Python 3

Contents

[## Compiler 2](#_Toc197093001)

[## Vars / Data 2](#_Toc197093002)

[## Input 2](#_Toc197093003)

[## Operations 2](#_Toc197093004)

[## Bools 2](#_Toc197093005)

[## Conditionals 3](#_Toc197093006)

[## Loops 3](#_Toc197093007)

[## Lists 3](#_Toc197093008)

[## Logic 3](#_Toc197093009)

[## Bubble Sort 4](#_Toc197093010)

[## Functions / Methods 5](#_Toc197093011)

[## Scopes 5](#_Toc197093012)

[## Return 5](#_Toc197093013)

[## Recursion 6](#_Toc197093014)

[## Tuples 6](#_Toc197093015)

[## Dictionaries 6](#_Toc197093016)

[## Modules 7](#_Toc197093017)

[## Random 7](#_Toc197093018)

[## Error Handling 8](#_Toc197093019)

[## String operations 8](#_Toc197093020)

[## Scoreboard Interface 10](#_Toc197093021)

[## File Handling 13](#_Toc197093022)

# ## Compiler

# Online Python compiler (interpreter) to run Python online.

# Write Python 3 code in this online editor and run it.

# ## Vars / Data

pi = 3.14159

name = "Name"

numS = "123"

numI = int(numS)

# ## Input

print(type(numI + 12), float(numS), numI)

age = input("enter your age")

name = input("Enter your name: ")

print("Hello, ", str(name), "!")

get num from input: a = float(input('enter your number: '))

# ## Operations

s = 9, t = 2;

integer division s // t = 4

exp s \*\* t = 81

string concat 'a' + 'bc'

# ## Bools

True, False

&& -> and || -> or xor (exclusive or)

# ## Conditionals

eq == neq != leq <= geq >= less < greater >

if condition :

elif condition :

else :

# ## Loops

iterative: for i in [1, 2, 3] : for i in range(1, 101) -> range [1, 101) :

for i in ‘string’ : print i -> ‘s’, ‘t’, ‘r’, ‘i’, ‘n’, ‘g’ // with \n

conditional: i = 1 ; i = 0 ; string = ‘string’

while i <= 10 : while i <= **len**(string) mm

print (i) print (string[i])

i += 3 i += 1

# ## Lists

myList = [1, 2, 3]

fruits = [ ‘apple’, ‘banana’, ‘pear’, ‘strawberry’]

print(fruits[:3]) print(fruits[::-1]) – reverse order

fruits = **list**(‘apple’, ‘banana’, 1) ‘apple’ in fruits -> True

fruits[2] = ‘pear’ len(fruits) fruits.sort() fruits .reverse()

fruits.append(‘berry’) fruits.remove(‘apple’) fruits.pop(3)

# ## Logic

Complement: ~ Binary: bin(1) -> ‘0b1’

Bitwise And: & (12&23=4) Bitwise Or: | (12|23=31) Bitwise Xor: ^ (12^23=27)

Shift Left a << b Shift Right a >> b

# ## Bubble Sort

O(n^2)

numbers = [3, 7, 11, 8, 6, 0, 15]

print(numbers)

bubbleLength = len(numbers)

while (bubbleLength != 1) :

for i in range(0, bubbleLength - 1) :

if (numbers[i] > numbers[i + 1]) :

temp = numbers[i]

numbers[i] = numbers[j]

numbers[j] = temp

bubbleLength -= 1

print(numbers)

### Works!

OR

length = len(numbers)

for i in range(length)

for j in range(0, length - i - 1) :

if numbers[j] > numbers[j+1] :

numbers[j], numbers[j+1] = numbers[j],numbers[j+1]

# ## Functions / Methods

# -> can set default value with ‘=’, non-defaults go first tho

def myFunction(endl, string = ‘you have not sent a parameter’) :

for char in string:

print(char, endl)

string p = **(‘This is our function’)**

myFunction(‘\n’, ‘hello’)

# ## Scopes

global (outside function) / local (declare in function, can make global with keyword)

x = 10

def function():

global x

x = 20

# ## Return

# can only return one thing

def reverse():

return print(string[::-1])

# anything in-function after return won’t be executed

reversed = reverse(‘hello’)

print(‘data ’, reversed)

