**1. TRUE / FALSE QUESTIONS**

_____In a nested loop, the inner loop goes through all of its iterations for each iteration of the outer loop.
_____In order to draw an octagon with turtle graphics, you would need a loop that iterates eight times.
_____In a nested loop, the inner loop goes through all of its iterations for every single iteration of the outer loop.
_____Python function names follow the same rules as those for naming variables.
_____A hierarchy chart shows all the steps that are taken inside a function.
_____A local variable can be accessed from anywhere in the program.
_____Different functions can have local variables with the same names.
_____In Python, you cannot write functions that accept multiple arguments.
_____Functions make it easier for programmers to work in teams.
_____Calling a function and defining a function mean the same thing.
_____A flowchart shows the hierarchical relationships between functions in a program.

**2. COMPLETION QUESTIONS:** Fill in the blanks.

**a)** The function header begins with the keyword _____ and is followed by the name of the function.
**b)** The top-down design breaks down the overall task of a program into a series of _____.
**c)** A(n) _____ chart is a visual representation of the relationships between functions.
**d)** A variable is accessible only to statements in the variable's _____.

**3. ALGORITHM WORKBENCH QUESTIONS**

**a)** What will the following program display?

```
def main():
   x = 1
   y = 3.4
   print(x, y)
   change_us(x, y)
   print(x, y)
def change_us(a, b):
   a = 12.1
   b = 5.5
   print(a, b)
main()
```

**b)** Write a function named `square` that receives one parameter (side of square in pixels) from the caller then it draws a square of desired size by using turtle graphics.

**MULTIPLE CHOICE QUESTIONS**

**4.** The first line in a function definition is known as the function_____.
**a)** header
**b)** block
**c)** return
**d)** parameter

**5.** The _____ design technique can be used to break down an algorithm into functions.

**a)** subtask
**b)** block
**c)** top-down
**d)** simplification

**6.** A(n) _____ chart is also known as a structured chart.

**a)** flow
**b)** data
**c)** hierarchy
**d)** organizational

**7.** The _____ of a local variable is the function in which that variable is created.

**a)** global reach
**b)** definition
**c)** space
**d)** scope

**8.** A(n) _____ is any piece of data that is passed into a function when the function is called.

**a)** global variable
**b)** argument
**c)** local variable
**d)** parameter

**9.** A(n) _____ is a variable that receives an argument that is passed into a function.

**a)** global variable
**b)** argument
**c)** named constant
**d)** parameter

**10.** What does the following program do?

```
student = 1
while student <= 3:
    total = 0
    for score in range(1, 4):
        score = int(input("Enter test score: "))
        total += score
    average = total/3
    print("Student ", student, "average: ", average)
    student += 1
```

**a)** It accepts 4 test scores for 3 students and outputs the average of the 12 scores.
**b)** It accepts 3 test scores for each of 3 students and outputs the average for each student.
**c)** It accepts 4 test scores for 2 students, then averages and outputs all the scores.
**d)** It accepts one test score for each of 3 students and outputs the average of the 3 scores.

**11.** What does the following program do?

```
import turtle
def main():
    turtle.hideturtle()
    thing(100,0,50,'blue')
def thing(x, y, width, color):
    turtle.penup()
    turtle.goto(x, y)
    turtle.fillcolor(color)
    turtle.pendown()
    turtle.begin_fill()
    for count in range(2):
        turtle.forward(width)
        turtle.left(90)
    turtle.end_fill()
main()
```

**a)** It draws a blue square.
**b)** It draws a blue triangle.
**c)** It draws 2 blue lines.
**d)** Nothing since you cannot call a function with turtle graphics.

**PROGRAMS**

**12.** Write a program that estimates the value of the mathematical constant e by using the formula [Note: Your program can stop after summing 10 terms.]

