Week1 act1

input:

print('Hip Hip Hurray',end =' ')

print('Hip Hip Hurray')

output: Hip Hip Hurray Hip Hip Hurray

print('' ,end = '') # end = '' | join two print output into one single line of display

Week 2 onwards

Arithmetic Operators:

% Remainder , x = y % z

// floor division , x = y // z (no remainder, integer vaule)

\*\* Power of y value , x = y \*\* z

Relational Operator - Conditional Operators:

== | Equal

!= | Not Equal

> | Greater than

< | Less than

>= | Greater than or Equal

<= | Less than or Equal

Logical Operators - Conditional Operators (Boolean Expression):

and | Only return True when both BE are True

or | Two Expression. Either one is True

not | Not is False

int | integer int() whole number:

float | float float() decimal number

bool | True or False bool() 1 is True, 0 is False

str | string str() can make value into string

Enter value or string:

- string = input('') | variable is string = input()

- int = int(input('') | variable is first int, second int is int, input is enter value or string.

- float = float(input('') | first float is variable, second float is float, input is input value or string.

Math functions and constant:

- factorial(x) | is n! 7!

- ceil(x) | Return value/variable inside (). int will return int. float will return up to whole number.

- exp(x) | Returns e\*\*x - returns E raised to the power of x (E\*\*x). 'E' (approximately 2.718282)

- pow(x,y) | Returns x raised to the power y

- sprt(x) | Returns the square root of x

- trunc(x) | Return value/variable inside () round down

- pi | (3.14159...) to calculate circle

- e | mathematical constant e (2.71828...)

python example (after 'math.' put the math function, x is return)

import math

math.sqrt(x)

print('' + str())

+ str() | inside '()' parentheses are variable. inside the parenthese contain value and make into string.

Escape sequence

\ if there is print (' ' ' ') syntax error. add in \\ | (' \'\' ')

\n new line

\t new tab

String formatting

({} {} .format()) | inside .format parenthese can put str, value or variable

{:f} .format(3.142)

output: 3.14200

{:.2f} .format(22.2259)

output: 22.23

default alignment is left

{:10} .format(test)

'test '

{:>10} right alignment

' test'

{:<10} left alignment | {:10} and {:>10} is the same

'test '

{:\_<10}'.format('test')

'\_\_\_\_\_\_test'

{:^10}'.format('test')

' test '

{:\*^10}'.format('test')

'\*\*\*test\*\*\*'

example

{:>9.2f}

{:s} | string

{:d} | decimal

{:f} | float

Week 3

name = 'Ngee Ann' | 01234567 | 'N' is name[0] | ' ' is name[4] | 'n' is name[-1]

name[0:4] 'Ngee' (start from 0 to 4) | name[:4] 'Ngee' (from beginning to 4) | name[5:] 'Ann' (start from 5 to end)

Other string operations: a = 'Hello' | b = 'Python'

+ | a+b display 'HelloPython' + is mean to join two variable together

a + ' ' + b display "Hello Python"

\* | a\*2 display 'HelloHello' \* is display a variable twice.

in | 'H' in a = True | but 'h' in a = False (a = 'Hello' only have Cap lock 'H' so it's True

There no 'h' small h in a)

not | 'M' not in a = True

Bulid in string function

name[len(name)-1]

a = 'Hello Python'

a.capitalize() display: 'Hello python' | only first letter is Caps the rest are small letters

a.lower() display: 'hello python'

a.upper() display: 'HELLO PYTHON'

a.find('on') display: 10 | 'on' is at 10 position onwards 'o' at [10]

a.find('ON') display: error | there is no 'ON' in a

a.replace('o','z') display: 'Hellz Pythzn' | replace all 'o' to 'z'

a.alpha() display: False | .alpha() in the parenthese has no alphabetic/letter. it is empty.

a.split(' ') / (" ") display 'Hello','Python' | must have space in between parentheses. if not will have error message

input (12,32,12)

variable = variable.split( ‘ , ’ )

output

what is listing

Peter | 0 = P , e = 1 , t = 2 , e = 3 , r = 4 | number is the position of the alphabet

Peter, John, Mary, David | Peter = 0 , John = 1 , Mary = 2 , David = 3 | number are index of the name

