Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

Clase 05: Funciones

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

30 de Marzo, 2015

¿Qué aprendimos las clases pasadas?

¿Qué aprendimos las clases pasadas?

Print permite imprimir código en consola.

Sintaxis

```
print(variable_1, variable_2, variable_3, ...)
```

Input retorna un str con un valor ingresado por el usuario.

Sintaxis

```
input(mensaje_para_usuario)
```

¿Qué aprendimos la clase pasada?

Condicionales: permiten ejecutar (o no) trozos de código *si* se cumple una *condición*.

Sintaxis

```
if(condicion_if):
    bloque_de_código_if
elif(condicion_elif):
    bloque_de_código_elif
else:
    bloque_de_código_else
bloque_de_código_fuera_del_if_else
```

¿Qué aprendimos la clase pasada?

Condición: variable de tipo bool.

Operadores que retornan bool

$a == b$, $a! = b$, $a < b$, $a <= b$, $a > b$ y $a >= b$.

Operadores entre booleanos

not a, a *or* b y a *and* b.

¿Qué aprendimos la clase pasada?

while: permite ejecutar varias veces la misma sección de código.

Sintaxis

```
while(condición):
    bloque_de_código_while
bloque_de_código_fuera_del_while
```

break: sale automáticamente del loop.

continue: retorna al inicio del loop.

¿Qué aprendimos la clase pasada?

Ejemplo:

```
1 i = 0
2 while(True):    # loop infinito(?)
3   i += 1
4   if(i == 5):
5     continue    # dejo de ejecutar para caso i == 5
6   if(i == 11):
7     break       # salgo del loop
8   print(i)
9 print("Fin loop")
```

Observación

¿Cuál es la diferencia entre '=' y '=='?

```
1 a = int(input("a: "))
2 b = 0
3 if(a == 3):
4   b == 1
5 else:
6   b == 2
7 print(b) # >>> 0
```

Observación

¿Cuál es la diferencia entre '=' y '=='?

```
1 a = int(input("a: "))
2 b = 0
3 if(a == 3):
4   b == 1
5 else:
6   b == 2
7 print(b) # >>> 0
```

Error! b == 1 compara 'b' con '1', no *asigna* '1' a 'b'

Observación

¿Cuál es la diferencia entre '=' y '=='?

```
1 a = int(input("a: "))
2 b = 0
3 if(a == 3):
4   b = 1
5 else:
6   b = 2
7 print(b) # >>> 1 o 2 dependiendo de a
```

Tarea: Comentarios generales

	0	1	2	3	4	5	6	7	8
0	5	3			7				
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

Programar un jugador de sudoku en consola.

Tarea: Comentarios generales

Código base en el siding:

```
1  # Importamos la librería. El archivo sudoku.py
2  # debería estar en la misma carpeta que este código
3  import sudoku
4
5  # Cargamos el tablero 1
6  # (el fácil, que aparece como ejemplo en el enunciado)
7  sudoku.cargarTablero(1)
8
9  # Mostramos lo que hay en la casilla (0,0) -> 0
10 print(sudoku.obtener(0,0))
11 # Mostramos lo que hay en la casilla (2,0) -> 8
12 print(sudoku.obtener(2,0))
```

Tarea: Comentarios generales

Primer intento:

```
1  import sudoku
2
3  sudoku.cargarTablero(1)
4  print(sudoku.obtener(0,0),...,sudoku.obtener(0,8))
5  print(sudoku.obtener(1,0),...,sudoku.obtener(1,8))
6  print(sudoku.obtener(2,0),...,sudoku.obtener(2,8))
7  print(sudoku.obtener(3,0),...,sudoku.obtener(3,8))
8  print(sudoku.obtener(4,0),...,sudoku.obtener(4,8))
9  print(sudoku.obtener(5,0),...,sudoku.obtener(5,8))
10 print(sudoku.obtener(6,0),...,sudoku.obtener(6,8))
11 print(sudoku.obtener(7,0),...,sudoku.obtener(7,8))
12 print(sudoku.obtener(8,0),...,sudoku.obtener(8,8))
```

