Print Exit Print Mode

About the exam

Dear Student,

Greetings!
You have completed the "Final assessment" exam.
At this juncture, it is important for you to understand your strengths and focus on them to achieve the best results.
We present here a snapshot of your performance in "Final assessment" exam in terms of marks scored by you in each section, question-wise response pattern and difficulty-wise analysis of your performance.

This Report consists of the following sections that can be accessed using the left navigation panel:

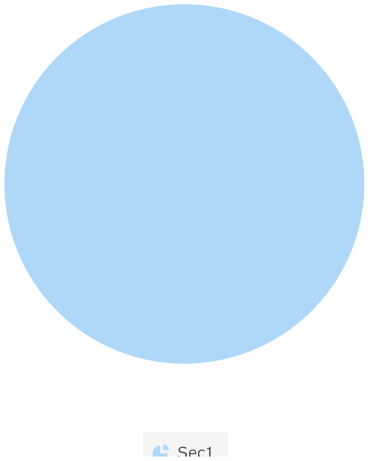
- **Overall Performance:** This part of report shows the summary of marks scored by you across all sections of the exam and the comparison of your performance across all sections.

- **Section-wise Performance:** You can click on a section name in the left navigation panel to check your performance in that section. Section-wise performance includes the details of your response at each question level and difficulty-wise analysis of your performance for that section.

NOTE : For Short Answer, Subjective, Typing and Programing Type Questions stidents will not be able to view Bar Chart Report in the Performance Analysis.

Section	Questions Attempted	Correct	Score
Sec1	40/40	11	26

Marks Obtained Subject Wise



NOTE : Subject having negative marks are not considered in the pie chart. Pie chart will not be shown if all the subject contains 0 marks.

Sec1

The Sec1 section comprises of a total of 40 questions with the following difficulty level distribution: -

Difficulty Level	No. of questions
EASY	18
MODERATE	15
HARD	7

Question wise details

Please click on question to view detailed analysis

✓ = Correct Option ☞ = Your Option ⚑ = Not Evaluated 📌 = Evaluated ✔ = Correct ⊗ = Incorrect ❗ = Not Attempted

Question Details

⊗ Q1.Which of the following statement creates a dictionary whose keys are elements of list 'keys' and assoicted values are from list 'vals' ?

Difficulty Level : **Moderate**

Response : **3** Status : **Incorrect**

✓ Option 1 : dict(zip(keys, vals))
Option 2 : dict(keys, vals)
☞ Option 3 : { key:val for key in keys for val in vals }
Option 4 : {keys: vals}

⊗ Q2.Referring to sample code, which line contains an error, assuming z is defined ?

```
def func(**kw):
 x = 1,2,3       #line1
 a,b,c = 1,2,3   #line2
 y = z           #line3
 d,e = 1,2,3     #line4
```

Difficulty Level : **Easy**

Response : **1** Status : **Incorrect**

☞ Option 1 : line3
Option 2 : line2
✓ Option 3 : line4
Option 4 : line1

⊗ Q3.Which of the following modules warn about PEP 8 inconsistencies present in a python script ?

Difficulty Level : **Easy**

Response : **2,4** Status : **Incorrect**

Option 1 : pep8check
☞ ✓ Option 2 : pep8
✓ Option 3 : flake8
☞ Option 4 : pep8chk

⊗ Q4.What is the output of the following code ?

```
def multipliers():
 return [lambda x : i * x for i in range(4)]

print([m(2) for m in multipliers()])
```

Difficulty Level : **Hard**

Response : **3** Status : **Incorrect**

✓ Option 1 : [6,6,6,6]
Option 2 : [0,1,2,3]
☛ Option 3 : [0,2,4,6]
Option 4 : [0,0,0,0]

✓ Q5.Which of the following statement sets the metaclass of class A to B ?

Difficulty Level : **Moderate**
Response : **2** Status : **Correct**

Option 1 : class A(meta=B): pass
☛ ✓ Option 2 : class A: __metaclass__ = B
Option 3 : class A(metaclass=B): pass
Option 4 : class A: __metatype__ = M

✓ Q6.Which of the following is not a standard level for logging various events using 'logging' module ?