Create listing

**Usually we assign the list to a variable name so that we can refer to the list subsequently:**

emptyList = [ ]

friendsList = [ 'Peter', 'John', 'Mary', 'David' ]

marksList = [ 89, 77, 55, 69 ]

Nested List: list in list

friendsList = ['Peter', ['John', 'Mary'], 'David']

matrix = [ [1,2,3], [4,5,6], [7,8,9] ] | list 1, 2, 3

Rojak list (list contain different element, string, float and integer)

mixedList = ['Peter', 100, 23.5, [10, 20]]

Basic Operators List

friends = [ ‘Peter’, ‘John’, ‘Mary’, ‘David’ ] | friends = [ 1 : 3 ] | output : [ ‘John’, ‘Mary’ ]

Basic Operators for List: list1 = [ 1, 2, 3, 4, 5] | list2 = [ ‘a’, ‘b’, ‘c’ ] | list3 = [ 1, 2, 3, 4, 5]

+ | list1 + list2 | output: [1, 2, 3, 4, 5, ‘a’, ‘b’, ‘c’]

In | 1 in list1 output: True (1 is in list1) | 1 in list2 output: False (1 is not in list2)

== | list1 == list2 output: False | list1 == list3 output: True

Basic Functions for List:

len() | len(list1) | output: 5 | len() can return number of element inside value, string and list

min() | min(list1) | output: 1 | min() is smallest element

max() | max(list1) | output: 5 | max() is largest element

Built – in List : letters = [ ‘a’, ‘b’ ]

append(x) | letters.append( ‘c’ ) | output: [ ‘a’, ‘b’, ‘c’ ] | add an element x , to the end of list.

extend(L) | letters.extend(letter) | output: ['a', 'b', 'c', 'a', 'b', 'c'] | basically duplicate element inside letter and add in the list.

Insert(i,x) | letters.insert(3, ’z’) | output: ['a', 'b', 'c', 'z', 'a', 'b', 'c'] | position is i , x is insert an item.

Built-in List methods : letters = ['a', 'b', 'c', 'z', 'a', 'b', 'c']

remove(x) | letters.remove(‘c’) | output: ['a', 'b', 'z', 'a', 'b', 'c'] | first item of x is removed from list.

letters.remove('d') | output: ValueError | Error occurs if x is not in list.

pop([i]) | letters.pop(2) 🡺 ‘c’ | output: ['a', 'b', 'z', 'a', 'b', 'c'] | Remove the item at the given position in the list and return it. (pop is to take and store away in storage)

letters.pop() ‘c’ | output: ['a', 'b', 'z', 'a', 'b'] | Removes and returns the last item in the list if the argument is not stated.

letters = ['a', 'b', 'a', 'b']

index(x) | letters.index('a') | output: 0 | Return the index in the list of the first item whose value is **x**.

letters.index('c') | output: ValueError | Error occurs if x is not in the list

count(x) | letters.count('a') | output: 2 | number of times x appears in list

reverse() | letters.reverse() | output: ['b', 'a', 'b', 'a'] | Reverse the elements of the list in place.

sort() | letters.sort() | output: ['a', 'a', 'b', 'b'] | sort the items of the place in order

clear() | letters.clear() | remove all items in list.

Week 4 If else

Single – selection statement

Pseudocode:  
IF temperature > 37.7 THEN

Display “Tom is having a fever”

ENDIF

If Statement – a block

* If there is more than one statement to execute when condition is true

if *condition*:

*true\_statement\_1*

*true\_statement\_2*

*:*

*true\_statement\_n*

Double – selection statement (If, else)

Pseudocode:

IF temperature is greater than 37.5 THEN

display “Tom is having a fever.”

ELSE

display “Tom is not having a fever.”

display “He can perform his normal duties.”

ENDIF

input:

import random

num1 = random.randint(0,100) | pick random number from 1 to 100

Compound Conditions

and | x>y and x>z

(temperature >= 20) and (temperature <= 25) or 20 <= temperature <= 25

or | x!=y or x!=z

(temperature < 20) or (temperature > 25)

not | not (x==y)

(temperature < 20) or (temperature > 25)

is equivalent to

not((temperature >= 20) and (temperature <= 25))

cond 1 | cond2 | cond1 and cond2 | cond1 or cond2 | not cond1

false | false | false | false | true

false | true | false | true | true

true | false | false | true |false

true | true | true | true |false