Tarea: Comentarios generales

Primer intento:

```
1  import sudoku
2
3  sudoku.cargarTablero(1)
4  print(sudoku.obtener(0,0),...,sudoku.obtener(0,8))
5  print(sudoku.obtener(1,0),...,sudoku.obtener(1,8))
6  print(sudoku.obtener(2,0),...,sudoku.obtener(2,8))
7  print(sudoku.obtener(3,0),...,sudoku.obtener(3,8))
8  print(sudoku.obtener(4,0),...,sudoku.obtener(4,8))
9  print(sudoku.obtener(5,0),...,sudoku.obtener(5,8))
10 print(sudoku.obtener(6,0),...,sudoku.obtener(6,8))
11 print(sudoku.obtener(7,0),...,sudoku.obtener(7,8))
12 print(sudoku.obtener(8,0),...,sudoku.obtener(8,8))
```

... y si en el control les pedimos mostrar un tablero de 100×100?

Tarea: Comentarios generales

Otro elemento frecuente (para dejar espacios en blanco):

```
1 if(i==0 or i == 3 or i == 6):
2   print(" ")
```

Tarea: Comentarios generales

Otro elemento frecuente (para dejar espacios en blanco):

```
1 if(i==0 or i == 3 or i == 6):
2   print(" ")
```

Alternativa (que funciona para tablero de 100×100).

```
1 if(i%3==0):
2   print(" ")
```

Tarea: Comentarios generales

Otro elemento frecuente (para dejar espacios en blanco):

```
if(i==0 or i == 3 or i == 6):
  print(" ")
```

Alternativa (que funciona para tablero de 100×100).

```
if(i%3==0):
  print(" ")
```

Consejo: Si su código podría adaptarse a un tablero de 100×100, entonces su tarea está perfecta.

Tarea: Comentarios generales

Observación final

Solo usen la materia vista en clases!

Tarea: Comentarios generales

Observación final

Solo usen la materia vista en clases!

No usen **for**s, ni **list**as, ni códigos raros de google.

Tarea: Comentarios generales

Observación final

Solo usen la materia vista en clases!

No usen **for**s, ni **list**as, ni códigos raros de google.

Necesito que aprendan a trabajar con **while**s y pensar con **control de flujo simple**. Comandos mágicos solo los van a perjudicar (les irá mal en controles y pruebas =/).

Control Sorpresa!

```
1  a = int(input("Ingrese número: "))
2  var = False
3  b=1
4  while(b<a):
5    c=1
6    while(c<b):
7      if(a**2 == (b**2+c**2)):
8        var = True
9        break
10     c+=1
11   b+=1
12 print(var)
```

- ¿Qué imprime el programa si la entrada es 3?
- ¿Qué imprime el programa si la entrada es 5?
- ¿Qué hace este programa?

Funciones: Motivación

En muchas ocasiones necesitamos ejecutar el mismo código en varias partes distintas.

Funciones: Motivación

En muchas ocasiones necesitamos ejecutar el mismo código en varias partes distintas.

... es decir, debemos copiar y pegar.

Funciones: Motivación

"Evalué polinomio $x^4 + \frac{1}{2}x^3 + 2x^2 - x$ *para un* x *cualquiera."*

```
1 x = 4
2 res = x**4+x**3/2+2*x**2-x
3 print(res)
```

Funciones: Motivación

¿Cómo evalúo función para distintos valores de x?

```
1  x = 4
2  res = x**4+x**3/2+2*x**2-x
3  print(res)
4
5  x = 5
6  res = x**4+x**3/2+2*x**2-x
7  print(res)
8
9  x = 6
10 res = x**4+x**3/2+2*x**2-x
11 print(res)
12
13 x = 7
14 res = x**4+x**3/2+2*x**2-x
15 print(res)
```

Funciones: Motivación

Marcador game de tenis.

