Python
Bool/Funciones

Clase #4

IIC1103 – Introducción a la Programación

IDEs Para Python

- Jupyter: https://jupyter.org/
- Spyder: https://github.com/spyder-ide/spyder
- Pycharm Community: https://www.jetbrains.com/pycharm/
- IDLE, Repl.it, Clearn,…

Otros: https://www.programiz.com/python-programming/ide

El plan de hoy es...

Primero: resolver el misterio

round(): cambió su comportamiento en Python 3.0: “exact halfway cases are now rounded to the nearest even result instead of away from zero”

Este comportamiento se llama “round half to even” o “banker’s rounding”

Y siguiendo con las aclaraciones...

```
>> 0.1 + 0.2
```

- Leer: https://www.pylenin.com/blogs/python-float-arithmetics/

Ahora si: el plan de hoy es...

- Recordar la clase pasada
- Un nuevo tipo de variable
- Introducción a funciones *built-in*
- Operadores de comparación y booleanos

Repaso en Menti

Recordatorio: ¿qué vimos la clase pasada?

- Instrucciones que conocemos hasta el momento
  - `print("algun texto")`
    - → `print(“hola mundo!”)` escribe “hola mundo”
  - `input("algun texto")`
    - → `input("ingrese valor")` escribe “ingrese valor”, espera leer un texto
  - `variable = valor`
    - → `a = 4` crea una variable llamada a donde guarda el valor 4

Recordatorio: ¿qué vimos la clase pasada?

- Instrucciones que conocemos hasta el momento
  - `int("texto con nº entero")`
    - → `int("345")` convierte un texto en un número entero
  - `float("texto con nº real")`
    - → `float("45.3")` convierte un texto en un número real
  - `str(numero)`
    - → `str(345)` convierte un número a un texto

Operaciones matemáticas

op	Descripción
+	suma
-	resta
*	multiplicación
/	división
//	división entera
%	resto
**	potencia

Prioridades de operadores

1	paréntesis
2	potencias
3	multiplicación, división
4	suma, resta

Problema #1

- Cuántos chocolates hay? 25
- Cuántas personas hay? 4
- Repartir 6 chocolates para cada uno, sobra 1

Solución

- `choc = int(input("Cuántos chocolates hay?"))`
- `pers = int(input("Cuántas personas hay?"))`
- `print("Repartir",choc//pers,"chocolates para cada uno, sobra",choc%pers)`

Problema #2

- Voy a tirar un dado.
- Adivina cuál número me sale? 6
- Salió un 5.
- Adivinaste = False

Solución: Algoritmo

1. Escribir `Voy a tirar un dado.`

2. Tirar un dado

3. Escribir `Adivina cuál número me sale?` y leer nº ingresado por usuario

4. Escribir cual nº salió

5. Escribir `Adivinaste = True o False`

Solución: Seudocódigo

Éste aún no es código Python, tiene algunas instrucciones en castellano

- `print("Voy a tirar un dado")`
- `dado = tirar un dado (nº entero)`
- `n = int(input("Adivina cual numero me sale?"))`
- `print("Salio un "+str(dado))`
- `print("Adivinaste = "+ True / False)`

Funciones

- Reciben argumentos/parámetros
- Hacen cálculos
- Retornan/devuelven algún valor

Built-in functions

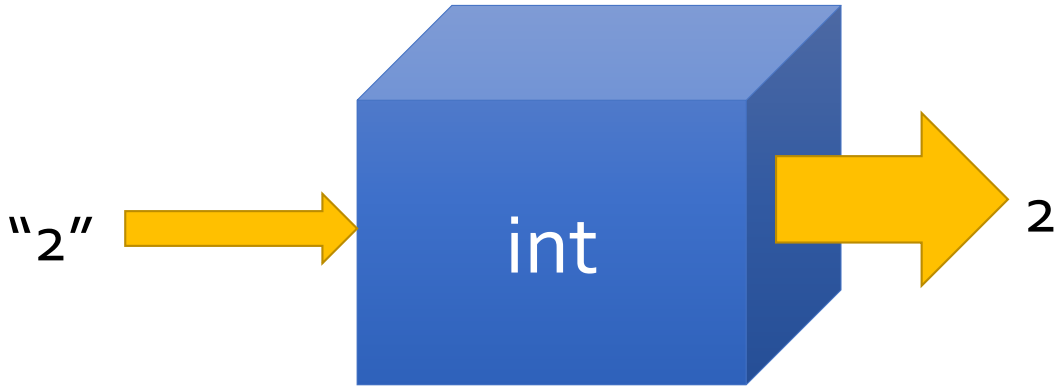
Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