Difficulty Level : **Easy**
Response : **1** | Status : **Correct**

☛ ✓ Option 1 : LOG
Option 2 : INFO
Option 3 : CRITICAL
Option 4 : DEBUG

✗ Q7.Which methods are defined for an iterator class ?

Difficulty Level : **Easy**
Response : **1** Status : **Incorrect**

☛ Option 1 : iter, has_next, next
Option 2 : iter, next
✓ Option 3 : __iter__, __next__
Option 4 : __iter__, __has_next__, __next__

✗ Q8.What is the output of the following code ?

```
   def smart_divide(func):
    def wrapper(*args):
        a, b = args
        if b == 0:
            print('oops! cannot divide')
            return

        return func(*args)
    return wrapper


@smart_divide
def divide(a, b):
    return a / b

print(divide.__name__)
print(divide(4, 16))

print(divide(8,0))
```

Difficulty Level : **Hard**
Response : **1** | Status : **Incorrect**

☛ Option 1 :

```
wrapper
0.25
oops! cannot divide
```

Option 2 :

```
smart_divide
0.25
oops! cannot divide
None
```

✓ Option 3 :

```
wrapper
0.25
oops! cannot divide
None
smart_divide
```

Option 4 :
```
0.25
oops! cannot divide
```

✖ Q9.What is the output of the following code ?

```
def deprecated(func):
    def newFunc(*args, **kwargs):
        warnings.warn('Call to deprecated function {}'.format(func.__name__), ca
        return func(*args, **kwargs)
    return newFunc


@deprecated
def prod(x,y):
    'Returns product of two numbers'
    return x * y

print(prod(12, 12))
print(prod.__name__)
print(prod.__doc__)
```

Difficulty Level : **Hard**

Response : **4** Status : **Incorrect**

Option 1 :
```
144
deprecated
```

✓ Option 2 :
```
144
newFunc
None
```

Option 3 :
```
144
deprecated
None
```

☛ Option 4 :
```
144
newFunc
```

✖ Q10.What is the output of the following code ?

```
def foo(n):
    if (n < 3): yield 1
    else: return
    yield 2

n = 2
f = foo(n)
for i in range(n): print(f.__next__())

n = 5
f = foo(n)
for i in range(n): print(f.__next__())
```

Difficulty Level : **Moderate**

Response : **4** Status : **Incorrect**

✓ Option 1 :
```
1
2
StopIterationError
```

Option 2 :
```
1
StopIterationError
```

Option 3 :
```
1
2
3
```

☛ Option 4 :
```
1
2
```

✖ Q11.What is the output of the following code ?

```
class Person(object):
    def __init__(self, name):
        print("My name is ", name)

class Bob(Person):
    def __init__(self, name='Bob'):
        print('My name is Bob')

    def ClassID(self):
        print("I'm the father")
```

```
class Sue(Person):
    def __init__(self, name='Sue'):
        print('My name is Sue')

    def ClassID(self):
        print("I'm the mother")

class Child(Bob, Sue):
    def __init__(self, name='X'):
        super(Child, self).__init__(name)

    def ClassID(self):
        print("I'm the child")

Ann = Child('Ann')
Ann.ClassID()
```

Difficulty Level : **Easy** |

Response : **2**   Status : **Incorrect** |

Option 1 :
```
My name is Sue
I'm the child
```
☞ Option 2 :
```
My name is Ann
I'm the child
```
Option 3 :
```
My name is Ann
My name is Bob
I'm the child
```
✓ Option 4 :
```
My name is Bob
I'm the child
```

⊗ Q12.What is the output of the given statement ? '{0:$>2d} * {1:$>2d} = {2:$>2d}'.format(5, 10, 5*10)

Difficulty Level : **Moderate**

Response : **2** | Status : **Incorrect**

✓ Option 1 : $5 * 10 = 50

☞ Option 2 : $5 * $10 = $50

Option 3 : 5 * 10 = 50

Option 4 : 5 * $10 = 50

⊗ Q13.What is the output of the following code ?

```
def f1(a, b):
 f1.s = 'some value'
 return a+b

try:
  print(f1.s)
except Exception as e:
  print(str(e))

f1(3,4)

try:
  print(f1.s)
except Exception as e:
  print(str(e))
```

Difficulty Level : **Hard** |

Response : **3**   Status : **Incorrect** |

✓ Option 1 :
```
'function' object has no attribute 's'
some value
```
Option 2 :
```
some value
some value
```
☞ Option 3 :
```
'function' object has no attribute 's'
'function' object has no attribute 's'
```
Option 4 :
```
some value
'function' object has no attribute 's'
```

✔ Q14.Which of the following method is used by a user defined class to support '+' operator ?