Dos instancias de copiar pegar:

- Mostrar marcador jugador.
- Condiciones para ganar punto.

Funciones: Motivación

Mostrar marcador.

```
11  # Muestro puntos Jugador 1
12  p1_string = "0"
13  if puntos_1 == 1: p1_string = "15"
14  elif puntos_1 == 2: p1_string = "30"
15  elif puntos_1 == 3: p1_string = "40"
16  elif puntos_1 == 4: p1_string = "v"
17  print("Jugador 1:",p1_string)
18  # Muestro puntos Jugador 2
19  p2_string = "0"
20  if puntos_2 == 1: p2_string = "15"
21  elif puntos_2 == 2: p2_string = "30"
22  elif puntos_2 == 3: p2_string = "40"
23  elif puntos_2 == 4: p2_string = "v"
24  print("Jugador 2:",p2_string)
```

Funciones: Motivación

Ganar punto.

```
5   p = int(input("¿Qué hizo el punto? (1 o 2)"))
6   if p == 1: # Agrego punto a 1
7     puntos_1 += 1
8     # Cuando supero el 40 y hay diferencia de 2
9     if(puntos_1 > 3 and (puntos_1 - puntos_2) > 1):
10      print("Game jugador 1")
11      break
12  if p == 2: # Agrego punto a 2
13    puntos_2 += 1
14    # Cuando supero el 40 y hay diferencia de 2
15    if(puntos_2 > 3 and (puntos_2 - puntos_1) > 1):
16      print("Game jugador 2")
17      break
```

Funciones: Motivación

"Haga un programa que calcule el coeficiente polinomial $C(m, n)$."

$$C(m, n) = \frac{m!}{(m-n)!n!}$$

$$n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n = \prod_{i=1}^{n} i$$

Funciones: Motivación

$$C(m,n) = \frac{m!}{(m-n)!n!}$$

```
1 # Calcular un factorial
2 n = int(input("n: "))
3 f = 1; i = 1
4 while(i < n+1):
5   f *= i
6   i+=1
7 print(n,"! =",f)
```

Funciones: Motivación

$$C(m,n) = \frac{m!}{(m-n)!n!}$$

```
1 # Calcular un factorial
2 n = int(input("n: "))
3 f = 1; i = 1
4 while(i < n+1):
5   f *= i
6   i+=1
7 print(n,"! =",f)
```

Necesitamos calcular 3 factoriales... así que debemos copiar este código 3 veces.

