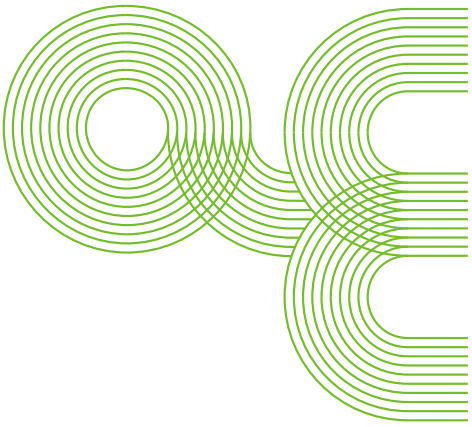
Universidade Vigo

Informática – Prácticas



Escola de Enxeñaría Aeronáutica e do Espazo
Pavillón Manuel Martínez-Risco
Campus universitario
32004 Ourense

http://aero.uvigo.es
mailto:aero.info@uvigo.es

Referencia: 1.0
Documento: practicas-inf
Fecha: 5 de diciembre de 2016
Páginas: 9

Índice

1. Primeros pasos

Objetivos: Trabajar con python. Conocer tipos de datos simples. Realizar cálculos simples con tipos de datos simples. Introducir y visualizar datos con la consola. Usar cadenas de caracteres.

1. Ejecuta IDLE (Python 3.5) , escribe la sentencia `print("hola mundo")`. Guarda tu fichero como `P1_ejercicio1.py`, ejecuta el programa.
2. Prueba el funcionamiento del siguiente programa donde se muestran algunos de los tipos de datos que ofrece python:

```
edad = 20
precio = 49.5
dni = "12345678Z"
beca = True
complejo = 1.3 + 2.5j
print(edad, precio, dni, beca, complejo)
```

   Modifica el programa anterior cambiando el valor de alguna de las variables e imprimiéndolas de nuevo por pantalla. Comprueba que también es posible cambiar el tipo de una variable asignándole un valor de otro tipo distinto.

3. Crea un programa en el que se realicen las siguientes operaciones :
   a) Calcula una nota media de dos números decimales.
   b) Visualiza por pantalla esa nota
   c) Para calcular el precio final de un artículo, asigna a una variable un precio base por ejemplo de 100 euros y a otra variable el iva del 21 %, calcula el precio final de un producto precio base + iva.
   d) Visualiza por pantalla ese precio.
   e) Asigna a una variable una edad. En otra variable almacena si es mayor de edad o no.
   f) Visualiza por pantalla ese resultado.
   g) Calcula el resto de una división y visualízalo por pantalla.
   h) Calcula una potencia y visualiza su valor por pantalla.
   i) Asigna a dos variables peso y altura, en la misma línea de código. Visualízalas por pantalla.
4. En el siguiente programa se muestran ejemplos del tratamiento de cadenas de texto en python. Analiza la sintaxis de los diferentes operadores y funciones con cadenas.

```
url_escuela = "aero.uvigo.es"
print (url_escuela[2])
print (url_escuela[5:10])
usuario = "rlaza"
dominio = "uvigo.es"
print(usuario + "@" + dominio)
linea = 80 * "-"
print(linea)
dni = "12345678Z"
print(len(dni))
codigo = "OU-" + str(32000)
print(codigo)
print(codigo.lower())
ciudad = "Ourense"
print(ciudad.upper())
print(url_escuela.split("."))
print(url_escuela.split(".")[1])
```

```
print(url_escuela.find("."))
print(url_escuela.replace("."," "))
```

---

5. Completa este programa para que calcule la letra de un dni:

```
letras_dni = "TRWAGMYFPDXBNJZSQVHLCKE"
dni = ..............
posicion = dni%23
letra_dni = .........
NIF = ..............
print("NIF = ",NIF)
```

---

6. Escribe un programa que solicite por teclado grados fahrenheit y muestre por pantalla su valor en grados Celsius. Para ello debes aplicar la siguiente fórmula: `grados_celsius = ( grados_fahrenheit - 32 ) / 1.8`
7. Escribe un programa en python para leer nombre, apellidos y dni de una persona y mostrarlos por pantalla.
8. Modifica el anterior programa para mostrar los datos por pantalla utilizando el método format.
9. Escribe un programa que lea el nombre y los apellidos de una persona y obtenga (y visualice) su correo electrónico, teniendo en cuenta que el nombre de usuario se obtiene como inicial del nombre, seguida del segundo apellido (más `@alumnos.uvigo.es`).
10. Escribe un programa que lea las coordenadas x e y de un punto y las visualice por pantalla en la forma (x,y).
11. Escribe un programa para calcular la distancia entre dos puntos.
12. Escribir un programa en python que calcule el perímetro y el área de una circunferencia a partir de su radio.