¿Investigar más?

- Todas las funciones que vienen en Python (sin necesidad de importar nada) se describen aquí:
- https://docs.python.org/3/library/functions.html

Funciones

- Reciben argumentos/parámetros
- Hacen cálculos
- Retornan/devuelven algún valor



random - http://docs.python.org/3/library/random.html

Table Of Contents

9.6. random — Generate pseudo-random numbers
- 9.6.1. Notes on Reproducibility
- 9.6.2. Examples and Recipes

Previous topic
9.5. fractions — Rational numbers

Next topic
9.7. statistics — Mathematical statistics functions

This Page
Report a Bug
Show Source

Quick search
Go
Enter search terms or a module,

9.6. random — Generate pseudo-random numbers

Source code: Lib/random.py

This module implements pseudo-random number generators for various distributions.

For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, for random sampling without

`random.randint(a, b)`
Return a random integer *N* such that `a <= N <= b`.

On the r egative exponential, gamma, bution is available.

Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range [0.0, 1.0). Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of 2**19937-1. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The functions supplied by this module are actually bound methods of a hidden instance of the `random.Random` class. You can instantiate your own instances of `Random` to get generators that don't share state.

¿Investigar más?

- Cómo los computadores generan números al azar (y la diferencia entre números aleatorios y seudo-aleatorios): https://www.howtogeek.com/183051/htg-explains-how-computers-generate-random-numbers/

- Leer: https://docs.python.org/3/library/random.html (¿Qué semilla usa para generar números seudo-aleatorios?, ¿qué entrega random.random()?)
  - Generador de números random en Python es Mersenne Twister

Solución

```
import random
print("Voy a tirar un dado")
n = int(input("Adivina cual numero me sale?"))
dado = random.randint(1,6)
print("Salio un "+str(dado))
print("Adivinaste = "+str(dado==n))
```

Un nuevo tipo de variable…

- **`bool`**: valor de verdad, solo dos valores posibles: `True` y `False`
- `a = 5`
- `b = 3`
- `a>b #True`
- `b>a #False`
- `c = a>b # c es True`

- `d = "hola"`
- `d == "chao" #False`
- `d != "chao" #True`

operadores booleanos	
==	son iguales?
!=	son distintos?
<	menor
>	mayor
<=	menor o igual
>=	mayor o igual

Tipos de variable…

- Hasta el momento:
  - `int`
  - `float`
  - `bool`
  - `str`

- **`type`**`(3)`
- `type(True)`

¿Cómo se diferencian? (además de la funcionalidad)

Algunas funciones retornan un valor, otras no devuelven nada, simplemente ejecutan alguna acción (como imprimir)

- `print("hola")`
- `x = random.randint(1,6)`

Resumen de hoy

- **Clase pasada:**
  - Variables y asignación `x =3`
  - Input/output (`input, print, int, float, str`)
- **Hoy vimos**:
- Operadores matemáticos: `+, -, *, /, //, %, **`
- `import random` ➔ `random.randint(a,b)`
- tipo `bool`: `True` o `False`
- **Próxima clase**: if / else / elif

Bibliografía Adicional

- Tipos básicos: http://runest.ing.puc.cl/basics.html
- Cómo se generan los números aleatorios: https://www.howtogeek.com/183051/htg-explains-how-computers-generate-random-numbers/
- Referencia de random: https://docs.python.org/3/library/random.html