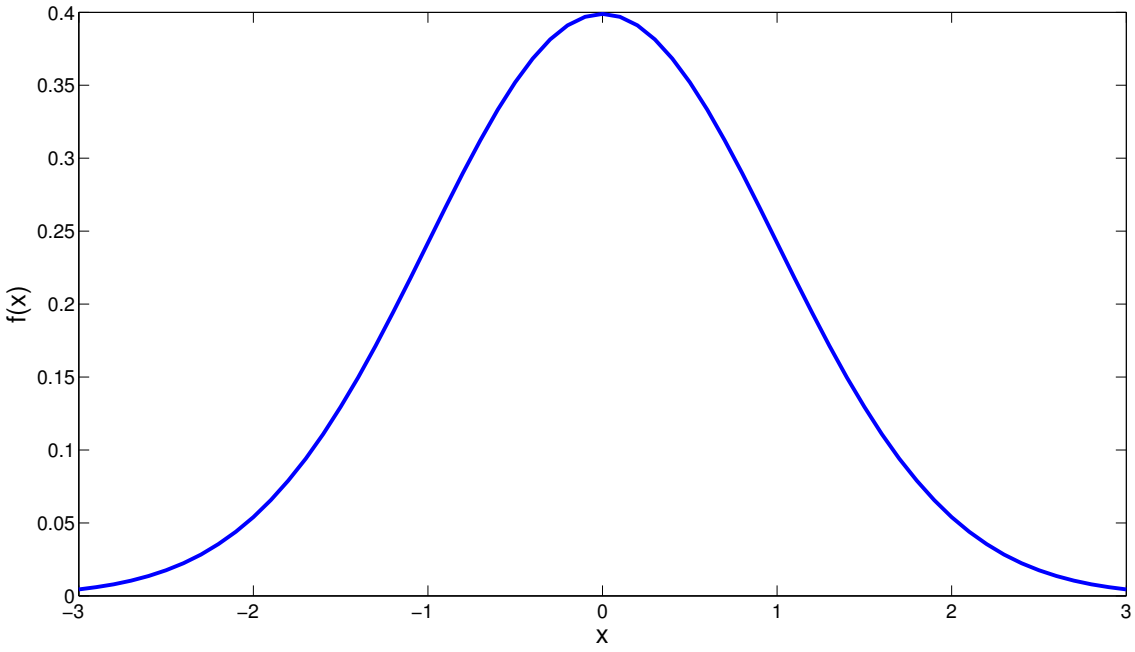
Funciones: Motivación

```
1  m = int(input("m: "))
2  n = int(input("n: "))
3
4  # Calculo m!
5  f_m = 1; i = 1
6  while(i < m+1):
7    f_m *= i; i+=1
8
9  # Calculo n!
10 f_n = 1; i = 1
11 while(i < n+1):
12   f_n *= i; i+=1
```

```
14 # Calculo (m-n)!
15 f_m_n = 1; i = 1
16 while(i < (m-n)+1):
17   f_m_n *= i; i+=1
18
19 # Obtengo resultado final
20 res = f_m/(f_m_n*f_n)
21 print("C(",m,",",n,") =",
       res)
```

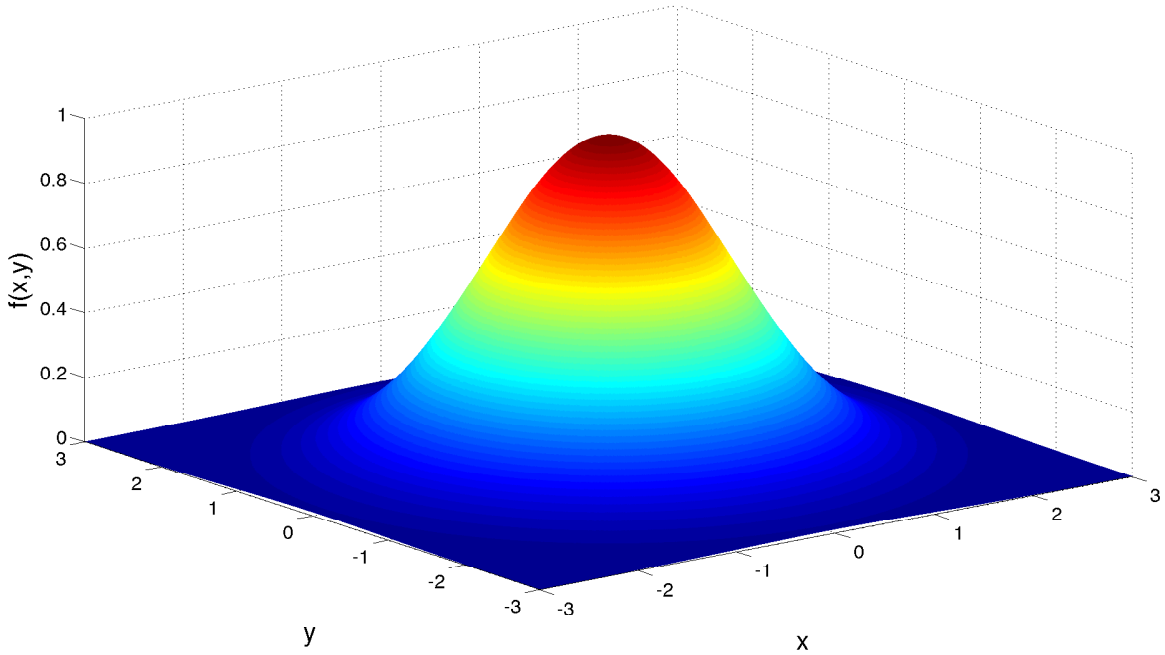
Funciones: Approach matemático

Funciones: Approach matemático



$$y = f(x) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{\frac{-x^2}{2}}$$