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots$$

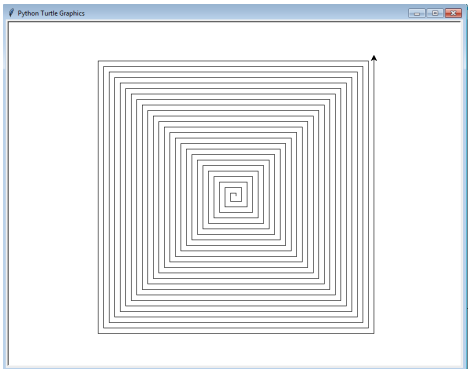
**13.** Write a program that computes the value of $e^x$ by using the formula [Note: Your program can stop after summing 10 terms.]

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

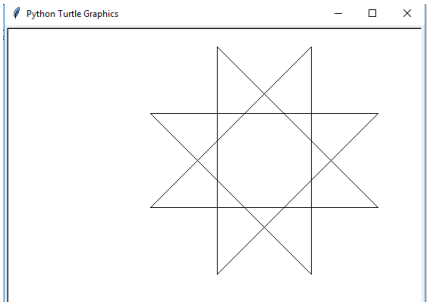
**14.** *Displaying a number pattern by using loops*: Use nested loops that display the following pattern:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

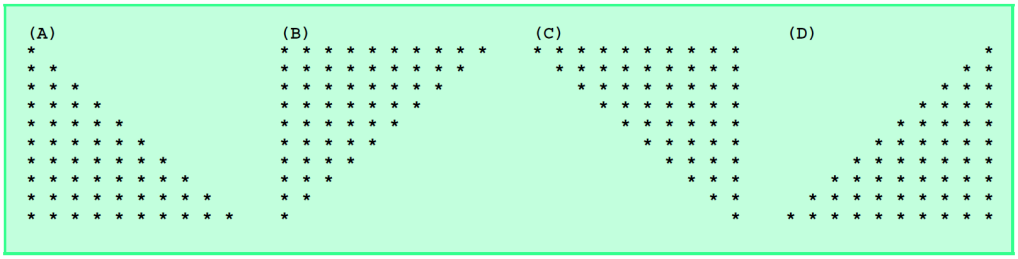
**15.** Use a loop with the turtle graphics library to draw the design shown in Figure.



**16.** Use a loop with the turtle graphics library to draw the design shown in Figure.




**17.** Write a program that prints the following patterns separately, one below the other each pattern separated from the next by one blank line. Use **for** loops to generate the patterns. All asterisks (*****) should be printed by a single statement of the form **print ('*',end=''),** (which causes the asterisks to print side by side separated by a space). (*Hint*: The last two patterns require that each line begin with an appropriate number of blanks.) Extra work: Combine your code from the four separate problems into a single program that prints all four patterns side by side by making clever use of nested **for** loops. For all parts of this program—minimize the numbers of asterisks and spaces and the number of statements that print these characters.



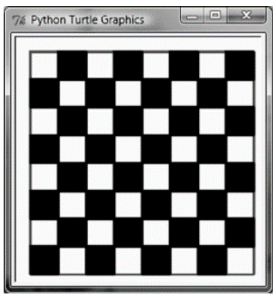

**18.** *(Pythagorean Triples)* A right triangle can have sides that are all integers. The set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for **side1**, **side2** and **hypotenuse** all no larger than 20. Use a triple-nested **for**-loop that tries all possibilities. This is an example of "brute force" computing. You will learn in more advanced computer science courses that there are many interesting problems for which there is no known algorithmic approach other than sheer brute force.

**19.** (*Display four patterns using loops*) Use nested loops that display the following patterns in four separate programs:

```
Pattern A        Pattern B        Pattern C        Pattern D

1                1 2 3 4 5 6              1        1 2 3 4 5 6
1 2              1 2 3 4 5              2 1        1 2 3 4 5
1 2 3            1 2 3 4              3 2 1        1 2 3 4
1 2 3 4          1 2 3              4 3 2 1        1 2 3
1 2 3 4 5        1 2              5 4 3 2 1        1 2
1 2 3 4 5 6      1              6 5 4 3 2 1        1
```

**20.** (*Turtle: chessboard*) Write a program to draw a chessboard, as shown in Figure.




**21.** Compute π : You can approximate by using the following series:

$$\pi = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots + \frac{(-1)^{n+1}}{2n-1}$$

Write a program that displays the π value for *n* = 100000, 200000, . . ., and 1000000.

