Question Time 1

```
x = 1
if x>0:
    x = x-2
elif x<0:
    x = x+3
else:
    x = 0
```

What is the value of x?

A. -1 B. 0 C. 1 D. 2 E. None of these

Answer 1

- A. -1

Question Time

```
x = 1
if x > 0:
    x = x - 2
if x < 0:
    x = x + 3
    print (x)
```

What is the output?

A. 2   B. -1   C. 4   D. None of these

Answer

- D. 2

Question Time

```
x = 1
if x==2 or 3 or 4:
    print ('yes')
else:
    print ('no')
```

What is the output?

A. 'yes' B. 'no' C. 'The Denver Broncos'
D. An error message

Answer

- *A. 'yes'*
  - "**`if x==2 or 3 or 4`**" will always evaluate to True because it is equivalent to "**`if (x==2) or (3) or (4),`**" which would evaluate to True if any one of the three expressions evaluate to True.
  - Note that "if 3" and "if 4" always evaluate to True

Question Time

```
x = 'EWNNES'
n = len(x)
if x[n-1:n]=='ES':
     print ('South East')
elif x[n-2:n-1]=='NE':
     print ('North East')
else:
     print (x[n/2])
```

Output? (A) **South East**

(B) **North East**

(C) An error message is printed

(D) None of these

Answer

- (C) An error message is printed
  - The control would enter else statement
  - TypeError: string indices must be integers

Question Time

```
def f(x):
    z = 2*x;
    y = z+1;
    return y
if __name__=='__main__':
    z = 10;
    x = f(4)
    print (z,x)
```

What is the output?

A. 10 4   B. 10 9   C. 8 4   D. 8 9

Answer

- B. 10 9

Question Time

```
def f(x):
    y = 2*x
    print (y)

if __name__==‘__main__’:
    f(4)
    z = f(4)
    print (z)
```

What is the output:

```
A:    8       B: 8          C: 8      D:  8
      8          8             None
                 None          8
```

Answer

- B.
  ```
  8
  8
  None
  ```

Question Time

```
>>> s1 = input('First String: ')
>>> n1 = s1.count('ab')
>>> s2 = input('Next String: ')
>>> n2 = s2.count('ab')
>>> s = s1 + s2
>>> B = n1+n2 == s.count('ab')
```

What can you say about the value of `B`?

A. Always True B. Always False
C. Can be either True or False

Answer

- C. Can be either True or False
  - True case:
    - S1=“ababccc”
    - S2=“bccababc”
  - False case:
    - S1 = “aba”
    - S2 = “babcc”
  - If S1 ends with “a” and S2 begins with “b,” the condition would be False since S1+S2 will have an additional “ab”

Question Time

```
>>> s = 'abcabcabc'
>>> s.find(‘ca’)
2
>>> n = s.find(‘bc’)+s.find(‘bc’)
>>> print (n)
```

What is the green box?

A. 2   B. 4   C. 7

Answer

- **A. 2**

Question Time

```
>>> s = 'abcdef'
>>> s.replace('bc','xx')
'axxdef'
>>> u = s.replace('de','yy')
>>> print (u)
🟩
```

What is the green box?

`A. 'axxdef'` `B. 'abcyyf'` `C. 'axxyyf'`

Answer

- B. `‘abcyyf’`
  - Recall, `replace()` does not modify the original string unless we overwrite it

Question Time

```
s = '12345'
t = 'x'
for c in s:
    t = t+t
print (len(t))
```

Output?

A. 10 B. 15 C. 32 D. None of These

Answer

- c. 32

Question Time

```
T = ''
S = 'abcabcabc'
for c in S:
    if T.count(c)==0:
        T = T + c
print (T)
```

Output?