Difficulty Level : **Easy**

Response : **1** | Status : **Correct**

☛ ✓ Option 1 : __add__()
Option 2 : plus()
Option 3 : __plus__()
Option 4 : add()

✖ Q15. Which of the following modules warn about common sources of errors present in a python script?

Difficulty Level : **Easy** |

Response : **1,2,4** Status : **Incorrect** |

☛ ✓ Option 1 : pyflakes
☛ Option 2 : pyerrors
✓ Option 3 : flake8
☛ Option 4 : pywarn

✖ Q16.If a property named 'temp' is defined in a class, which of the following decorator statement is required for setting the 'temp' attribute ?

Difficulty Level : **Moderate**

Response : **2** | Status : **Incorrect**

Option 1 : "@property.setter.temp"
☛ Option 2 : "@property.set.temp"
✓ Option 3 : "@temp.setter"
Option 4 : "@temp.set"

✖ Q17.Given the statement, d1 = dict(), which of the following statement is not valid for assigning a key-value pair to dictionary 'd1' ?

Difficulty Level : **Easy** |

Response : **2** Status : **Incorrect** |

Option 1 : d1[4+1] = 15
☛ Option 2 : d1[3] = 9
✓ Option 3 : d1 = {1, 4}
Option 4 : d1 = {1:4}

✖ Q18.What is the output of the following code ?

```
import logging

logging.warning('A Warning')
logging.info('A Info')
logging.error('An Error')
logging.debug('Debugging')
```

Difficulty Level : **Moderate**

Response : **1** | Status : **Incorrect**

☛ Option 1 :
```
WARNING:root:A Warning
DEBUG:root:Debugging
```
Option 2 :
```
WARNING:root:A Warning
INFO:root:An Info
```
Option 3 :
```
ERROR:root:An Error
DEBUG:root:Debugging
```
✓ Option 4 :
```
WARNING:root:A Warning
ERROR:root:An Error
```

✔ Q19.Which of the following code produces the below shown output ?

```
Base Created
Child Created
```

Difficulty Level : **Hard** |

Response : **2** Status : **Correct** |

```
class Base(object):
    def __init__(self):
        print('Base Created')

class Child(Base):
```

Option 1 :

```
class Child(Base):
    def __init__(self):
        super(Child, self).__init__()
        print('Child Created')

b = Base()
c = Child()
```

☛ ✓ Option 2 :

```
class Base(object):
    def __init__(self):
        print('Base Created')

class Child(Base):
    def __init__(self):
        super(Child).__init__()
        print('Child Created')

b = Base()
c = Child()
```

Option 3 :

```
class Base(object):
    def __init__(self):
        print('Base Created')

class Child(Base):
    def __init__(self):
        Child.__bases__[0].__init__(self)
        print('Child Created')

b = Base()
c = Child()
```

Option 4 :

```
class Base(object):
    def __init__(self):
        print('Base Created')

class Child(Base):
    def __init__(self):
        super().__init__()
        print('Child Created')

b = Base()
c = Child()
```

✓ Q20.Which of the following keyword is used for creating a method inside a class ?

Difficulty Level : **Easy**

Response : **1** | Status : **Correct**

☛ ✓ Option 1 : def
Option 2 : class
Option 3 : sub
Option 4 : method

✗ Q21.What does PEP stand for ?

Difficulty Level : **Easy** |

Response : **4** Status : **Incorrect** |

✓ Option 1 : Python enhancement Protocol
Option 2 : People enhancing Python
Option 3 : People empowering Python
☛ Option 4 : Python enlarging Protocol

✗ Q22.Which of the following statement retreives names of all builtin objects ?

