CC - AI

Programación para la Computación Científica - IA

Python for Programmers

II sesion

Universidad Sergio Arboleda

Prof. John Corredor

Input values

```
valor = input()

# Podemos mostrar un mensaje antes de leer el valor

valor = input("Introduce un valor: ")

# La función int() de entero, devuelve un número entero a partir de una cadena

valor = int(valor)

3500

valor + 1000  # Ahora ya es operable
```

Challenge 01

1) Make a program that reads 2 numbers per keyboard and determines the following aspects (it is enough to show True or False):

- If the two numbers are equal
- If the tw
- If the fir
- If the se

```
n1 = float( input("Introduce el primer número: ") )
n2 = float( input("Introduce el segundo número: ") )

print("¿Son iguales? ", n1 == n2)
print("¿Son diferentes?", n1 != n2)
print("¿El primero es mayor que el segundo?", n1 > n2)
print("¿El segundo es mayor o igual que el primero?", n1 <= n2)
```

Challenge 02

- 2) Using logical operators, determine if a text string entered by the user has a length greater than or equal to 3 and in turn is less than 10 (it is sufficient to show True or False)

```
cadena = input("Escribe una cadena: ")
print("¿La longitud de la cadena es mayor o igual que 3 y menor que 10?",
len(cadena) >= 3 and len(cadena) < 10 )
```

Challenge 03

3) Make a program that complies with the following algorithm using

```
magic_number = 12345679
user_number = int(input("Introduce un número del 1 al 9: "))
user_number *= 9
magic_number *= user_number
print("El número mágico es:", magic_number)
```

Multiply the user_number by 9 in itself
Multiply the magic_number by the user_number itself
Finally it shows the final value of the magic_number per screen

Challenge 04

```
n1 = float(input("Introduce un número: ") )
n2 = float(input("Introduce otro número: ") )
opcion = 0

print("¿Qué quieres hacer? \n1) Sumar los dos números\n2) Restar los dos
números\n3) Multiplicar los dos números")
opcion = int(input("Introduce un número: ") )

if opcion == 1:
   print("La suma de",n1,"+",n2,"es",n1+n2)
elif opcion == 2:
   print("La resta de",n1,"-",n2,"es",n1-n2)
elif opcion == 3:
   print("El producto de",n1,"*",n2,"es",n1*n2)
else:
   print("Opción incorrecta")
```

Challenge 05

Make a program that asks the user how many numbers he wants to enter. Then read all the numbers and perform an arithmetic mean:

```
suma = 0
numeros = int(input("¿Cuántos números quieres introducir? ") )
for x in range(numeros):
    suma += float(input("Introduce un número: ") )
print("Se han introducido",numeros,"números que en total han sumado",suma,"y la media es",suma/numeros)
```

Challenge 06

Using the range () function and conversion to lists generates the following lists dynamically:

```
print( list( range( 0, 11 ) ) )
print( list( range( -10, 1 ) ) )
print( list( range( 0, 21, 2 ) ) )
print( list( range( -19, 0, 2 ) ) )
print( list( range( 0, 51, 5 ) ) )
```

All multiple numbers from 5 from 0 to 50 [0, 5, 10, ..., 50]

Challenge 07

Given two lists, you must generate a third with all the elements that are repeated in them, but you must not repeat any element in the new list:

```
lista_1 = ["h","o","l","a"," ", 'm',"u","n","d","o']
lista_2 = ["h","o","l","a"," ", 'l',"u","n","a']

lista_3 = []

for letra in lista_1:
    if letra in lista_2 and letra not in lista_3:
        lista_3.append(letra)

print(lista_3)
```

Challenge 08

Given two lists, you must generate a third with all the elements that are repeated in them, but you must not repeat any element in the new list:

```
lista_1 = ["h","o","l","a"," ", 'm',"u","n","d","o']
lista_2 = ["h","o","l","a"," ", 'l',"u","n","a']

lista_3 = []

for letra in lista_1:
    if letra in lista_2 and letra not in lista_3:
        lista_3.append(letra)

print(lista_3)
```

Bibliotecas

Una de las ventajas de Python es la gran cantidad de objetos y funciones que hay disponibles en forma de bibliotecas (o módulos). Para acceder a las funciones de una biblioteca, tenemos varias opciones:

1. Importar el módulo en cuestión y después acceder a la función mediante el nombre de la biblioteca. Así, para utilizar la función sin, que está en math, haríamos:

```
import math
print math.sin(2.0)
```

2. Importar la función del módulo y utilizarla directamente:

```
from math import sin
print sin(2.0)
```

3. Importar todas las funciones del módulo y utilizarlas directamente:

```
from math import *
print sin(2.0)
```

Bibliotecas

Si queremos importar varios módulos en una sola sentencia, podemos separarlos por comas:

```
import sys, math, re
```

Algunas de las bibliotecas que pueden ser útiles para la asignatura son:

sys	da acceso a variables del intérprete y a funciones que interactúan con él.
types	define nombres para los tipos de objetos del intérprete.
string	operaciones sobre cadenas.
re	expresiones regulares.
math	funciones matemáticas.
tempfile	creación de ficheros temporales.