A. 'ccc' B. 'abc' C. `cba'
D. None of These

Answer

- **`B.  'abc'`**

String manipulation 1

- Implement the following function so that it performs as specified.

```
def Q1(s):
    """" Returns True if the characters at the start and
    end of s are the same and occur nowhere else in s

    PreCondition: s is a string with length greater
    than or equal to 3. """"
```

Text

String manipulation 1 (Ans)

```
def Q1v1(s):
   """ Returns True if the characters at the start and end of s are the
   same and occur nowhere else in s
   PreCondition: s is a string with length equal to 3 or greater."""
   n = len(s)
   t = s[1:n-1]
   return s[0]==s[n-1] and t.count(s[0])==0
```

String manipulation 2

- Imagine playing around with this script:

```
s = input('Enter a string that has length greater than or equal to 2: ')
t = s.replace(s[0],'x')
u = t.replace('x',s[0])
print (s,u)
```

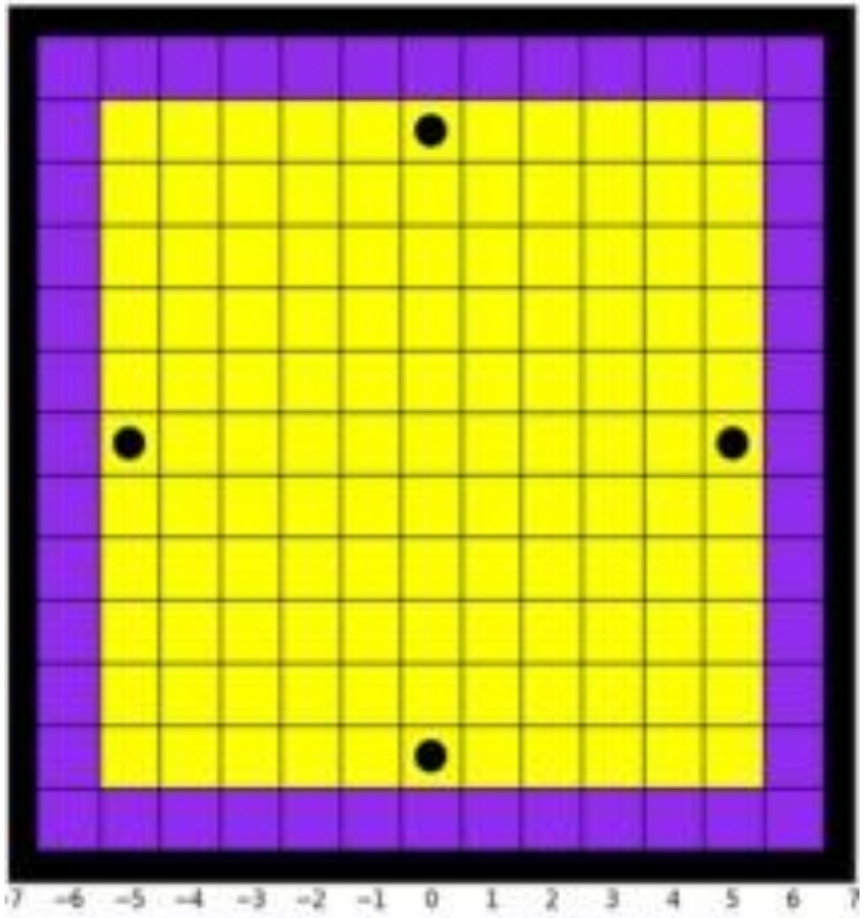
- Sometimes it is the case that the printed values of s and u are the same and sometimes it is observed that they are different. Give a Boolean expression that is True if u and s have the same value and is False otherwise. Hint. Consider some small examples.

String manipulation 2 (Ans)

```
if s[0] == 'x' or s.count('x') == 0:
    return TRUE
else:
    return FALSE
```

Random walk

- A random walk simulation produces a travel string comprised of the characters N, S, E, and W. The travel string encodes the hop directions associated with the robots journey from (0,0) to a purple boundary tile. Here is a display of an n = 5 “hopping arena” highlighting its four middle edge tiles (solid black dots):



Random walk 1

- Assume that x and y are initialized with the (x,y) coordinates of the robot’s location and that the value of n is the size of the hopping arena. Give a Boolean expression that is True if the robot is on a middle edge tile and False otherwise.