Difficulty Level : **Easy**

Response : **2** | Status : **Incorrect**

Option 1 : import builtins; builtins.builtins_names
☛ Option 2 : import sys; sys.builtins_names
✓ Option 3 : import builtins; builtins.__dict__.keys()
Option 4 : import sys; sys.builtins.__dict__.keys()

✗ Q23.Which of the following modules help in checking performance of python code ?

Difficulty Level : **Easy** |

Response : **4** Status : **Incorrect** |

Option 1 : timecheck
Option 2 : performcheck
✓ Option 3 : timeit
☛ Option 4 : pcheck

✗ Q24.What is the output of the following code ?

```
class MyError(Exception):
 def __init__(self, value):
     self.value = value
 def __str__(self):
     return repr(self.value)

try:
    print('Hello World!')
    raise MyError('Oops something went wrong')
except MyError as e:
    print('Error Message :',e)
```

Difficulty Level : **Moderate**

Response : **3** | Status : **Incorrect**

✓ Option 1 :
```
Hello World!
Error Message : 'Oops something went wrong'
```
Option 2 :
```
Hello World!
```
☛ Option 3 :
```
Hello World!
System Error : Exiting
```
Option 4 :
```
Hello World!
'Oops something went wrong'
```

✓ Q25.Which of the following module is not used for parsing command line arguments automatically ?

Difficulty Level : **Moderate** |

Response : **2** Status : **Correct** |

Option 1 : getopt
☛ ✓ Option 2 : cmdparse
Option 3 : optparse
Option 4 : argparse

✗ Q26.Which methods are invoked on entring to and exiting from the block of code written in 'with' statement ?

Difficulty Level : **Easy**

Response : **2** | Status : **Incorrect**

Option 1 : __enter__(), __quit__()
☛ Option 2 : enter(), exit()
Option 3 : enter(), quit()
✓ Option 4 : __enter__(), __exit__()

✗ Q27.What is the type of variable 'a' defined as 'a = (5)' ?

Difficulty Level : **Moderate** |

Response : **2** Status : **Incorrect** |

Option 1 : tuple
☛ Option 2 : str
✓ Option 3 : int
Option 4 : list

✗ Q28.Which of the following statement retreives names of all builtin module names ?

Difficulty Level : **Easy**

Response : **4** | Status : **Incorrect**

✓ Option 1 : import sys; sys.builtin_module_names
Option 2 : import builtins; builtins.builtins_names
Option 3 : import builtins; builtins.module_names
☛ Option 4 : import sys; sys.builtins_names

Q29.Which code extracts the matched data from the object returned by f1 in the given

✗ Q29.Which code extracts the matched data from the object returned by f1 in the given sample code ?

```
import re

def f1(data):
    p = re.compile('(?P[A-Z]{2,3}) (?P[0-9]{3})')
    return p.search(data)
```

Difficulty Level : **Moderate**

Response : **3** Status : **Incorrect**

Option 1 :
```
obj = f1('CS 101')
dept, num = obj.get('dept'), obj.get('num')
```
✓ Option 2 :
```
obj = f1('CS 101')
dept, num = obj.group('dept'), obj.group('num')
```
☞ Option 3 :
```
obj = f1('CS 101')
dept, num = obj[0], obj[1]
```
Option 4 :
```
obj = f1('CS 101')
dept, num = obj['dept'], obj['num']
```

✓ Q30.If A and B are sets, which is a valid set operation

Difficulty Level : **Easy**

Response : **2** Status : **Correct**

Option 1 : A * B
☞ ✓ Option 2 : A ^ B
Option 3 : A ! B
Option 4 : A + B

✗ Q31.What is the output of the following code?

```
class class1:
    a = 1

    def f1():
        a = 2
        class1.a += 1
        print(class1.a)
        print(a)

class1.f1()
class1.f1()
```

Difficulty Level : **Moderate**

Response : **1** Status : **Incorrect**

☞ Option 1 :
```
2
3
3
2
```
Option 2 :
```
2
2
3
3
```
✓ Option 3 :
```
2
2
3
2
```
Option 4 :
```
2
2
2
2
```