Bibliotecas

La biblioteca sys

exit(n) aborta la ejecución y devuelve el valor n a la shell.

argv lista de los argumentos que se le pasan al programa.

Ejemplo: el siguiente programa escribe los argumentos con los que se le llama:

```
import sys
for i in sys.argv:
print i
```

stdin, stdout, stderr objetos de tipo fichero que corresponden a la entrada estándar, salida estándar y salida de error.

La biblioteca types

Define nombres para los tipos que proporciona el intérprete. Por ejemplo: NoneType, para el tipo de None; IntType, para el tipo entero; LongType, para el tipo entero largo; FloatType, el tipo real; etc.

La siguiente función recibe como parámetros una lista y un segundo parámetro. Si este pará metro es un entero, borra de la lista el elemento de la posición correspondiente; si no es entero, elimina el primer elemento igual a él:

```
from types import *
def delete(list, item):
    if type(item) is IntType:
        del list[item]
    else:
        list.remove(item)
```

La biblioteca string

Esta biblioteca define diversas variables y funciones útiles para manipular cadenas. Muchas de sus funciones han sido sustituidas por métodos sobre las cadenas.

Entre las variables tenemos:

digits	’0123456789’.
hexdigits	’0123456789abcdefABCDEF’.
letters	Letras minúsculas y mayúsculas.
lowercase	Letras minúsculas.
octdigits	’01234567’.
uppercase	Letras mayúsculas.
whitespace	Blancos (espacio, tabuladores horizontal y vertical, retorno de carro, nueva línea y nueva página)

La biblioteca string (funciones)

atof(s) convierte la cadena s en un flotante.
atoi(s,[b]) convierte la cadena s en un entero (en base b si se especifica).
atol(s,[b]) convierte la cadena s en un entero largo (en base b si se especifica).
lower(s) pasa s a minúsculas.
join(l[,p]) devuelve la cadena que se obtiene al unir los elementos de la lista l separados por blancos o por p si se especifica.
split(s[,p]) devuelve la lista que se obtiene al separar la cadena s por los blancos o por p si se especifica.
strip(s) devuelve la cadena que se obtiene al eliminar de s sus blancos iniciales y finales.
upper(s) pasa s a mayúsculas.

La biblioteca math

Define una serie de funciones matemáticas idénticas a las de la librería estándar de C: acos(x), asin(x), atan(x), atan2(x, y), ceil(x), cos(x), cosh(x), exp(x), fabs(x), floor(x), fmod(x, y), frexp(x), hypot(x, y), ldexp(x, y), log(x), log10(x), modf(x), pow(x, y), sin(x), sinh(x), sqrt(x), tan(x), tanh(x).

También define las constantes pi y e.

Representaciones de Expresiones Sencillas

Ejemplo

```
#!/usr/bin/env python
class AST:                    #-----------------------------------------------------
        def __init__(self): pass
        def mostrar(self): pass                    # muestra la expresión utilizando paréntesis
        def numero_operaciones(self): pass  # muestra el número de operaciones de la exp.
        def interpreta(self): pass                 # evalua la expresión
```

Representaciones de Expresiones Sencillas (2)

Ejemplo

```
class Numero(AST): #-----------------------------------------------
        def __init__(self, valor):
                self.valor = valor
        def mostrar(self):
                return str(self.valor)
        def numero_operaciones(self):
                return 0
        def interpreta(self):
                return self.valor
```

Ejemplo

```
class Operacion(AST): #-----------------------------------
    def __init__(self, op, izda, dcha):
        self.op = op
        self.izda = izda
        self.dcha = dcha
def mostrar(self):
    return '(' + self.izda.mostrar() + self.op + self.dcha.mostrar() + ')'
def numero_operaciones(self):
    return 1 + self.izda.numero_operaciones() + self.dcha.numero_operaciones()
def interpreta(self):
    if self.op=="+":
        return self.izda.interpreta() + self.dcha.interpreta()
    else: # si no, debe ser '*'
        return self.izda.interpreta() * self.dcha.interpreta()
```

Ejemplo

```
# Programa principal ------------------------------------------------------------
# Introducimos el árbol de la expresión 4*5 + 3*2
num1=Numero(4)
num2=Numero(5)
num3=Numero(3)
num4=Numero(2)
arbol1=Operacion('*',num1,num2)          # 4*5
arbol2=Operacion('*',num3,num4)          # 3*2
arbol_final=Operacion('+',arbol1,arbol2) # arbol1+arbol2
```