Funciones: Approach matemático



$$z = f(x, y) = \mathrm{e}^{\frac{-(x^2+y^2)}{2}}$$

Funciones: Approach matemático

Elementos:

- Parámetros de entrada.
- Valor de salida.
- Ecuación que lleva de la entrada a la salida.

$$z = f(x, y) = \mathrm{e}^{\frac{-(x^2+y^2)}{2}}$$

Funciones: Approach matemático

Las funciones en python tienen los mismos elementos.

Funciones: Approach matemático

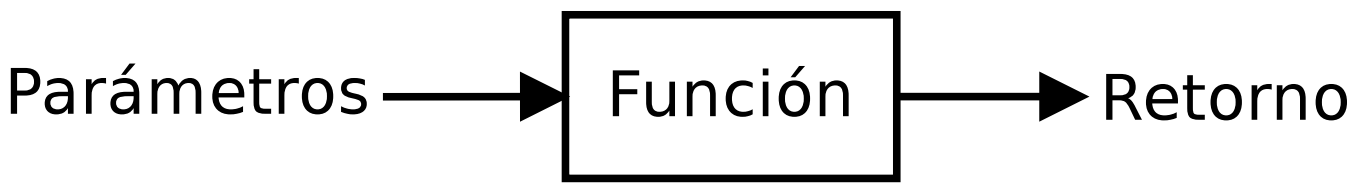
Las funciones en python tienen los mismos elementos.

Parámetros de entrada → Set de variables.
Valor salida → Retorno (algún tipo de dato).
Ecuación → Código.

Funciones: Approach matemático

Las funciones en python tienen los mismos elementos.

Parámetros de entrada → Set de variables.
Valor salida → Retorno (algún tipo de dato).
Ecuación → Código.



Funciones

Observación: Ustedes ya han usado funciones.

Funciones

Observación: Ustedes ya han usado funciones.

input(mensaje): Muestra un mensaje en consola y retorna 'str' entregado por el usuario.

print(var_1,var_2, ...): Imprimie mensaje en consola.
print(var_1,var_2, ..., end=" "): Imprimie mensaje sin salto de línea.

sudoku.cargarTablero(n): Carga un tablero de dificultad n.
sudoku.obtener(i,j):Retorna el valor de la casilla (i,j).

Funciones

Observación: Ustedes ya han usado funciones.

input(mensaje): Muestra un mensaje en consola y retorna 'str' entregado por el usuario.

print(var_1,var_2, ...): Imprimie mensaje en consola.
print(var_1,var_2, ..., end=" "): Imprimie mensaje sin salto de línea.

sudoku.cargarTablero(n): Carga un tablero de dificultad n.
sudoku.obtener(i,j):Retorna el valor de la casilla (i,j).

... Lo que no han hecho, es definir sus propias funciones.

Funciones

Funciones: Permiten definir un nombre para un trozo de código. Pueden recibir parámetros y retorna un valor.

Funciones

Funciones: Permiten definir un nombre para un trozo de código. Pueden recibir parámetros y retorna un valor.

Sintaxis: Definir función

```
def nombre_función(param_1, param_2, ...):
    inst_1
    ...
    inst_n
    return ret
```

Sintaxis: Llamar a una función

```
out = nombre_función(in_1, in_2, ...)
```

Funciones

Pasos para utilizar funciones:

- Definir la función.
- Llamar la función desde tu código.

Funciones

1. **Definir la función**: Aquí definimos el comportamiento de la función (su código).

