Clase 5 – Funciones

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

05-09-2019

Resumen de la clase

```
while operación lógica:
    comando 1
```

- Se evalúa al “entrar” al ciclo, es decir, si la operación lógica se cumple, se ejecuta el ciclo por primera vez.
- Si la operación lógica se cumple, o es verdadera, el código se repite.
- Si la operación lógica es falsa, entonces el ciclo termina.

```
for i in range(a,b):
    comando 1
```

Donde:

- `i` es la variable que incrementa
- `a` es el valor inicial de `i`
- `i` aumenta hasta `b-1`

Contenidos

1. Ejercicios ciclos
2. Funciones
3. Ejercicios funciones

Ejercicio 1

- Escribe un programa donde un usuario ingrese un número *n.*
  - El número n **debe ser** mayor que 0.
  - El programa debe imprimir todos los números entre 1 y *n.*
  - Cuando se imprima un número par se debe imprimir esto en consola y cuando sea impar también.

P3 I1 2018-2

Pregunta 3

"Ruleta DCC" es el juego más exitoso de la tradicional "Fonda de Don Yadran", que organiza el Departamento de Ciencia de la Computación (DCC). Por supuesto, funciona en Python. Desafortunadamente se perdió el código, así es que te piden que lo vuelvas a crear.

Para participar, hay que apostar 100, por lo que el juego parte con 100 puntos. La ruleta genera solamente números aleatorios entre 1 y 36. El juego termina si sale 7 (en ese caso, quien estaba jugando se lleva el puntaje acumulado), si sale 13 (en ese caso, pierde todo), o si se le terminan los puntos.

Cuando empiezan a salir los números:

- Si detectas una secuencia de 3 números pares, se suma al puntaje todos los números de ruleta (no importa si es 7 o 13, ni si hay secuencias de números pares o impares) que salgan hasta que aparece uno de los tres números pares (el que también se suma). En el siguiente turno, la secuencia de números se reinicia.
- Si detectas una secuencia de 3 números impares, se resta al puntaje la suma de los siguientes 5 números de ruleta que salgan (no importa si es 7 o 13, ni si hay secuencias de números pares o impares). En el siguiente turno, la secuencia de números se reinicia.

Lo que debes hacer es implementar un programa que simule el juego "Ruleta DCC". A continuación se muestran algunos ejemplos del funcionamiento del programa:

Funciones

- Recordemos este ejemplo:

```
#ejemplo paquete random
import random
dado=random.randint(1,6)
numero_mult=int(input("¿Por qué número quieres multiplicar el resultado del
dado?"))
print(dado*numero_mult)
```

```
¿Por qué número quieres multiplicar el resultado del dado? 1200
6000
```

Funciones

- Recordemos este ejemplo:

```
#ejemplo paquete random
import random
dado=random.randint(1,6)
numero_mult=int(input("¿Por qué número quieres multiplicar el resultado del
dado?"))
print(dado*numero_mult)
```

```
¿Por qué número quieres multiplicar el resultado del dado? 1200
6000
```

- La primera línea servía para poder usar el *package* `random`
- En la segunda línea, generamos un número aleatorio entre 1 y 6 y lo agregamos a la variable dado.

Funciones

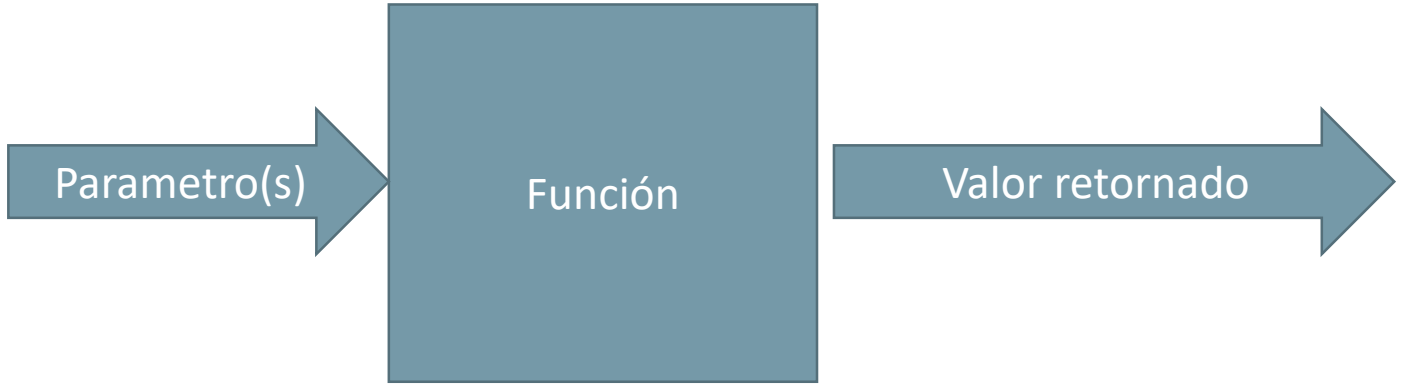
- Como sabemos que `dado` almacena el número aleatorio, eso quiere decir que `random.randint(1,6)` lo genera.