# ## Recursion

def iterFactorial(n): def factorial(n):

result = 1 if n < 1:

for i in range(n, 0, -1): return 1

result \*= i else :

return result return n \* recursion(n - 1)

# ## Tuples

[1, 2, 3] -> (1, 2, 3) curve parenthesis, comma separated

#can’t change once assigned

tuple1 = (1, 2, 3, 4, 5) print(tuple1[1]) -> YES tuple1[2] = 0 -> NO

list1 = list(tuple1)

list1[3] = 4

tuple1 = tuple(list1)

# ## Dictionaries

like json

employee = {

‘name’ : ‘Jerry’,

‘age’ : 39,

‘salary’ : ‘$34000’

}

print(employee[‘name’], employee[‘age’])

for key in employee : -> gets keys for val in employee.values() -> gets values

print(key + “ : ” + str(employee[key]))

# ## Modules

A Module is a section of code, self-contained and scalable.

A Package is a collection of Modules, has an init.py file

{strings.py}

# A module for string operations

def reverseString(string):

return string[::-1]

import strings

print(strings.reverseString(‘olleh’))

print(\_\_name\_\_)

# ## Random

randint(1, 10) <- random int within range

import random

score = 10

randomNumber = random.randint(1,10)

while True:

userNumberInput = int(input(‘Guess: ’))

if userNumberInput == randomNumber :

print(“You Guessed it!. Score: ” + str(score))

break

else:

print(‘Better luck next time!’)

score -= 1

# ## Error Handling

try :

print(‘opened’)

a = int(input(‘~ ’))

except ValueError :

print(‘invalid user input’)

except TypeError :

print(‘type error’)

except KeyboardInterrupt : <- Ctrl+C

print(‘keyboard interrupt’)

except ZeroDivisionError : <- Ctrl+C

print(‘zero div’)

except Exception as error :

print(‘User Error: ’ + str(error))

finally:

print(‘closed’)

# ## String operations

string.isupper() string.islower() string.lower() string.upper()

string.swapcase() string.isdigit() string.replace(‘1’, ‘Hello’)

string.split(‘o’) ‘e’ in string (T/F) f’Hello my name is {string}’

def answer(string) :

alphabets = ‘’

result = 0

for char in string :

if char.isdigit() :

result += int(char)

else :

alphabet += char

return(alphabets, result)

print()

# ## Scoreboard Interface

from typing import List

# Write any import statements here

def getMinProblemCount(N: int, S: List[int]) -> int:

max = 0

odds = False

for i in S :

if i > max :

max = i

if not odds and i % 2 == 1 :

odds = True

problems = max // 2

if odds :

problems += 1

# I don't want peace, I want problems, always

return problems

JAVA 8

// Write any import statements here

class Solution {

public int getMinProblemCount(int N, int[] S) {

// 1. Look for largest number in array

// 2. num problems = max / 2 + odds

// 3. if there are odds, split, add one.

boolean odds = false;

int max = 0;

for (int i : S) {

if (i >= max) max = i;

if (!odds && i % 2 == 1) odds = true;

}

int problems = max / 2;

problems = odds ? problems + 1 : problems;

// Write your code here

return problems;

}

}

from typing import List

# Write any import statements here

def getMinimumDeflatedDiscCount(N: int, R: List[int]) -> int:

# Error Case Handling

if len(R) > R[len(R) - 1] : return -1

return recursiveAuxDisc(0, R)

def recursiveAuxDisc(N: int, R: List[int]) -> int :

length = len(R)

# Error Case Handling

if length > R[length - 1] : return -1

# Base Case

if length == 1 : return N

if R[length-2] >= R[length-1] :

R[length-2] = R[length-1] - 1

N = N + 1

# Recursive step

return recursiveAuxDisc(N, R[:length-1])

# ## File Handling

Can open in modes: w, w+, a, a+, r, r+, rb, wb.

Modes: wb, w, w+, a, a+, will create a file if it doesn’t exist.

w - write, a - append, r – read, b – binary.

w+ - write and read, a+ - append and read

with open(‘files/data.txt’, ‘r’) as file

file.read() file.readline() file.readlines() file.readlines(5) file.write()

#binary file.load() file.dump()

# ## Classes

class {}