CS 177
Review
Week 7
Announcements
ANY QUESTIONS?
Table of Contents

- Binary
- Assignment statements
- Numeric Data Types
- Strings
- Lists
- Functions
- Conditionals
- Loops
Binary

- What is the decimal equivalent of \((10010)_2\)?
- What is the largest decimal number that could be represented using 6 bits?
Binary

- What is the decimal equivalent of $(10010)_2$?  
  - 18

- What is the largest decimal number that could be represented using 6 bits?  
  - 63
Assignments

- **var = value**
  - value is stored in var

- **var = expression**
  - expression is evaluated and the result is stored in var

- `<variable> = eval(input(<prompt>))`
  - input(…) returns the input as a string!
  - eval “evaluates” it, to a Python value

- **v1, v2 = expr1, expr2**
  - Simultaneous assignments : v1 = expr1, v2 = expr2
Exercise

x = input(“Enter value”)
y = eval(input(“Enter value”))
z = x+y
Solution

x = input("Enter value")  ➔ Takes in a string as input

y = eval(input("Enter value"))  ➔ Evaluates a string to int

z = x+y  ➔ ERROR!

TypeError: Can't convert 'int' object to str implicitly
Numeric Data Type

- Whole numbers are represented using the integer (int) data type.
- Numbers that can have fractional parts are represented as floating point (or float) values.
Exercise

If, $x=10$, $y=3.0$, what will be the value of the following operations?

type($x$), type($y$)

$x+y$

$x/y$

$x//y$

$x\%y$

$x\%3$

$x/5$
Solution

If, x=10, y=3.0, what will be the value of the following operations?

type(x), type(y) → <class 'int'>, <class 'float'>

x+y → 13.0

x/y → 3.333

x//y → 3.0

x%y → 1.0

x%3 → 1

x/5 → 2.0
String

- Strings are defined with quotations
  - "hello"

- Stored as a sequence of characters in memory

- Characters could be accessed using indices

- Strings are immutable
  - We cannot assign to strings like we do to lists
Exercise

str1 = “abcdefgh”, str2 = “ijklmno”

length = len(str1)

str1[length]

str1[0] = ‘A’

str1 = str1+str2

str1 = str1+i

c = ord(str1[1])+5

chr(c)
Solution

str1 = “abcdefgh”, str2 = “ijklmno”

length = len(str1) → 8

str1[length] → IndexError: string index out of range

str1[0] = ‘A’ → TypeError: 'str' object does not support item assignment

str1 = str1+str2 → 'abcdefghijklmno'

str1 = str1+p → NameError: name 'p' is not defined

c = ord(str1[1])+5 → 103

chr(c) → ‘g’
Exercise

str1 * str2
3*str1
str1[:-3]
str1[3:5]
str1[3,1]
str1[5:]
str1[:3] + str2[-1:]
Solution

str1 * str2 → TypeError: can't multiply sequence by non-int of type 'str'

3*str1 → 'abcdefghijklmnoabcdefghijklmnoabcdefghijklmno'

str1[:-3] → 'abcdefghijkl'

str1[3:5] → 'de'

str1[3,1] → TypeError: string indices must be integers

str1[5:] → 'fghijklmno'

str1[:3] + str2[-1:] → 'abco'
Lists

- Multiple elements are stored consecutively in memory
- Lists can be nested
  - `[[1,2], 4, [6, 10, [11, 12]], 5]`
- Lists can also be empty
  - `[]`
- Lists elements can be of different types
  - `[1, 2, ["ABCD", "EFG"], 3, 4]`
Indexing Lists

- Indexing provides us with a quick mechanism for accessing a given element within the list.
- The index starts at 0 NOT 1 so the last element is the length of the list – 1.
- \( L[i] \) will give you the \( i \)th element of the list.
Exercise

mylist = [1,2,[3,4,"abcd"],"efgh"]

mylist[1]

How do you access the character ‘c’ from mylist?

len(mylist)

mylist[len(mylist)-1][len( mylist[len(mylist)-1] )-1 ]

mylist[-2]

mylist[-1*len(mylist)]
Solution

mylist = [1, 2, [3, 4, "abcd"], "efgh"]

mylist[1] \rightarrow 2

Access the character ‘c’ from mylist? \rightarrow mylist[2][2][2]

len(mylist) \rightarrow 4

mylist[len(mylist)-1][len( mylist[len(mylist)-1] )-1] \rightarrow ‘h’

mylist[-2] \rightarrow [3, 4, 'abcd']

mylist[-1*len(mylist)] \rightarrow 1
Exercise

```python
list(range(5))
list(range(-4))
list(range(3,5))
list(range(5,3))
list(range(5,3,-1))
list(range(5,3,-1))
list(range(10,0,-5))
list(“abcde”)
list(range(4)) + [4,5,6,7]
```
Solution

list(range(5)) → [0, 1, 2, 3, 4]
list(range(-4)) → []
list(range(3,5)) → [3, 4]
list(range(5,3)) → []
list(range(5,3,-1)) → [5, 4]
list(range(10,0,-5)) → [10, 5]
list("abcde") → ['a', 'b', 'c', 'd', 'e']
list(range(4)) + [4,5,6,7] → [0, 1, 2, 3, 4, 5, 6, 7]
Functions

- Functions take input and produce output
- Output is provided by the “return” statement
  - Otherwise the function does not provide output
- At the call site of the function the arguments get bound
  - The arguments can rebind variables that have already been defined for the duration of the call
- You can use variables defined outside the function but you must be careful!
Function Calls