Funciones

- Como sabemos que `dado` almacena el número aleatorio, eso quiere decir que `random.randint(1,6)` lo genera.
- No obstante, `random` es el *package*. Entonces ¿Qué es `.randint(1,6)`?
- `randint(1,6)` es una **función** del *package* `random`.

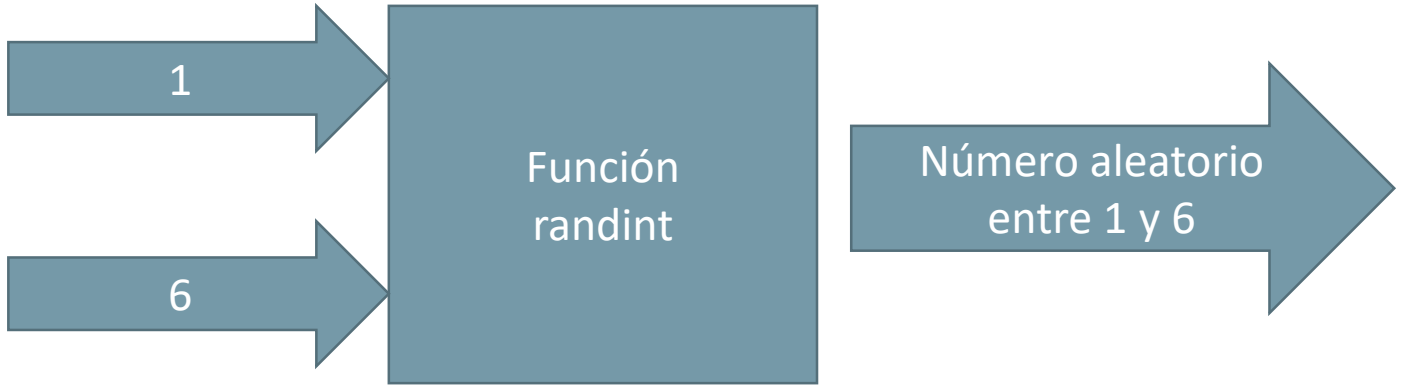
Funciones

- ¿Cómo podríamos caracterizar a una función?
- Las funciones en Python:
  - Pueden recibir parámetros
  - Hacen un cálculo o ejecución de cierto código
  - Devuelven un valor



Funciones

- Para el ejemplo que vimos anteriormente: `randint(1,6)`
- Los parámetros de la función son 1 y 6. La función se llama "`randint`". El valor retornado es un número aleatorio entre 1 y 6.



Funciones

- ¿Cómo podemos definir una función en Python?

```
def nombre_funcion(a,b,c):
    #ejecución de cierto código
    return (valor retornado)
```

- Veamos un ejemplo simple. Digamos que queremos crear una función que sume dos números.

Funciones

- **¿Por qué nos gustaría ocupar funciones?**
- No repetir código
- Separar funcionalidades de nuestro código
- Facilidad de comprensión
- Integración entre programas distintos

Funciones

- Para crear una función que sume dos números, ocuparemos el siguiente código:

```
def suma_de_dos_numeros(num1, num2):
    variable_con_valor_retornado = num1 + num2
    return variable_con_valor_retornado
```

break

5 min

Funciones

- Obviamente sumar dos números no es un buen ejemplo, dado que lo podemos hacer directamente con el operador +.
- Tenemos el siguiente problema: Pedro está aprendiendo geometría. En particular, quiere sacar el área de un círculo. Pedro tiene tres radios distintos, con medidas 2,3 y 4 cm. Además de sacar el área con esos radios, Pedro quiere saber si es que elevar el área de un círculo al cuadrado es equivalente a solo elevar al cuadrado los radios.