Random walk 1 (Ans)

- (abs(x)==n and y==0) or (abs(y)==n and x==0)

Random walk 2

- A hop is “predictable” if it is in the same direction as the previous hop. Here is a travel string that includes 3 predictable hops: ’EWNNNWWN’. Complete the following function so that it performs as specified.

```
def nPredictable(s):
  """ Returns an int that is the number of
  predictable hops in s. Precondition: s is
  a travel string.
  """
```

Random walk 2 (Ans)

```
def nPredictable1(s):
    """Same spec as in the question."""
    count = 0
    for k in range(1,len(s)):  #k = position to check if previous is
                               #the same
        if s[k] == s[k-1]:
            count = count + 1
    return count
```

Q1

- Assign a value to x so that the character ’A’ is printed out:

```
x  =  ______________________________

if  x%2==0 and x%3==1:
  print (’A’)
```

Q1 (Ans)

- Any even number whose remainder when divided by 3 is 1 will work
  - E.g., 4, 10

Q2

- Assign values to x and y so that the character ‘D’ is printed out:

```
 x  = ____________________________

  y  = ____________________________
if   not ((0<=x<=3) and (0<=y<=3)):
   print (’A’)
elif y<=1 or y>=2:
   print (’B’)
elif x<=1 or x>=2:
   print (’C’)
else:
  print (’D’)
```

Q2 (Ans)

- $x = 1.5$; $y = 1.5$
  - Thinking about these constraints geometrically can help.

Q3

- What would be the output if the following code is executed?

```
x = float(10/4)
print(x)
```

Q3

- 2.5

Q4

- Suppose the functions in modules M1.py and M2.py are to be used by module M.py. Briefly explain why it is safer to implement M.py with

```
import M1
import M2
```