✗ Q32.What is the output of the following code ?

```
class grandpa(object):
    pass

class father(grandpa):
    pass

class mother(object):
    pass

class child(mother, father):
    pass
    pass

print(child.__mro__)
```

Difficulty Level : **Moderate**

Response : **2** | Status : **Incorrect**

✓ Option 1 : ",,,,"

👉 Option 2 : ",,,,"

Option 3 : ",,,,"

Option 4 : ",,,,"

✗ Q33.Which of the following string can be assigned to format argument of basicConfig function, in logging module, inorder to view a level followed by a message in each line of log file ?

Difficulty Level : **Moderate** |

Response : **2** Status : **Incorrect** |

Option 1 : "%(levelname):%(message)"

👉 Option 2 : "%(level):%(message)"

Option 3 : "%(level)s:%(message)s"

✓ Option 4 : "%(levelname)s:%(message)s"

✗ Q34.Which of the following is true about decorators ?

Difficulty Level : **Easy**

Response : **3** | Status : **Incorrect**

Option 1 : Decorators always return None

Option 2 : A Decorator function is used only to format the output of another function

👉 Option 3 : dec keyword is used for decorating a function

✓ Option 4 : Decorators can be chained

✗ Q35.What is the output of the following code ?

```
class A: pass
class B(A): pass
class C(object): pass
class D(C): pass

a = A()
b = B()
c = C()
d = D()

print(isinstance(a, type(b)))
print(issubclass(C,C))
print(isinstance(d,D))
print(issubclass(C, (D,A,B,C)))
```

Difficulty Level : **Moderate** |

Response : **3** Status : **Incorrect** |

✓ Option 1 :

```
False
True
True
True
```

Option 2 :

```
False
True
True
False
```

👉 Option 3 :

```
False
False
True
Flase
```

Option 4 :

```
False
False
True
True
```

✓ Q36.What is the output of the following code ?

```
class MyType(type): pass

class SubType(MyType): pass
class MyObject(object):
    __metaclass__ = MyType

print(MyType.__class__)
print(SubType.__class__)
print(MyObject.__class__)
```

Difficulty Level : **Hard**

Response : **3** | Status : **Correct**

Option 1 :

Option 2 :

☛ ✓ Option 3 :

Option 4 :

✖ Q37.Which of the following functions correctly check if a given element is present atleast two times in a list and retun True?

```
def check_twice1(lst, elm):
    return lst.count(elm) > 1

def check_twice2(lst, elm):
    return (elm in lst and elm in lst[lst.index(elm)+1:])

def check_twice3(lst, elm):
    c = 0
    for x in lst:
        if x == elm: c += 1
    return c

def check_twice4(lst, elm):
    try:
        lst.remove(elm)
        lst.remove(elm)
    except:
        return False
    return True
```

Difficulty Level : **Hard** |

Response : **3** Status : **Incorrect** |

Option 1 : check_twice1, check_twice2

Option 2 : check_twice2

☛ Option 3 : check_twice1

✓ Option 4 : check_twice1, check_twice2, check_twice4

✔ Q38.Which of the keyword is used to display a customised error message to the user ?

Difficulty Level : **Easy**

Response : **3** | Status : **Correct**

Option 1 : error

Option 2 : except

☛ ✓ Option 3 : raise

Option 4 : yield

✔ Q39.Which of the following is true about property decorator ?

Difficulty Level : **Easy** |

Response : **1** Status : **Correct** |

☛ ✓ Option 1 : property decorator is used either for getting, setting or deleting an attribute

Option 2 : property decorator is used only for getting an attribute

Option 3 : property decorator is used only for setting an attribute

Option 4 : property decorator is used either for setting or getting a attribute.

✓ Q40.Which of the following expression does not create a tuple and assign it to variable 't1' ?

Difficulty Level : **Moderate**

Response : **4** | Status : **Correct**

	Option 1 :	t1 = (1, 2, 3)
	Option 2 :	t1 = ('a', 'b', 'c')
	Option 3 :	t1 = 1, 2, 3
👉 ✓	Option 4 :	t1 = tuple(1, 2, 3)