2. Primeros algoritmos y entrada/salida

Objetivos: Realizar algoritmos simples con flujo de control y bucles.

1. Escribe un programa que lea una calificación numérica por teclado y visualice su nota textual.
2. Escribe un programa que lea un número y visualice si es par o no.
3. Escribe un programa que lea la edad de una persona y muestre por pantalla si es mayor de edad o no.
4. Escribe un programa para calcular el mayor de dos números. Repite el programa pero con tres números.
5. Escribe un programa que acepte tres parámetros: dos números y un carácter. Si el carácter es + los números se suman, con - se restan, con * se multiplican, con / se dividen, y con ^ se eleva el primero al segundo.
6. Escribe un programa que solicite por teclado un número e imprima por pantalla los cuadrados de los números desde 1 hasta el número introducido.
7. Escribe un programa que calcule la media de una serie de notas introducidas por teclado.
8. Escribe un programa que calcule el mayor y el menor de una serie de números introducidos por teclado.
9. Escribe un programa que muestre por pantalla la tabla de multiplicar de un número introducido por teclado.
10. Modifica el programa anterior usando una la sentencia format.
11. Escribe un programa para calcular si un número es primo.
12. Escribe un programa para mostrar los cuadrados de los números del 1 al 100. Otro Igual pero sólo de los números pares entre el 1 y el 100.

3. Funciones

Objetivos: Usar funciones en python. Refuerzo de bucles. Aumento entrada/salida. Comprobación automática.

1. Prueba el siguiente programa que utiliza una función para calcular la nota final a partir de tres notas (ponderando su valor).

```
def calcula_nota(test1, test2, practicas):
    return test1*0.35 + test2*0.35 + practicas*0.3

print(calcula_nota(7.5, 8.5, 7))
print(calcula_nota(4.7, 7.2, 6.8))
```

2. Observa como el siguiente programa utiliza además una función que devuelve más de un elemento para leer por teclado las tres notas:

```
def pide_notas():
    prueba1 = float(input("Nota 1ª prueba: "))
    prueba2 = float(input("Nota 2ª prueba: "))
    practicas = float(input("Nota prácticas: "))
    return prueba1, prueba2, practicas

def muestra_nota(test1, test2, practicas):
    final = test1*0.35 + test2*0.35 + practicas*0.3
    print("Nota final: {0:.2f}".format(final))

nota1, nota2, nota3 = pide_notas()
muestra_nota(nota1,nota2,nota3)
```

3. El siguiente programa repite la entrada y visualización de notas hasta que se indica que se quiere finalizar

```
def pide_notas():
    prueba1 = float(input("Nota 1ª prueba: "))
    prueba2 = float(input("Nota 2ª prueba: "))
    practicas = float(input("Nota prácticas: "))
    return prueba1, prueba2, practicas

def muestra_nota(test1, test2, practicas):
    final = test1*0.35 + test2*0.35 + practicas*0.3
    print("Nota final: {0:.2f}".format(final))


repetir = "si"
while (repetir != "no"):
    nota1, nota2, nota3 = pide_notas()
    muestra_nota(nota1,nota2,nota3)
    repetir = input("Quiere seguir (si/no) ")
    repetir = repetir.lower()
else:
    print("\nFin del programa")
```

4. Modifica el programa anterior de tal manera que alguna entrada de un valor negativo como nota termina el bucle (claro, sin calcular ninguna media).
5. Modifica la función `muestra_nota()` que coja ponderaciones por defecto como argumentos. Experimenta con la llamada.
6. Escribe una función para calcular la secuencia de Collatz de un número entero positivo a_0, donde tenemos como secuencia: $a_n = 1/2 \cdot a_{n-1}$, si a_n es par, y $a_n = 3 \cdot a_{n-1} + 1$, si a_n es impar. La secuencia termina si llegamos al número 1. Por ejemplo, para $a_0 = 3$ tenemos la secuencia: 3, 10, 5, 16, 8, 4, 2, 1. (Nota: no se sabe si se llega siempre a 1 dado cualquier número.).
7. Escribe una función para calcular el factorial de un número y pruébala en un programa.
8. Escribe una función para calcular si un año es bisiesto y pruébala en un programa.
9. Escribe un programa para jugar al juego de piedra, papel o tijera. El programa debe generar aleatoriamente un valor y preguntar al usuario por el suyo. Gana el que primero llegue a tres victorias.
10. Escribe utilizando funciones un programa para encontrar las raíces de una ecuación de segundo grado ($ax^2 + bx + c = 0$). Pide los coeficientes como valores flotantes al usuario.
11. Escribe un programa que utilice una función para convertir coordenadas polares a rectangulares.
12. Escribe un programa que utilice funciones para convertir coordenadas cilíndricas a rectangulares y al revés. Usa tus funciones para comprobar automáticamente tu implementación ya que $cyl2rec(rec2cyl(x,y,z)) = (x,y,z)$, o no?