Funciones

1. Definir la función: Aquí definimos el comportamiento de la función (su código).

```
"""
Esta función retorna la suma de los dígitos
del número 'n'
"""
def sumar_digitos(n): # <- Función recibe 1 parámetro
  # Código de la función
  suma = 0
  while(n!=0):
    suma+=n%10
    n//=10
  # Retornamos la suma de los dígitos
  return suma
```

Obs: La función se define una sola vez en tu programa.

Funciones

Parámetros:

- Lista de variables que recibe la función para trabajar.
- Podría no tener parámetros.

Funciones

Parámetros:

- Lista de variables que recibe la función para trabajar.
- Podría no tener parámetros.

Código:

- El código de una función no tiene ninguna restricción (puede ser un programa completo).

Funciones

Parámetros:

- Lista de variables que recibe la función para trabajar.
- Podría no tener parámetros.

Código:

- El código de una función no tiene ninguna restricción (puede ser un programa completo).

Retorno:

- Sirve para entregar un resultado a quien llame a la función.
- Una función podría no retornar nada.
- Al retornar la función se acaba.

Funciones

Ejemplo:

```
"""
Esta función retorna true si n es primo
"""
def es_primo(n):
  # Si n==1 retorno False de inmediato
  if(n == 1):
    return False
  i = 2
  while(i<n):
    # Si encuentro un divisor exacto retorno False
    if(n%i==0):
      return False
    i+=1
  # Si llego acá es porque el número era primo
  return True
```

Funciones

2. Llamar función:

- Desde tu código puedes *llamar* a funciones ya definidas.
- Al llamarla debes dar valor a **todos** sus parámetros.
- La función devolverá su valor de retorno.

Funciones

2. Llamar función:

- Desde tu código puedes *llamar* a funciones ya definidas.
- Al llamarla debes dar valor a **todos** sus parámetros.
- La función devolverá su valor de retorno.

```
15 a = int(input("Ingrese un número: "))
16 b = int(input("Ingrese otro número: "))
17 s_a = sumar_digitos(a)
18 s_b = sumar_digitos(b)
19 print("La multiplicación es:",s_a*s_b)
```

Funciones

Dato freak: Se pueden definir valores por *defecto* para parámetros de una función mediante un =.

```
1 # Función con 2 parámetros con valores por defecto
2 def raiz(num,exp=0.5,delta=0):
3   return num**exp+delta
4
5 print(raiz(36,0.33,1))
6 print(raiz(36))
7 print(raiz(36,0.33))
8 print(raiz(36,delta=4))
```

Funciones: Ejemplos

"Evalué polinomio $x^4 + \frac{1}{2}x^3 + 2x^2 - x$ *para un* x *cualquiera."*

Funciones: Ejemplos

Antes:

```
1  x = 4
2  res = x**4+x**3/2+2*x**2-x
3  print(res)
4
5  x = 5
6  res = x**4+x**3/2+2*x**2-x
7  print(res)
8
9  x = 6
10 res = x**4+x**3/2+2*x**2-x
11 print(res)
12
13 x = 7
14 res = x**4+x**3/2+2*x**2-x
15 print(res)
```

Funciones: Ejemplos

Después:

```
1 def f(x):
2   res = x**4+x**3/2+2*x**2-x
3   return res
4
5 print(f(4))    # >>> 4 -> 316.0
6 print(f(5))    # >>> 5 -> 732.5
7 print(f(6))    # >>> 6 -> 1470.0
8 print(f(7))    # >>> 7 -> 2663.5
```

(Explicar cómo sería la ejecución de este programa)

Funciones

Importante: Toda función debe ser *definida* antes de ser *llamada*.

```
1 f(4)    # >>> NameError: name 'f' is not defined
2 f(5)
3 f(6)
4 f(7)
5
6 def f(x):
7   res = x**4+x**3/2+2*x**2-x
8   print(x,"->",res)
```

Funciones: Ejemplos

Marcador game de tenis.

Dos instancias de copiar pegar:

- Mostrar marcador jugador.
- Condiciones para ganar punto.

