to value variable.

Consider the file `abc.txt` has the following content:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

We have to write python code to read total data and print to the console.

```
1 try:
2     f=open('abc.txt','r')
3     //line-1
4 except:
5     print('Unable to open the file')
6 print(data)
```

Which code should be inserted at Line-1 to meet the given requirement ?

☒ `data=f.read()` (Correct)

☐ `data=f.readline()` (Incorrect)

☐ `data=f.readlines()`

☐ `data=f.load()`

Explanation

To read total data, we have to use `f.read()` method.

To write 'Python Certificaton' to abc.txt file, which of the following is valid code?

☐

```
1 f=open('abc.txt','b')
2 f.write('Python Certificaton')
3 f.close()
```

☒

```
1 f=open('abc.txt','r')
2 f.write('Python Certificaton')
3 f.close()
```

 (Incorrect)

☐

```
1 f=open('abc.txt')
2 f.write('Python Certificaton')
3 f.close()
```

☐

```
1 f=open('abc.txt','w')
2 f.write('Python Certificaton')
3 f.close()
```

 (Correct)

Explanation

To write text data we have to open the file in 'w' mode

30

Consider the data present in the file: abc.txt

BHARATSOFT,50,60,70,80,90

MICROSOFT,10,20,30,40,50

Which of the following is valid code to read total data from the file?

☐

```
1 with open('abc.txt', 'r') f:  
2     data=f.read()
```

☒

```
1 with open('abc.txt') as f:  
2     data=f.read()
```

 (Correct)

☐

```
1 with open('abc.txt', 'w') as f:  
2     data=f.read()
```

☐

```
1 with open('abc.txt') as f:  
2     data=f.readline()
```

Explanation

The default mode is 'r' and hence the following is valid syntax:

with open('abc.txt') as f:

data=f.read()

Assume that we are writing python code for some voting application.

You need to open the file voters_list.txt and add new voters info and print total data to the console?

```
1 with open('voters_list.txt','a+') as f:
2     f.write('New voters info')
3     #Line-1
4     data=f.read()
5     print(data)
```

Which Line should be inserted at Line-1 ?

☒ `f.seek(0)` (Correct)

☐ `f.flush()`

☐ `f.begin()`

☐ `f.close()`

Explanation

After writing the data, to read total data we have to move to beginning of the file. Hence we should use `f.seek(0)`

You are creating a function that manipulates a number. The function has the following requirements:

A float passed to the function

The function must take absolute value of the float

Any decimal points after the integer must be removed.

Which two math functions should be used?

☐ `math.frexp(x)`

☒ `math.floor(x)` (Correct)

☐ `math.fabs(x)` (Correct)

☐ `math.fmod(x)`

☒ `math.ceil(x)` (Incorrect)

Explanation

`fabs(x)` Returns the absolute value of `x`

`floor(x)` Returns the largest integer less than or equal to `x`

Hence the following line will perform the required operation:

```
print(floor(fabs(-123.456)))
```

You are writing an application that uses the `sqrt` function. The program must reference the function using the name `sq`.

Which of the following import statement required to use?

☐ `import math.sqrt as sq`

☒ `import sqrt from math as sq` (Incorrect)

☒ `from math import sqrt as sq` (Correct)

☐ `from math.sqrt as sq`

Explanation

The following is the valid syntax: `from math import sqrt as sq`

34

Consider the code:

```
1 import math
2 l=[str(round(math.pi)) for i in range (1, 6)]
3 print(l)
```

What is the result?

☒ `['3', '3', '3', '3', '3']` (Correct)

☐ `['3', '3', '3', '3', '3','3']` (Incorrect)

☐ `['1', '2', '3', '4', '5']`

☐ `['1', '2', '3', '4', '5','6']`

Explanation

`range(1,6)` means we have to consider from 1 to 5 and every time we are performing `round(math.pi)` and hence the output is: `['3', '3', '3', '3', '3']`

35

You are writing code that generates a random integer with a minimum value of 5 and maximum value of 11. Which of the following 2 functions we required to use?

☐ `random.randint(5,12)`

☒ `random.randint(5,11)` (Correct)

☒ `random.randrange(5,12,1)` (Correct)

☐ `random.randrange(5,11,1)`

Explanation

1. `randint(begin,end)` generates a random int value between given 2 numbers(boundaries are inclusive)
2. `randrange([start],stop,[step])` returns a random number from the range $start \leq x$

36

Consider the following code:

```
1 import random
2 fruits=['Apple','Mango','Orange','Lemon']
```

Which of the following will print some random value from the list?

☐ `print(random.sample(fruits))`

☒ `print(random.sample(fruits,3)[0])` (Correct)

☒ `print(random.choice(fruits))` (Correct)

☐ `print(random.choice(fruits)[0])`

Explanation

`random.sample(population, k)`

Return a k length list of unique elements chosen from the population sequence or set.

We are developing an application for the client requirement. As the part of that we have to create a list of 7 random integers between 1 and 7 inclusive.

Which of the following code should be used?

☐ `import random`
`randints=[random.randint(1,7) for i in range(1,8)]` (Correct)

☒ `import random`
`randints=[random.randint(1,8) for i in range(1,8)]` (Incorrect)

☐ `import random`
`randints=random.randrange(1,7)`

☐ `import random`
`randints=random.randint(1,7)`

Explanation

`randints=[random.randint(1,8) for i in range(1,8)]`

In this case there may be a chance of 8 also.

`randints=random.randrange(1,7)`

It will generate a random number from 1 to 6 but not list.

`randints=random.randint(1,7)`

it generates a random number from 1 to 7 but not list

Consider the code:

```
1 import random
2 fruits=['Apple','Mango','Orange','Lemon']
3 random_list=[random.choice(fruits)[:2] for i in range(3)]
4 print("".join(random_list))
```

Which of the following are possible outputs?

☐ ApApAp (Correct)

☒ ApMaOr (Correct)

☐ LeMaOr (Correct)

☐ OrOrAM

Explanation

The above code selects 3 random words from the list and from every word first 2 characters will be collected and joined into a single string.

Hence the following outputs are possible: ApApAp, ApMaOr, LeMaOr

We are developing a mathematical function to find area for the given circle, if r is the radius then area is : $\pi * r^2$

Which of the following is valid function for this requirement?

☐

```
import math
def find_area(r):
    return math.pi*math.fmod(r,2)
```

☐

```
import math
def find_area(r):
    return math.pi*math.fabs(r)
```

☒

```
import math
def find_area(r):
    return math.pi*math.pow(r,2)
```

 (Correct)

☐ None of these

Explanation

`fmod(x, y)` Returns the remainder when x is divided by y

`fabs(x)` Returns the absolute value of x

`pow(x, y)` Returns x raised to the power y

Consider the python code:

```
1 import random
2 print(int(random.random()*5))
```

Which of the following is true?

☐ It will print a random int value from 0 to 5

☐ It will print a random int value from 1 to 5

☒ It will print a random int value from 0 to 5 (Incorrect)

☐ It will print a random int value from 0 to 4 (Correct)

☐ It will print 5

Explanation

random() function will generate a random float value which is >0 and <1 .

random()*5 generates a float value which is >0 and <5 .

Hence `int(random.random()*5)` will print a random int value from 0 to 4.