4. Estructuras de datos simples

Objetivos: Usar estructuras de datos simples como listas, tuplas, conjuntos y diccionarios en python. Refuerzo de bucles y funciones. Aumento entrada/salida con ficheros. Retocar cadenas de caracteres.

1. Crea un fichero con una lista de asignaciones de pasajeros de asientos donde en cada línea aparece primero nombre y luego identificador del asiento separados por coma, por ejemplo:

   ```
   Maria, 2A
   Manuel, 21D
   Dorotea, 14F
   Samuel, 10C
   ```

   Escribe un programa que lea tal fichero y lo convierta automáticamente a un fichero donde en la primera línea aparecen todos los asientos ocupados y en la segunda línea los nombres en el orden según sus asientos.

2. Aumenta el programa del apartado anterior a un pequeño sistema de reservas para un avión A320. El programa debe actuar con un menu principal donde la entrada de una letra selecciona un módulo de actuación, ejemplos `r` activa el módulo de reserva, `l` activa el módulo de generar lista, `q` sale del programa.

   El menu debe contener por lo menos los seguientes módulos:

   - leer un fichero (con formato como en el apartado arriba)
   - listar todos los asientos acupados
   - añadir un reserva con un nombre y un asiento (ojo, solo un pasajero por asiento, y solo identificadores de asientos posibles)
   - guardar las reservas en un fichero
   - borrar una reserva (o bien con nombre o bien con asiento)
   - mostar número de asientos todavía libres
   - aumenta con funcionalidades que te resultan útiles (tales pueden ayudarte también para la depuración del programa)

   Para no complicarte la vida al principio, puedes asumir que las filas de los asientos tienen números consecutivos y que siempre hay la misma candidad por fila. Emplea bien y con sentido las estructuras de datos ya conocidas, por ejemplo, tuplas para unir pasajero con asiento, dicionarios para la ocupación del avión, conjuntos para los caracteres del menu, etc. Notarás que cuando más modular desarrollas el programa, más fácil es completar la tarea.

5. Funciones recursivas

Objetivos: Usar recursicón, observar ventajas y inconvenientes, análisis del flujo del programa, uso de estructuras de datos simples para almacenar resultados intermedios y aliviar el cálculo.

1. Escribe una función recursiva que calcule el factorial de un número positivo entero.
   - Realiza un búcle principal para visualizar resultados hasta que se quiere terminar (por ejemplo, introduciendo un número negativo).
   - Experimenta hasta que valores se puede calcular en un tiempo que consideras razonable.
2. Escribe una función recursiva que calcule un coeficiente binomial $\binom{n}{k}$ según la fórmula
   $$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$
   - Pide los dos valores n y k al usuario.
   - Realiza un búcle principal para visualizar resultados hasta que se quiere terminar (por ejemplo, introduciendo un número negativo para n).
   - Haz unos experimentos hasta que valores python es capaz de calculcar los coeficientes binomiales con este método en un tiempo que consideras razonable (¿cuál es la k respecto a la n que produce el coeficiente más grande?).
3. Escribe una función que calcule el n-ésimo valor de la serie de Fibonacci (recuerda, la serie comienza: 1,1,2,3,5,8,13,21,34,...). Quitando los dos casos iniciales $f_0 = 1$ y $f_1 = 1$, la fórmula recursiva es $f_n = f_{n-1} + f_{n-2}$.
   - Pide el valor n al usuario.
   - Realiza un búcle principal para visualizar resultados hasta que se quiere terminar (por ejemplo, introduciendo un número negativo para n).
   - Haz unos experimentos hasta que valores python es capaz de calculcar la serie con este método en un tiempo que consideras razonable.
4. Analiza las llamadas que realizan tus programas a las diferentes funciones, es decir, busca una respuesta a la pregunta: ¿Cuántas veces se llama a la función recursiva con un parámetro determinado?

   Mejora los programas anterioes considerablemente (respecto al tiempo de cálculo) usando una estructura de datos simple para almacenar valores ya calculados que solamente hay que buscar en la estructura en vez de recalcular.
5. Usa el módulo `timeit` y su documentación disponible en la red para medir más exacto el tiempo de cálculo de tus funciones.