Name : 曾焕志  
学号: H24115069

**#Part 1-1**

s1 = "spam"

s2 = "ni!"

a = "The knights who say," + s2

print(a)

b = 3 \* s1 + 2 \* s2

print(b)

c = s1[1]

print(c)

d = s1[1:3]

print(d)

e = s1[2]+ s2[:2]

print(e)

f = s1 + s2[-1]

print(f)

g = s2[len(s2)//2]

print(g)

**#Part 1-2**

#(a) "NI"

#(b) "ni!spamni!"

#(c) "SpamNi! SpamNi! SpamNi!"

#(d) "span"

#(e) "spm"

a2 = s2.upper()

print("a=",a2)

b2 = s2 + s1 + s2

print("b=",b2)

c2 = (s1.capitalize() + s2.capitalize() + " ") \* 3

print("c=",c2)

d2 = s1.replace("m","n")

print("d=",d2)

e2 = s1.replace("a","")

print("e=",e2)

**#Part 1-3**

a3 =print("Looks like %s and %s for breakfast" % ("spam", "eggs"))

b3 =print("There is %d %s %d %s" % (1, "spam", 4, "you"))

#c3 =print("Hello %s" % ("Suzie", "Programmer")) is wrong because there is only one string formatting which defines the value "Suzie".

#There needs to be an additional string formatting to define the value "Programmer"

c3 =print("Hello %s %s"%("Suzie","Programmer"))

d3 =print("%0.2f %0.2f" % (2.3, 2.3468))

e3 =print("%7.5f %7.5f" % (2.3, 2.3468))

f3 =print("Time left %02d:%05.2f" % (1, 37.374))

#g3 =print("%3d" % ("14")) is wrong because %d defines integer values. The usage of "" converts 14 into a string which can's be defined by %d

g3 =print("%3d" % (14))

**#Part 1-4**

#(a)

x1 = 5 #set value for variable x

y1 = 3 #set value for variable y

if x1 >= y1: #"if" statement to change the value of x if x is equal to or bigger than y

x1 = x1 - 2 #the new value for x if the "if" statement is fulfilled

print(x1) #print the variable x

#(b)

tc = 100 #set value for variable tc

tf = (9/5) \* tc + 32 #set value for variable tf (9 divided by 5 multiplied by the integer value of tc plus the integer 32)

print(tf) #prints the variable tf

#(c)

x3 = 0 #set value for x

while x3 < 5: #"while" statement to check if the value x is less than 5 continuously in a loop as

long as the value is true while "if" statements only does the action once if the value is determined to be

x3 = x3 + 1 #new value for variable x3 as long as the "while" statement is true

print(x3) #print variable x (result depends on whether or not the "while" statement is true or false

#(d)

x4 = 1 #set value for x

i4 = 1 #set value for i

while x4 <= 5: #"while" conditional that needs to be fulfilled

x4 = x4 \* i4 #new value for x as long as the "while" condition, x being less than or equal to 5, is true

i4 = i4 + 1 #new value for i as long as the "while" condition, x being less than or equal to 5, is true

print(x4) #prints the value of x, which is 6 (after looping 5 times). The concept is more or less the same as problem (c), but it uses less than

#(e)

x5 = 0 #set value for x

while x5 < 6: #"while" conditional that needs to be fulfilled

if x5 % 2 == 0: #"if" conditional that needs to be fulfilled as long as "while" condition, x being less than 6, is true

print('even', x5) #the action done if "if" condition, modulus(remainder) of x divided by 2 is equal to 0, is true. The action is print the

#string "even" and integer x

else: #"else" statement to do a certain action if "if" value is false

print('odd', x5) #the action done if "if" statement is false. which is printing the "odd" string and integer x

x5 = x5 + 1 #makes sure that the program ends by increasing the x value by 1 after each loop so the while condition will eventually be false

#the resulting print will either show odd and the number if the number is odd, and even and the number if the number is even. the result will always change

#since it always loops until x is less equal to or more than 6. All the results are printed because the "print" command is inside of the "while" function

#(f)

i6 = 0 #set value for i

while i6 < 6: #"while" condition,i being less than 6, that needs to be true to perform an action

j6 = 0 #set value for j

while j6 < i6: #"while" condition that needs to be true to perform an action as long as the previous "while" condition is true

print("\*") #action done as long as the second "while" condition,j being less than i, is true, which is printing the "\*" string

j6 = j6 + 1 #makes sure the program ends by increasing the j value by one each loop so the while condition will eventually be false

i6 = i6 + 1 #makes sure the program ends by increasing the i value by one each loop so the while condition will eventually be false

print() #prints a new paragraph after every loop is done

#(g)

score = 40 #set value for the variable "score"

while score > 1: #"while" condition,the "score" variable being bigger than 1, that needs to be true for an action be done

score = score/2 - 1 #action done as long as the while condition is true, which is dividing the score by 2 and subtracting it by 1

print(score) #prints the new variable after the "while" statement

#note: the result is 0.625 because it runs the program in a loop continuously until it is less than 1, however since there is no indent infront of "print"

#it only prints the final result once, after the while loop has ended.

