PRODDEC PYTHON DAY 7

pass keyword

```
# The pass statement is used as a placeholder for future code.
# When the pass statement is executed, nothing happens,
# but it is useful in cases where a statement is required syntactically.
# Many code is not allowed to begin, function definitions,
# class definitions, or if-elif-else clauses.

# Print the above paragraph comments.
if 1:
    pass
def myfunc():
    # Function with pass as only statement for forty function.
    pass
class MyClass:
    # Class with pass.
    pass
a = 33
b = 5
while b > 0:
    pass # If what used to be anything, if it is no even number.
    print(b)
    b = b - 1
```

Two Dimensional List

```
# Create and print the three dimensional list.
MULTIARRAYTWODIMENSIONALLIST = [[1,2,3,4,5,6,7,8,9,10,11,12],
                                [13,14,15,16,17,18,19,20,21,22,23,24],
                                [25,26,27,28,29,30,31,32,33,34,35,36]]
# Accessing each item in list
print(MULTIARRAYTWODIMENSIONALLIST[0][0])
print(MULTIARRAYTWODIMENSIONALLIST[0][1])
print(MULTIARRAYTWODIMENSIONALLIST[0][2])
print(MULTIARRAYTWODIMENSIONALLIST[0][3])
print(MULTIARRAYTWODIMENSIONALLIST[0][4])
print(MULTIARRAYTWODIMENSIONALLIST[0][5])
print(MULTIARRAYTWODIMENSIONALLIST[0][6])
print(MULTIARRAYTWODIMENSIONALLIST[0][7])
print(MULTIARRAYTWODIMENSIONALLIST[0][8])
print(MULTIARRAYTWODIMENSIONALLIST[0][9])
print(MULTIARRAYTWODIMENSIONALLIST[0][10])
print(MULTIARRAYTWODIMENSIONALLIST[0][11])
1
2
3
4
5
6
7
8
9
10
11
12
# Accessing each item in list of list of list
print(MULTIARRAYTWODIMENSIONALLIST[1][0])
print(MULTIARRAYTWODIMENSIONALLIST[1][1])
print(MULTIARRAYTWODIMENSIONALLIST[1][2])
13
14
15
16
17
18
19
20
21
22
23
24
# Accessing each item in list of list of list
print(MULTIARRAYTWODIMENSIONALLIST[2][0])
print(MULTIARRAYTWODIMENSIONALLIST[2][1])
print(MULTIARRAYTWODIMENSIONALLIST[2][2])
25
26
27
28
29
30
31
32
33
34
35
36
```

Three Dimensional List

```
# Create and print the three dimensional list.
MULTIARRAYTHREEDIMENSIONALLIST = [[[1,2,3,4,5,6,7,8,9,10,11,12],
                                  [13,14,15,16,17,18,19,20,21,22,23,24],
                                  [25,26,27,28,29,30,31,32,33,34,35,36]],
                                  [[37,38,39,40,41,42,43,44,45,46,47,48],
                                  [49,50,51,52,53,54,55,56,57,58,59,60],
                                  [61,62,63,64,65,66,67,68,69,70,71,72]]]
# Accessing each item in list
print(MULTIARRAYTHREEDIMENSIONALLIST[0][0][0])
print(MULTIARRAYTHREEDIMENSIONALLIST[0][0][1])
print(MULTIARRAYTHREEDIMENSIONALLIST[0][0][2])
1
2
3
# Accessing each item in list of list
print(MULTIARRAYTHREEDIMENSIONALLIST[0][1][0])
print(MULTIARRAYTHREEDIMENSIONALLIST[0][1][1])
print(MULTIARRAYTHREEDIMENSIONALLIST[0][1][2])
13
14
15
# Accessing the items in list of list of list
print(MULTIARRAYTHREEDIMENSIONALLIST[1][0][0])
print(MULTIARRAYTHREEDIMENSIONALLIST[1][0][1])
print(MULTIARRAYTHREEDIMENSIONALLIST[1][0][2])
37
38
39
```