In your program, there will be a function named `pi` that receives a positive integer number from the caller. The function calculates the desired number of terms and displays the result.

Design your main program such that calling the function `pi` and generates a table similar to the following.

```
For n:  100000 calculated PI is  3.141583
For n:  200000 calculated PI is  3.141588
For n:  300000 calculated PI is  3.141589
For n:  400000 calculated PI is  3.141590
For n:  500000 calculated PI is  3.141591
For n:  600000 calculated PI is  3.141591
For n:  700000 calculated PI is  3.141591
For n:  800000 calculated PI is  3.141591
For n:  900000 calculated PI is  3.141592
For n: 1000000 calculated PI is  3.141592
```

**22.** Python allows you to repeat a string by multiplying it by an integer, e.g. 'Hi' * 3 will give 'HiHiHi'. Pretend that this feature does not exist, and instead write a function named repeat that accepts a string and an integer as arguments. The function should display a string of the original string repeated the specified number of times, e.g. `repeat('Hi', 3 )` should display 'HiHiHi'. Write a program that test/uses the repeat function. In your program the string and how many times of repetition is taken from user in the `main` then the main program calls `repeat` function to print the string as desired number of times.

**Example terminal output of the program is the followings where HKU is entered as string and 7 times inputted:**

```
Enter a string:HKU
How many times?7
HKUHKUHKUHKUHKUHKUHKU
```

**23.** Write a program in which there will be a function named display_date() which is called by main program and displays the given date in the following format. For instance if the function is called with 27 , 'October',2021, then program should print to the screen as **Today's date is October 27, 2021.** An example run of the program is shown below (*note that the main function receives the date info from the user then sent them to display_date function be printed in the desired format).*

```
Enter today's day: 27
Enter today's month: October
Enter today's year: 2021
Today's date is October 27, 2021.
```

**24.** There are three seating categories at a stadium. Class A seats cost $20, Class B seats cost $15, and Class C seats cost $10. Write a program that asks how many tickets for each class of seats were sold, then displays the amount of income generated from ticket sales. In your program there should be a function named `showIncome` which receives how many tickets are sold from each groups from the caller then calculates the income and display it on the screen. Main function should get the number of tickets sold from each classes and pass them to function.

Example terminal output of the program is given below:

```
Enter count of A seats: 12
Enter count of B seats: 11
Enter count of C seats: 14
Income from class A seats: $240
Income from class B seats: $165
Income from class C seats: $140
Total Income: $545
```

**25.** Write a function named `smaller` that is used to print the smaller of given two numbers sent by the caller. Demonstrate the function in a program by calling it from main function. The numbers will be taken from the user by the main and passed to the function through parameter list.

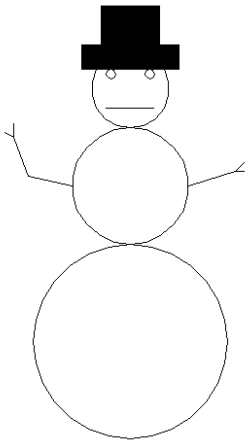
An example run of the program shown below:

```
Enter the first number: 1.73
Enter the second number:-2.65
Smaller one is -2.65
```

**Hint:** Function header for the smaller function should like the followings

```
def smaller (a,b):
```

**26.** Write a program that uses turtle graphics to display a snowman, similar to the one shown in the figure. In addition to a main function, the program should also have the following functions:



- `drawBase`. This function should draw the base of the snowman, which is the large snowball at the bottom.
- `drawMidSection`. This function should draw the middle snowball.
- `drawArms`. This function should draw the snowman's arms.
- `drawHead`. This function should draw the snowman's head, with eyes, mouth, and other facial features you desire.
- `drawHat`. This function should draw the snowman's hat.