Funciones

```
#ejemplo área de círculo (sin funciones)
import math
r1 = 2
r2 = 3
r3 = 4
print("Los radios son:",r1,r2,r3)

area1 = r1**2*math.pi
area2 = r2**2*math.pi
area3 = r3**2*math.pi

print("Áreas:",area1,area2,area3)

print("Áreas al cuadrado:",area1**2,area2**2,area3**2)

area1_r1cuad=(r1**2)**2*math.pi
area2_r2cuad=(r2**2)**2*math.pi
area3_r3cuad=(r3**2)**2*math.pi
print("Áreas con radios al cuadrado:",area1_r1cuad,area2_r2cuad,area3_r3cuad)
```

Funciones

```
#ejemplo área de círculo (con funciones)
import math

def area_circulo(radio):
  return (radio**2)*math.pi

r1 = 2
r2 = 3
r3 = 4
print("Los radios son:",r1,r2,r3)

area1 = area_circulo(r1)
area2 = area_circulo(r2)
area3 = area_circulo(r3)

print("Áreas:",area1,area2,area3)

print("Áreas al cua-
drado:",area_circulo(r1)**2,area_circulo(r2)**2,area_circulo(r3)**2)

area1_r1cuad=area_circulo(r1**2)
area2_r2cuad=area_circulo(r2**2)
area3_r3cuad=area_circulo(r3**2)
print("Áreas con radios al cuadrado:",area1_r1cuad,area2_r2cuad,area3_r3cuad)
```

Funciones

- Notemos que ocupamos el *package* `math`.
- Este *package* tiene varias funciones que pueden ser muy útiles, algunas de ellas son:
  - math.factorial(x)
  - math.floor(x)
  - math.exp(x)
  - math.log(x[, base])
  - math.sqrt(x)
  - math.cos(x)
  - math.sin(x)
  - math.tan(x)
- Y muchas más, pueden verlas en este link.

Ejercicio 0

- Escriba un programa que imprima en consola los números de 1 a N, donde N está dado por el usuario. Además, cuando uno de los números es primo el programa debe escribirlo en la consola.

Ejercicio 0

- Escriba un programa que imprima en consola los números de 1 a N, donde N está dado por el usuario. Además, cuando uno de los números es primo el programa debe escribirlo en la consola.

```
#imprimir primos
def esPrimo(num):
  ret_bool = True
  if num == 1:
    ret_bool = False
  else:
    for i in range(2,num):
      #print(i)
      if num%i == 0:
        ret_bool = False
        break
  return ret_bool


num = int(input("Ingrese N"))
for i in range(1,num+1):
  if esPrimo(i):
    print(i,"es primo.")
  else:
    print(i)
```

Ejercicio 1

- Escriba un programa que:
  - Reciba el ancho y largo de un rectángulo.
  - Por medio de dos funciones distintas, pueda calcular el área y perímetro.

Ejercicio 1

```
def area_rectangulo(largo,ancho):
        return largo*ancho

def perimetro_rectangulo(largo,ancho):
        return largo*2+ancho*2

largo = int(input("Ingrese el largo del rectángulo"))
ancho = int(input("Ingrese el ancho del rectángulo"))

print("El perimetro es:",perimetro_rectangulo(largo,ancho))

print("El área es:",area_rectangulo(largo,ancho))
```

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
  - ¿Qué funciones deberíamos tener?

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
    - Combustible
    - Tamaño estanque

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
    - Combustible
    - Tamaño estanque

```
combustible = 0
tam_estanque = 40
```

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
    - Combustible
    - Tamaño estanque
  - ¿Qué funciones deberíamos tener?

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
    - Combustible
    - Tamaño estanque
  - ¿Qué funciones deberíamos tener?
    - ¿Podemos agregar combustible?
    - Agregar combustible
    - ¿Cuánto combustible queda?

Ejercicio 2

- Creemos un programa que simule el agregar combustible a un auto.
  - ¿Qué variables deberíamos tener?
    - Combustible
    - Tamaño estanque
  - ¿Qué funciones deberíamos tener?
    - ¿Podemos agregar combustible?
    - Agregar combustible
    - ¿Cuánto combustible queda?

```
combustible = 0
tam_estanque = 40

def puedoAgregarEstaCantidadDeCombustible(cantidad):

def agregarCombustible(cantidad):

def cuantoCombustibleQueda():
```

Ejercicio 2

- ¿Cómo sería el flujo del usuario?

Ejercicio 2

- ¿Cómo sería el flujo del usuario?

```
fin = input('''Ingrese alguna de las siguientes opciones:
                1) Terminar ejecución
                2) Consultar cuánto combustible queda
                3) Agregar combustible (indicar cantidad)
                4) Consultar tamaño del estanque\n''')
while fin != "1":
 #cierto código
```

Ejercicio 2

- ¿Cómo sería el flujo del usuario?

```
fin = input('''Ingrese alguna de las siguientes opciones:
                1) Terminar ejecución
                2) Consultar cuánto combustible queda
                3) Agregar combustible (indicar cantidad)
                4) Consultar tamaño del estanque\n''')
while fin != "1":
 #cierto código
```