- When a function is called
  - The argument values are bound (assigned) to the parameter names
  - The program of the function is executed as if those program statements were substituted for the call.
  - Upon execution of the return statement, the return value is substituted for the function call
  - If no return statement is executed, and the program of the function body ends, the function returns None, indicating that no value is returned to the calling place.
Exercise

```
a = 5

def f1(a):
    a = a+5
    return a

1) def main():
    x = f1(4)
    print(x)

2) def main():
    f1(10)
    print(a)

3) def main():
    a = f1(a)
    print(a)
```

What will be the output of each of the main functions?
Solution

\[
a = 5
\]

\[
\text{def } f1(a):
    a = a+5
    return a
\]

1) \[
\text{def } main():
    x = f1(4)
    print(x)
\]
\textbf{Ans}: 9

2) \[
\text{def } main():
    f1(10)
    print(a)
\]
\textbf{Ans}: 5

3) \[
\text{def } main():
    a = f1(a)
    print(a)
\]
\textbf{Ans}: UnboundLocalError: local variable 'a' referenced before assignment
Conditionals

- **Boolean (logical) expressions:**
  - An expression that can be evaluated as `True` or `False`

- Given a boolean expression, **if** defines the statements that have to be executed if the expression evaluates to `True`

- **else** specifies what to do if the expression is `False`

- **Multi-way decision / elif:**
  - Decision statements can be nested within one another creating complex logic
Exercise

If A, B, C are boolean expressions, which of the following expressions are equivalent?

i)   A and B or C

ii)  A or B and C

iii) B and C or A

a)   i) and ii)
b)   i) and iii)
c)   i) ii) and iii)
d)   None
Solution

If A, B, C are boolean expressions, which of the following expressions are equivalent?

i) A and B or C
ii) A or B and C
iii) B and C or A

a) i) and ii)
b) i) and iii)
c) i) ii) and iii)
d) None
Exercise

i)
if (a and b):
    print (1)
else:
    if (c and d):
        print (2)
    else:
        print (3)

ii)
if (a and b):
    print (1)
elif (c and d):
    print (2)
else:
    print (3)

Are the two codes equivalent?
Solution

Are the two codes equivalent?

YES
Exercise

What will be the output of calling main(6)

```python
def isInt(x):
    if(type(x)==int):
        return 1;
    else:
        return 0;

def main(a):
    if(isInt(a)):
        print("correct Input")
    else:
        print("Incorrect Input")
```

a) correct input  
b) Incorrect input  
c) none  
d) error
What will be the output of calling main(6)

```python
def isInt(x):
    if(type(x) == int):
        return 1;
    else:
        return 0;

def main(a):
    if(isInt(a)):
        print("correct Input")
    else:
        print("Incorrect Input")
```

a) correct input  
b) Incorrect input  
c) none  
d) error
Exercise

What will be the value of count if the following statements are executed?

a) 7
b) 0
c) 1
d) infinite

```python
list1 = []
count=0
list1.append(5)
list2 = [5,5,5]
list1=list1+list2
list1.extend([5,5,5])
while(list1):
    list1.remove(5)
    count=count+1;
```
Solution

What will be the value of count if the following statements are executed?

a) 7
b) 0
c) 1
d) infinite

```python
list1 = [];
count=0
list1.append(5)
list2 = [5,5,5]
list1=list1+list2
list1.extend([5,5,5])
while(list1):
    list1.remove(5)
    count=count+1;
```
Loops

- The two common types of loops you will use in python are the:
  - While loop
  - For loop

- For loops
  - Loops have more controlled structure and cannot run infinitely
  - Commonly used with range function

- While loops
  - Expresses an execution condition
  - Only executes until that condition no longer holds
Exercise

Will the three codes produce same output?

1) for i in range(0,10,2):
   print i

2) i = 0
   while(i<10):
      i = i+1
      if(i%2==0):
         print(i)

3) i = 0
   while(i<10):
      if(i%2==0):
         print(i)
      i = i+1
Solution

Will the three codes produce same output?

1) for i in range(0,10,2):
    print i

2) i= 0
   while(i<10):
       i=i+1
       if(i%2==0):
           print(i)

3) i= 0
   while(i<10):
       if(i%2==0):
           print(i)
       i=i+1

0 2 4 6 8
2 4 6 8 10
Exercise

- What will be the value of count at the end of the loops?

```python
count = 10
for i in range(5):
    for j in range(10, 0, -1):
        count = count + 1
```
Solution

What will be the value of count at the end of the loops?

count = 10
for i in range(5):
    for j in range(10,0,-1):
        count = count+1

60
def method1(list1, list2):
    for elt in list2:
        list1.append(elt)
    return(list1)

Let,
num4 = range(4);
str5= “abcde”; 

What will be the output of the function for the following calls:

a) print(method1(list1,list2))
b) method1([], “abcd”) 
c) list2 = method1(list1,list2)
Solution

def method1(list1, list2):
    for elt in list2:
        list1.append(elt)
    return(list1)

What will be the output of the function for the following calls:

a) print(method1( list(num4),list(str5) ) )

Prints [0, 1, 2, 3, 'a', 'b', 'c', 'd', 'e'] onto the console

b) method1([], str5)

['a', 'b', 'c', 'd', 'e'] is returned but not assigned to anything

c) list2 = method1(str5,num4)

AttributeError: 'str' object has no attribute 'append'

Let,
num4 = range(4);
str5 = “abcde”;
ANY QUESTIONS?