Ejemplo

```
# Accedemos al árbol de tres formas diferentes mediante funciones miembro
print 'El árbol contiene la expresión:', arbol_final.mostrar()
print 'El árbol contiene en total %d operaciones' % arbol_final.numero_operaciones()
print 'La expresión se evalua como:', arbol_final.interpreta()
```

Al ejecutar el programa anterior, se construye un árbol con la expresión $4 * 5 + 3 * 2$ y se obtiene la siguiente salida:

```
El árbol contiene la expresión: ((4*5)+(3*2))
El árbol contiene en total 3 operaciones
La expresión se evalua como: 26
```

Challenge 09

Write a program looks this Expected Output :

- Addition of two complex numbers : (7-4j)
- Subtraction of two complex numbers : (1+10j)

```
print("Addition of two complex numbers : ",(4+3j)+(3-7j))
print("Subtraction of two complex numbers : ",(4+3j)-(3-7j))
print("Multiplication of two complex numbers : ",(4+3j)*(3-7j))
print("Division of two complex numbers : ",(4+3j)/(3-7j))
```

Challenge 10

Write a Python program to print a random sample of words from the system dictionary.

Expected Output :

```
import random
with open('/usr/share/dict/words', 'rt') as fh:
    words = fh.readlines()
words = [w.rstrip() for w in words]
for w in random.sample(words, 7):
    print(w)
```

```
sulkiest
whisper's
downturns
```

```
import random
def display_intro():
    title = "** Math Test **"
    print("*" * len(title))
    print(title)
    print("*" * len(title))
def display_menu():
    menu_list = ["1. Addition", "2. Subtraction", "3. Multi", "4. Integer Division", "5. Exit"]
    print(menu_list[0])
    print(menu_list[1])
    print(menu_list[2])
    print(menu_list[3])
    print(menu_list[4])
```

Your score is 100.0%. Thank you.

```
def display_separator():
    print("-" * 24)
def get_user_input():
    user_input = int(input("Enter your choice: "))
    while user_input > 5 or user_input <= 0:
        print("Invalid menu option.")
        user_input = int(input("Please try again: "))
    else:
        return user_input
def get_user_solution(problem):
    print("Enter your answer")
    print(problem, end="")
    result = int(input(" = "))
    return result
```

Your score is 100.0%. Thank you.

```
def check_solution(user_solution, solution, count):
    if user_solution == solution:
        count = count + 1
        print("Correct.")
        return count
    else:
        print("Incorrect.")
        return count
def display_result(total, correct):
    if total > 0:
        result = correct / total
        percentage = round((result * 100), 2)
    if total == 0:
        percentage = 0
    print("You answered", total, "questions with", correct, "correct.")
    print("Your score is ", percentage, "%. Thank you.", sep = "")
```

```
def menu_option(index, count):
    number_one = random.randrange(1, 21)
    number_two = random.randrange(1, 21)
    if index is 1:
        problem = str(number_one) + " + " + str(number_two)
        solution = number_one + number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
    elif index is 2:
        problem = str(number_one) + " - " + str(number_two)
        solution = number_one - number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
    elif index is 3:
        problem = str(number_one) + " * " + str(number_two)
        solution = number_one * number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
```

```
    else:
        problem = str(number_one) + " // " + str(number_two)
        solution = number_one // number_two
        user_solution = get_user_solution(problem)
        count = check_solution(user_solution, solution, count)
        return count
def display_result(total, correct):
    if total > 0:
        result = correct / total
        percentage = round((result * 100), 2)
    if total == 0:
        percentage = 0
    print("You answered", total, "questions with", correct, "correct.")
    print("Your score is ", percentage, "%. Thank you.", sep = "")
```

```
  def main():
    display_intro()
    display_menu()
    display_separator()

    option = get_user_input()
    total = 0
    correct = 0
    while option != 5:
        total = total + 1
        correct = menu_option(option, correct)
        option = get_user_input()

    print("Exit the quiz.")
    display_separator()
    display_result(total, correct)
main()
```

References

- Kernighan, Brian W., and Dennis M. Ritchie. The C Programming Language. Vol. 2. Englewood Cliffs: prentice-Hall, 1988.
- Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. Operating System Concepts. Vol. 8. Wiley, 2013.
- https://planningtank.com/computer-applications/data-processing-cycle
- https://www.talend.com/resources/what-is-data-processing/
- https://peda.net/kenya/ass/subjects2/computer-studies/form-3/data-processing/dpc2
- https://www.academia.edu/38210518/What_is_Data_Processing
- https://www.studocu.com/en/document/polytechnic-university-of-the-philippines/information-and-communication-technology/lecture-notes/data-processing-lectures-in-data-processing/3167716/view
- http://download.nos.org/srsec330/330L2.pdf