Ejercicio 2

- ¿Cómo sería el flujo del usuario?

```
fin = input('''Ingrese alguna de las siguientes opciones:
                1) Terminar ejecución
                2) Consultar cuánto combustible queda
                3) Agregar combustible (indicar cantidad)
                4) Consultar tamaño del estanque\n''')
while fin != "1":
 #cierto código
```

https://repl.it/@FelipeLopez/IIC1103Funciones

Ejercicio 3: P2 I1 2018-2

Pregunta 2

Luego de las celebraciones de las fiestas patrias, notas que gastaste más dinero que el que tenías presupuestado gastar. Para que esto no te vuelva a ocurrir, decides partir registrando todos tus gastos con el fin de llevar una contabilidad y tener estadísticas de ellos. Por cada compra que hagas guardarás un registro de ella, el cual contiene una categoría (lo que compraste), el monto que gastaste, y el mes en que lo compraste. Como ya conoces las maravillas de `Python`, decides programar algunas funciones que te ayuden en esta tarea.

Para esto, puedes **importar** la librería `cuentas` que tiene las siguientes funciones **ya implementadas**:

- `obtener_cantidad(categoria)`: recibe un `string` con el nombre de la categoría y retorna un `int` con la cantidad de registros existentes para esa categoría. Puedes suponer que una categoría siempre tendrá al menos un registro. Por ejemplo, si `obtener_cantidad("bebida")` retorna 10, significa que hubo 10 registros de compras de bebidas.
- `obtener_mes(categoria, i)`: recibe un `string` con el nombre de la categoría y un `int` que representa el índice del registro (los cuales van desde 0 hasta la cantidad de registros existentes para esa categoría - 1). Retorna un `int` con el número del mes correspondiente al registro y categoría entregada. Por ejemplo, si `obtener_mes("bebida", 0)` entrega 9, significa que la primera bebida (índice `0`) fue comprada en el mes de septiembre (9).
- `obtener_monto(categoria, i)`: recibe un `string` con el nombre de la categoría y un `int` que representa el índice del registro (los cuales van desde 0 hasta la cantidad de registros existentes para esa categoría - 1). Retorna un `int` con el monto correspondiente al registro y categoría entregada. Por ejemplo, si `obtener_monto("bebida", 1)` entrega 15, significa que la segunda bebida (índice `1`) comprada costó 15.

Ejercicio 3: P2 I1 2018-2

Con el objetivo de generar estadísticas de tus gastos, **debes definir las siguientes funciones**:

a) **(20 pts)** `promedio_categoria(categoria)`: recibe un `string` con el nombre de la categoría y retorna un `float` con el promedio de gasto de los registros de esa categoría.

b) **(20 pts)** `total_mensual_categoria(mes, categoria)`: recibe un `int` que puede ir desde el 1 al 12 y representa el mes (1 equivale a enero y 12 a diciembre) y un `string` con el nombre de la categoría. Retorna un `int` con la cantidad total de dinero gastado en la categoría y mes dados.

c) **(20 pts)** `mensual_promedio_categoria(categoria)`: recibe un `string` con el nombre de la categoría y retorna un `float` con el gasto mensual promedio para la categoría dada.

Los gastos ya vienen cargados en la librería `cuentas`. Tu misión es programar las funciones mencionadas para que puedan ser utilizadas desde el siguiente código:

Ejercicio 3: P2 I1 2018-2

```
terminar = False
while not terminar:

    opcion = input('''¿Qué deseas hacer?
                    1) Obtener promedio de gastos de una categoría.
                    2) Obtener total de gastos de una categoría para un mes dado.
                    3) Obtener el gasto mensual promedio de una categoría.
                    4) Salir.''')

    if opcion == 1:
        categoria = input('Ingrese categoría: ')
        print(promedio_categoria(categoria))
    elif opcion == 2:
        categoria = input('Ingrese categoría: ')
        mes = int(input('Ingrese mes: '))
        print(total_mensual_categoria(mes, categoria))
    elif opcion == 3:
        categoria = input('Ingrese categoría: ')
        print(mensual_promedio_categoria(categoria))
    elif opcion == 4:
        terminar = True
```

Resumen de la clase

- Las funciones en Python:
  - Pueden recibir parámetros
  - Hacen un cálculo o ejecución de cierto código
  - Devuelven un valor

- ¿Cómo podemos definir una función en Python?

```
def nombre_funcion(a,b,c):
    #ejecución de cierto código
    return (valor retornado)
```

Bibliografía

- http://runest.ing.puc.cl/function.html
- A. B. Downey. Think Python: How to think like a computer scientist. Green Tea Press, 2013 -> Capítulo 3

Links

- https://repl.it/@FelipeLopez/IIC1103Funciones que contiene todos los ejemplos de la clase.

Clase 5 – Funciones

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

05-09-2019