11 Dec' 24

HW: Finish previous HWs.

→ Statements : It is a single instruction (or a simple line of code, without ;) given to the computer

Eg: x = 2 # Assignment statement
Gives/assigns a value to
the variable

Eg: print("Hello \n") # print statement

Other examples: Loop statements, Conditional statements

→ Conditional statements: These statements allow you to run different pieces of code based on some comparisons of values

Eg: if x > 20

tab ← else

y = x**2

y = x/2

→ x = 21 ⇒ (implies) y = 21**2 = 441

→ x = 10 ⇒ y = 10/2 = 5

→ x = 20 ⇒ y = 20/2 = 10

If you want to check multiple conditions, you use if, elif, elif ..., else

↓
else if

Ex: Create a function to give grade from exam score (0-100)

```
def gradingFunction(examScore):
    grade = "F"
    if examScore > 90:
        grade = "A"
    elif examScore > 80:
        grade = "B"
    else:
        grade = "F"
    return grade
```

```
print(gradingFunction(95)); # "A"
                      (86)  # "B"
                      (70)  # "F"
```

→ Conditional statements take boolean values and execute (run) code if 'true'

if x :

— <code>

The <code> will run only if x is true.

Comparisons of values can be done using Comparison operators — >, <, >=, <=, ==, !=

Eg: 20 > 10 # true

Eg: 20 > 20 # false

Eg: 20 >= 20 # true

Eg: 20 != 20 # false

Eg: 20 == 20 # true

Eg: 20 < 20 # false

Eg: 20 <= 20 # true

Logical/Boolean operators: and, or, not (!)

If x & y are booleans,

→ x and y = true if both x = true and y = true

x and y = false if either x, or y, or both are false

→ x or y = true if either x, or y, or both are true

= false, if both are false

→ !x = true, if x is false

= false, if x is true.

→ For conditional statements, you need a boolean to compare, but it can also be a combination of boolean variables using boolean operators

```
if (x and y) or z :                # same as if 'q' where q = (x and y) or z
    <code>
```

Logical operator precedence : Order is

(1) not

(2) and

(3) or

Eg: result = True or False and not True

= True or False and False

= True or False

= True

H.W: Do these manually and in python

→ ((23 > 42) and (36>12)) or (39<=23)

→ result= (True or False) and (not True)

→ is_valid = 10>5 and 2+3*2 !=10 or not (5<2)

→ check = not (15//3==5 and 7%2==0) or

2**3 <10

→ Full operator precedence :

(1) Parentheses ( )

(2) Exponentials **

(3) Unary operators, not, ~ bitwise NOT
+, -

Eg: +25, -33,

(4) Multiplication (*), Division (/), Floor division (//),
modulus (%) — left to right
associativity

(5) Addition (+), subtraction (-) — left to right

(6) Bitwise shift <<, >>

(7) Bitwise AND

(8) Bitwise OR

(9) Bitwise NOT

(10) Comparision operators (==, !=, >, <,
>=, <=, is, is not, in, not in)
→ left to right → chained comparisons
are also possible a < b < c

(11) Logical not

(12) Logical and

(13) Logical or

(14) Assignment (=, +=, -=, *=, /=, //=, %=,
**=, &=, |=, ^=, >>=, <<=)
→ right to left associativity

→ Lists: Ordered collection of items that can be changed. And the items can be of different datatypes.

→ 'ordered' means each item has a number in the list

→ list item numbering starts from zero

Eg: myList = [2, 37, "pray", 0.5, True]

2 → zeroth item
37 → first item
True → fourth item

myList[0] → 2

myList[2] → "pray"

myList.append("world")

myList = [2, 37, 'pray', 0.5, True, "world"]

append function adds the parameter as an item at the end of the list

myList.remove(37)

myList = [2, 'pray', 0.5, True, "world"]

myList[1] = 'praying'

myList = [2, 'praying', 0.5, True, "world"]

Next class: Arrays, Loops, structures, classes, dictionaries

HW: (1) Create a list of 10 items of different datatypes

(2) Use all the list operations we learnt about — and print the list after each operation

(Eg: change 4th item, append some items, remove some items etc.)