#(h)

x8 = 2 #set value for x

y8 = 7 #set value for y

while x8 < y8: #"while" condition, x being less than y, that needs to be true for the action to be done

x8 = 2 \* x8 #the action done as long as the "while" condition is true

print(x8) #prints the new value of x after the "while" statement because the command is outside of the while statement

#(i)

a9, b9 = 63, 105 #set value for variable a and b

while b9: #"while" conditions that run an action as long as the value of the variable b does not change

a9, b9 = b9, a9 % b9 #set variable a equal to variable b after the "while" statement. Set variable b after the "while" statement as the

#modulus(remainder) of a divided by b

print(a9) #prints the value of a after the while statement

print(b9) #prints the value of b after the while statement

#note: the loop happens 3 times. The first changes the value of a into 105 and b into 63. The second changes a into 63 and b into 42. The last results in

#the printed value of 21. The loop does not happen for the 4th time because the modulus result is 0. "While" loops will only loop if the value is true and

#non-zero. Last value for a is 21 and not 0 because since 0 is the last value of b, b

#(j)

n10 = 21 #set value for n

while n10 != 1: #"while" condition, n not equal to 1, that needs to be true to run an action

print(n10, end=", ") #the action that is done as long as the "while" condition is true which is printing the variable n and ", " behind n

if n10 % 2 == 0: #"if" condition n modulus 2 is equal to 0 ("==" ignores the datatypes of the 2 variable).

n10 = n10 // 2 #the new value for n if the "if" condition is true, which is n (floor) divided by 2 and rounded to the left.

else: #sets an action that would be done if the "if" condition is false

n10 = n10 \* 3 + 1 #the new value for n if the "if" condition is false, which is n times 3 plus 1

print(n10, end=".\n") #as long as the "while" condition is true, it will print the new variable n, a period behind n, and a new paragraph

n10 = 10 #there is no difference in how the program is run. The result however, always ends with 2, 1 because the program will always

#loop until n is equal to 1

while n10 != 1:

print(n10, end=", ")

if n10 % 2 == 0:

n10 = n10 // 2

else:

n10 = n10 \* 3 + 1

print(n10, end=".\n")

#Part 1-5

#(a)

x = 7 #set value for x

y = 8 #set value for y

if x < 7 or x <= 10 and y > 8: #sets the "if" conditional, which is either x being less than or equal to 10 (7 is allowed because the statemen uses

#"or" and not "and" meaning that as long as one of the conditions are met, the "if" statement is still true. "and"

#statements requires both to be true

print("ugh") #the action done if "if" conditions are true, which is printing the string "ugh"

else: #sets an alternative action if "if" conditions are false

print("yuck") #the action done if "if" conditions are false, which is printing the string "yuck"

#note: the result will be printing "yuck" because x = 7 does not fulfill the first if statement where x needs to be less than 7.

#(b)

phrase = "python" #sets the variable "phrase" value as the string "python"

vowels = "aeiou" #sets the value of the variable "vowels" as the string "aeiou"

count = 0 #sets the value of the variable "count" as the integer 0

while (not phrase[count] in vowels): #sets "while" condition, starting from the first character in the string, as long as there is no characters in

#variable phrase that is not also in the variable "vowels".

count = count + 1 #the action done as long as the "while" condition is true, which is adding the variable count by 1.

print(count) #prints the contents of the variable count after the "while" loop ends

#note:We use square brackets to identify the numerical position of the characters in our string starting from 0. Since the variable "count" is an integer, 0,

#it will scan the characters starting from the first character. By using the "not" relational operator and "while" loop as well as the "count = count +1"

#command, it will continuously increasing the value of "count" therefore makaing it scan the characters 1 by 1 to the right until it finds a character in

#the variable "vowels", which in this case is in the 4th position of the "phrase" variable, which is the letter o.

#(c)

if 'alpha' < 'zebra': #sets "if" condition, if the string 'alpha' is less than the string 'zebra', do a certain action

print('alpha < zebra') #the action done, which is printing the string 'alpha < zebra'

elif 'alpha' > 'zebra': #sets another "if" condition if the first "if" condition is false, which is 'alpha' bigger than 'zebra'

print('alpha > zebra') #the action done if first "elif" is true, prints the string 'alpha > zebra'

elif 'alpha' == 'zebra': #another "if" condition if the first "elif" condition is false, which is 'alpha' being equal to 'zebra'

print('alpha == zebra') #the action done if the second "elif" condition is true, prints the string 'alpha == zebra'

else: #runs an action if all cocnditions above are false

print('none of the above') #the action done if all conditions above are false, prints the string 'none of the above'

#note: the resulting print will be 'alpha < zebra' because based on the lexicographic order, the letter a comes before the letter z making a "less than" z

#Part 1-6 based on hints

thief = 1

while thief <= 4:

if thief == 1 and thief == 3 and thief == 4 and thief != 4:

print("The thief is",thief)

elif thief != 1 and thief != 3 and thief == 4 and thief != 4:

print("The thief is",thief)

elif thief != 1 and thief == 3 and thief != 4:

print("The thief is",thief)

elif thief != 1 and thief == 3 and thief == 4:

print("The thief is",thief)

thief = thief + 1