Funciones: Ejemplos

Mostrar marcador.

```
11  # Muestro puntos Jugador 1
12  p1_string = "0"
13  if puntos_1 == 1: p1_string = "15"
14  elif puntos_1 == 2: p1_string = "30"
15  elif puntos_1 == 3: p1_string = "40"
16  elif puntos_1 == 4: p1_string = "v"
17  print("Jugador 1:",p1_string)
18  # Muestro puntos Jugador 2
19  p2_string = "0"
20  if puntos_2 == 1: p2_string = "15"
21  elif puntos_2 == 2: p2_string = "30"
22  elif puntos_2 == 3: p2_string = "40"
23  elif puntos_2 == 4: p2_string = "v"
24  print("Jugador 2:",p2_string)
```

Funciones: Ejemplos

Definición:

```
1  """
2  Función que mapea del contador al puntaje
3  p: contador del jugador (entre 0 y 4)
4  retorna "0", "15", "30", "40" o "v"
5  """
6  def obtener_puntaje(p):
7    ret = "0"
8    if p == 1: ret = "15"
9    elif p == 2: ret = "30"
10   elif p == 3: ret = "40"
11   elif p == 4: ret = "v"
12   return ret
```

Funciones: Ejemplos

Uso de la función:

```
42  # Muestro puntos Jugador 1
43  print("Jugador 1:",obtener_puntaje(puntos_1))
44  # Muestro puntos Jugador 2
45  print("Jugador 2:",obtener_puntaje(puntos_2))
```

Funciones: Ejemplos

Ganar punto.

```
5   p = int(input("¿Qué hizo el punto? (1 o 2)"))
6   if p == 1: # Agrego punto a 1
7     puntos_1 += 1
8     # Cuando supero el 40 y hay diferencia de 2
9     if(puntos_1 > 3 and (puntos_1 - puntos_2) > 1):
10      print("Game jugador 1")
11      break
12  if p == 2: # Agrego punto a 2
13    puntos_2 += 1
14    # Cuando supero el 40 y hay diferencia de 2
15    if(puntos_2 > 3 and (puntos_2 - puntos_1) > 1):
16      print("Game jugador 2")
17      break
```

Funciones: Ejemplos

Definición:

```
14 """
15 Retorna True si el jugador ganó el game
16 jugador: Número del jugador que anotó el punto (1 o 2)
17 p_actual: Puntaje jugador (entre 0 y 5)
18 p_oponente: Puntaje contrincante (entre 0 y 5)
19 """
20 def fin_game(jugador,p_actual,p_oponente):
21   if(p_actual > 3 and (p_actual - p_oponente) > 1):
22     print("Game jugador",jugador)
23     return True
24   return False
```

Funciones: Ejemplos

Uso de la función:

```
28  p = int(input("¿Qué hizo el punto? (1 o 2)"))
29  if p == 1: # Agrego punto a 1
30    puntos_1 += 1
31    if(fin_game(1,puntos_1,puntos_2)):
32      break
33  if p == 2: # Agrego punto a 2
34    puntos_2 += 1
35    if(fin_game(2,puntos_2,puntos_1)):
36      break
```

Funciones: Ejemplos

"Haga un programa que calcule el coeficiente polinomial $C(m, n)$."

$$C(m, n) = \frac{m!}{(m-n)!n!}$$

$$n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n = \prod_{i=1}^{n} i$$

Funciones: Ejemplos

Antes:

```
1  m = int(input("m: "))
2  n = int(input("n: "))
3
4  # Calculo m!
5  f_m = 1; i = 1
6  while(i < m+1):
7    f_m *= i; i+=1
8
9  # Calculo n!
10 f_n = 1; i = 1
11 while(i < n+1):
12   f_n *= i; i+=1
```

```
14 # Calculo (m-n)!
15 f_m_n = 1; i = 1
16 while(i < (m-n)+1):
17   f_m_n *= i; i+=1
18
19 # Obtengo resultado final
20 res = f_m/(f_m_n*f_n)
21 print("C(",m,",",n,") =",
       res)
```