- As opposed to with

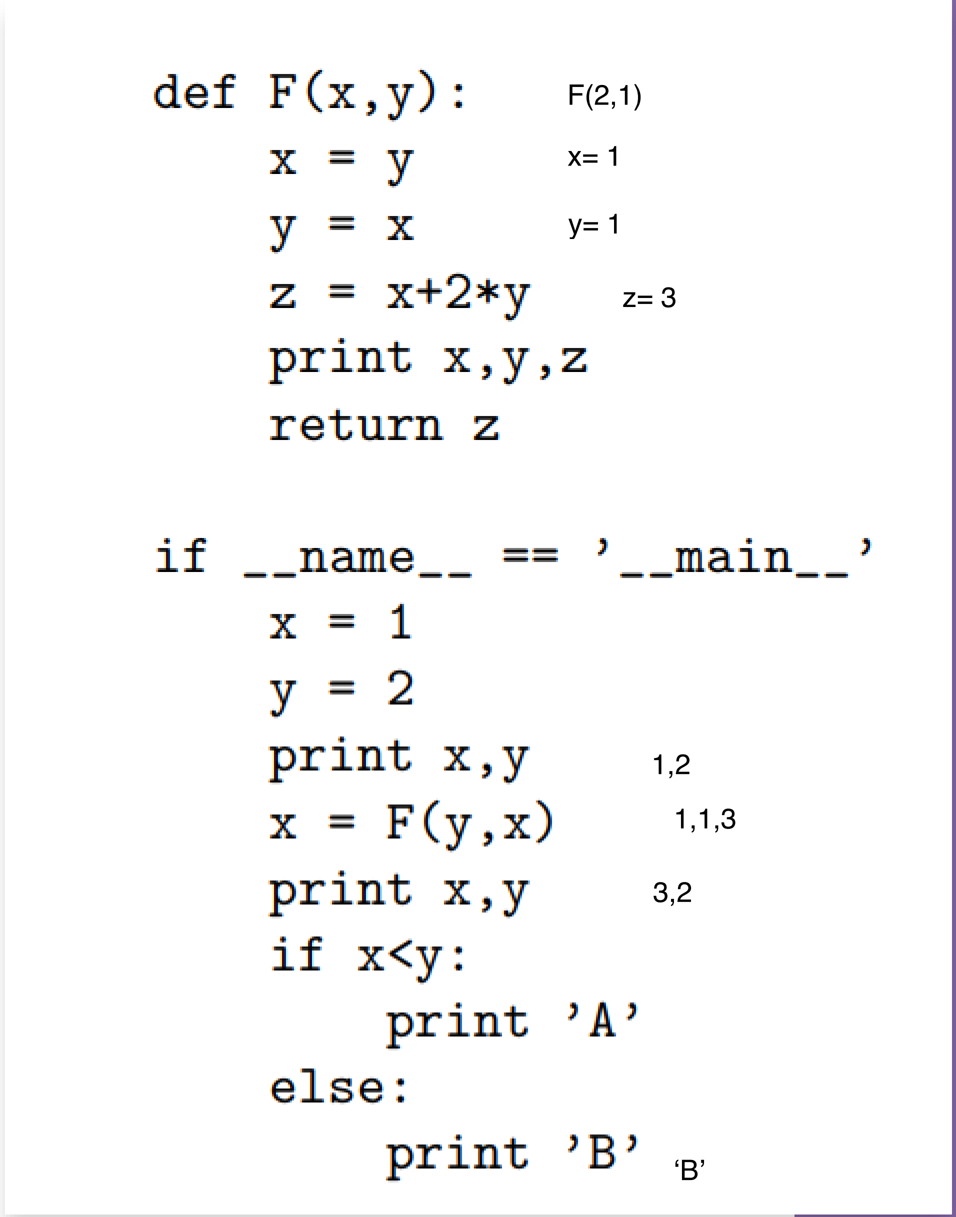
```
Import M1 *
Import M2 *
```

Q4 (Ans)

- you could import more than you bargained for
- With “import M1 *”, “import M2 *” everything inside modules M1, M2 is imported, which could lead to name conflicts

Q5

- Indicate what the output would be if the following application script is run:



```
def F(x,y):          F(2,1)
    x = y            x= 1
    y = x            y= 1
    z = x+2*y          z= 3
    print x,y,z
    return z

if __name__ == '__main__'
    x = 1
    y = 2
    print x,y          1,2
    x = F(y,x)          1,1,3
    print x,y          3,2
    if x<y:
        print 'A'
    else:
        print 'B'  'B'
```

Q5 (Ans)

```
# global space:
#  x: 1 y: 2
# -----------> prints 1 2
# Call to F
#     For F, x gets AS's y=2; For F, y gets AS's x=1
#     So, for F, x: 2, y: 1
#     For F: x gets F's y, so
#           x: 1   y:2
#     For F: y gets F's x, so
#           x: 1   y:1
#     For F: z gets F's x + 2 times F's y
#          so: x: 1 y:1 z= 3
#     -------------> prints 1  1  3
#      F returns 3
# global x gets returned value
# x: 3 y:2
# --------> prints 3 2
# ----------> prints 'B'
```

Loops 1

- Consider the following script

```
t = ’x’
s = input(’Enter a string: ’)
for c in s:
    t =  t + c + t
```

- Assuming that ’ba’ is assigned to s, what is the final value of t? Show work.

Loops 1 (Ans)

- soln='xbxaxbx'
- # showing work
  - # t: 'x', s: 'ba’
  - # c: 'b' makes t: 'xbx’
  - # c: 'a' makes t: 'xbxaxbx'

Loops 2

- Write a script that is equivalent to the following script but which uses a while-loop instead of a for-loop.

```
t = ’x’
s = input(’Enter a string: ’)
for c in s:
    t =   t + c + t
```

Loops 2 (Ans)

```
def loopsWhile(s):
    """supposed to be equivalent to above."""
    t = 'x'
    i = 0 # index in s to consider
    while i < len(s):
        c = s[i]
        t = t + c + t
        i+= 1
    return t
```