Updating the values in the list

```
# Write the list of primes to print.
print(PRIMESUNDERTWENTY[0],"/primes/",PRIME[0],"/primes/")
2 /primes/ 2 /primes/
# Other numbers, another short and an alphabet case too.
print(PRIMESUNDERTWENTY[1],"/primes/",PRIME[1],"/primes/")
3 /primes/ 3 /primes/
print(PRIMESUNDERTWENTY[2],"/primes/",PRIME[2],"/primes/")
5 /primes/ 5 /primes/
print(PRIMESUNDERTWENTY[3],"/primes/",PRIME[3],"/primes/")
7 /primes/ 7 /primes/
print(PRIMESUNDERTWENTY[4],"/primes/",PRIME[4],"/primes/")
11 /primes/ 11 /primes/
print(PRIMESUNDERTWENTY[5],"/primes/",PRIME[5],"/primes/")
13 /primes/ 13 /primes/
print(PRIMESUNDERTWENTY[6],"/primes/",PRIME[6],"/primes/")
17 /primes/ 17 /primes/
print(PRIMESUNDERTWENTY[7],"/primes/",PRIME[7],"/primes/")
19 /primes/ 19 /primes/
```

Exception handling

```
# Python has many built-in exceptions that are raised when your
# program encounters an error (something in the program goes wrong).
When these exceptions occur, the Python interpreter stops the current
process and passes it on to the calling process until it is handled.
If not handled, the program will crash.

# Try and Except
try:
    # In Python, exceptions can be handled using a try statement.
    # The critical operation which can raise an exception is placed
    # inside the try clause and the code that handles exceptions
    # is written in the except clause.
    # Printing the only accept for ZeroDivisionError
    print("Enter two numbers to divide.")
    print("Enter the numerator.")
    print("Enter the denominator.")
    # Printing the error message.
    print("Enter the numerator.")
    print("Enter the denominator.")
    ZeroDivisionError: division by zero
    print("Enter two numbers to divide.")
    print("Enter the numerator.")
    print("Enter the denominator.")
    # Printing the error message.
    print("Enter the numerator.")
    print("Enter the denominator.")
    ZeroDivisionError: division by zero
    print("Enter two numbers to divide.")
    print("Enter the numerator.")
    print("Enter the denominator.")
    # Printing the error message.
    print("Enter the numerator.")
    print("Enter the denominator.")
    ZeroDivisionError: division by zero
```

Try Finally

```
# Try finally can be followed with the finally statement.
# Code from the finally block
Enter the numerator.
Enter the denominator.
# This will always be printed.
# Finally statement will always execute.
try:
    Enter the numerator.
    Enter the denominator.
    Traceback (most recent call last):
    ZeroDivisionError: division by zero
    Enter the numerator.
    Enter the denominator.
    # This will always be printed.
    ZeroDivisionError: division by zero
```

Try Except Finally

```
# In this case, statement in the finally will execute.
# Code from the finally block
# Printing the only accept for ZeroDivisionError
Enter two numbers to divide.
Enter the numerator.
Enter the denominator.
Enter the numerator.
Enter the denominator.
Enter the numerator.
Enter the denominator.
Traceback (most recent call last):
ZeroDivisionError: division by zero
Enter two numbers to divide.
Enter the numerator.
Enter the denominator.
# Printing the error message.
Enter the numerator.
Enter the denominator.
ZeroDivisionError: division by zero
Enter two numbers to divide.
Enter the numerator.
Enter the denominator.
# Printing the error message.
Enter the numerator.
Enter the denominator.
ZeroDivisionError: division by zero
```

Try Except Else

```
# In the another critical another
# Another another critical another part
try:
    # Print the recipient of a number inputted by user.
    print("Enter the recipient of a number inputted by user.")
    print("Enter the recipient.")
    print("Enter the recipient.")
    # Printing the error message.
    print("Enter the recipient.")
    ZeroDivisionError: division by zero
    print("Enter the recipient.")
    print("Enter the recipient.")
    # Printing the error message.
    print("Enter the recipient.")
    print("Enter the recipient.")
    ZeroDivisionError: division by zero
    print("Enter the recipient.")
    print("Enter the recipient.")
    # Printing the error message.
    print("Enter the recipient.")
    print("Enter the recipient.")
    ZeroDivisionError: division by zero
```

User input validation

```
# Printing a user-entered value from the user entry in it correct.
# Printing a user-entered value from the user entry.
Enter Text:
    Traceback (most recent call last):
    ZeroDivisionError: division by zero
    # Error if this line executed if and only if all the above works.
    Enter Text:
    # Printing a user-entered, not statements.
    ZeroDivisionError:
    ZeroDivisionError:
    ZeroDivisionError:
```