Funciones: Ejemplos

Después:

```
1  # Retorna el factorial de "num"
2  def factorial(num):
3    f = 1; i = 1
4    while(i < num+1):
5      f *= i; i+=1
6    return f
7
8  # Código principal
9  m = int(input("m: "))
10 n = int(input("n: "))
11 # Obtengo resultado final
12 res = factorial(m)/(factorial(m-n)*factorial(n))
13 print("C(",m,",",n,") =",res)
```

Funciones: Ejemplos

Después:

```
1  # Retorna el factorial de "num"
2  def factorial(num):
3    f = 1; i = 1
4    while(i < num+1):
5      f *= i; i+=1
6    return f
7
8  # Código principal
9  m = int(input("m: "))
10 n = int(input("n: "))
11 # Obtengo resultado final
12 res = factorial(m)/(factorial(m-n)*factorial(n))
13 print("C(",m,",",n,") =",res)
```

Importante: Lo que pasa en una función, se queda en una función... a menos que se *retorne* algo.

Funciones: Ejemplos

Después después:

```
1  # Retorna el factorial de "num"
2  def factorial(num):
3    f = 1; i = 1
4    while(i < num+1):
5      f *= i; i+=1
6    return f
7
8  # Retorna C(m,n)
9  def binomial(m,n):
10   return factorial(m)/(factorial(m-n)*factorial(n))
11
12 # Código principal
13 m = int(input("m: "))
14 n = int(input("n: "))
15 # Obtengo resultado final
16 print("C(",m,",",n,") =",binomial(m,n))
```

Funciones

Ventajas:

1. Permiten reutilizar código.
2. Código más legible[1].
3. Código más mantenible.
4. Código más corto.

[1] Es importante dar buenos nombres a las funciones.

Funciones

Lección de vida

Eviten copiar y pegar código!

... por lo general se puede encapsular ese código dentro de una función.

Ejercicios

1) Cree una función que retorne el máximo entre dos números.

2) Cree un método que reciba los datos de un usuario y los muestre en consola. Los datos son: nombre, apellido, edad, correo y dirección. Para cada parámetro de un valor por defecto. En el código principal llame al método y pruebe distintas combinaciones de datos conocidos del usuario.

Ejercicios

4) Cree un método que reciba un rut y retorne su dígito verificador. **Algoritmo**: Multiplicar cada dígito del RUT por 2, 3, ..., 7, 2, 3, ... de atrás hacia adelante. Sumar las multiplicaciones parciales. Calcular el resto de la división por 11. El Dígito Verificador es 11 menos el resultado anterior. Si es 10, se cambia por 'k'; si es 11, por 0.

Ejercicios

[**Ex_rec 2014-2**] Considere el siguiente código:

```
1 # Condición: "d" es un entero mayor que cero
2 def misterio1(d):
3   a = 0
4   while(d > 0):
5     if(d%2 == 0):
6       a += 1
7     d //= 10
8   return a
```

i) ¿Cuál es el retorno de `misterio1(123)`?

ii) ¿Cuál es la relación entre `d` y el retorno de `misterio1(d)`? (explique la semántica de la función, no su código)

iii) De un ejemplo para el cual `misterio1(d)` retorne `6`.

Ejercicios

[**Ex_rec 2014-2**] Considere el siguiente código:

```
10 # Condición: "a" y "b" son enteros mayores que cero
11 def misterio2(a,b):
12   a1 = a; b1 = b
13   while(a1 != b1):
14     if(a1 < b1):
15       a1 += a
16     else:
17       b1 += b
18   return a1
```

i) ¿Cuál es el retorno de `misterio2(12,24)`?

ii) ¿Cuál es la relación entre `a`, `b` y el retorno de `misterio2(a,b)`? (explique la semántica de la función, no su código)

iii) De un ejemplo para el cual `misterio2(a,b)` retorne `6` tal que `a` y `b` sean distintos de `6`.