

PYTHON PER A LINGÜISTES

CONCEPTES BÀSICS DE PYTHON:

ESTRUCTURES DE DECISIÓ, BUCLES, FUNCIONS

Alex Peiró Lilja

Servei de Tecnologia Lingüística (STeL)

Universitat de Barcelona, 2023

RESUM: COMPUTACIÓ?

- Computació = Dades + Algorismes
- Dades: “Ingredients del pastís” (INPUTS), “El pastís llest per servir” (OUTPUT)
- Algorisme: un procés basat en instruccions específiques, “Preparar el pastís”
- Algorisme té, normalment, dades d'entrada, i/o de sortida
- Algorisme pot tenir diverses passos intermedis
- Per tal de supervisar els canvis, fem servir les variables

RESUM: TIPUS DE DADES?

- Diferents tipus, cadascun amb les seves respectives operacions
- Tipus numètic: enters (1, 4, 19, 26); flotants (1.5, 0.5, 6.667)
- Tipus boolean: *True*, *False*
- Tipus string: "u", "un", "un string", "un string &/\$.%="
- Estructura de dades tipus llista: [1, "hola", 5.5, 23, "món", False]

QUÈ SABEM SOBRE TREBALLAR AMB LES DADES?

- Hem après les operacions bàsiques de diferents tipus de dades
- El tipus de dades amb les que operarem delimita les operacions que podem realitzar:
 - Tipus numèrics es poden modificar utilitzant operacions aritmètiques
 - Els strings i les llistes poden ser concatenats o segmentats

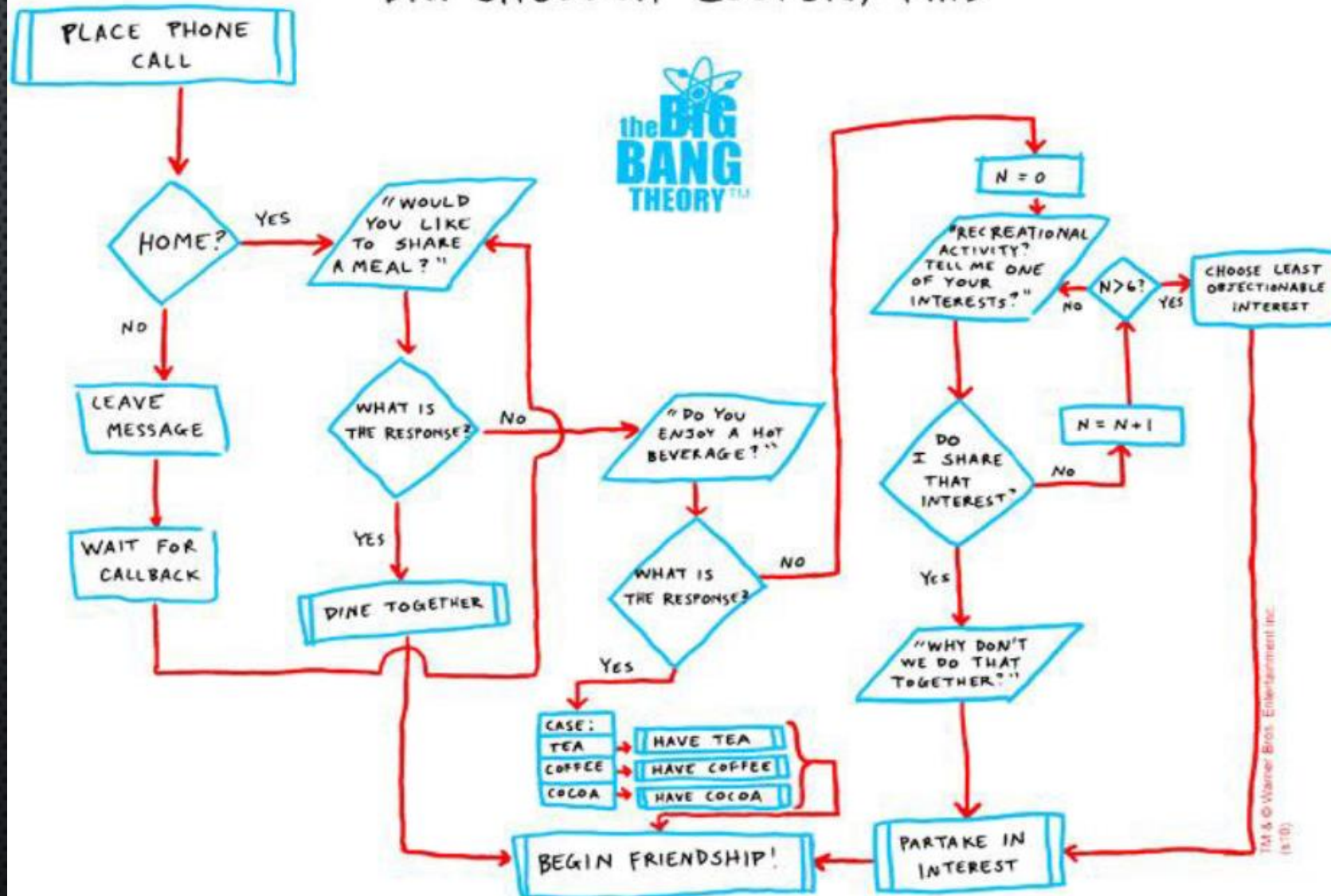
D'OPERACIONS SIMPLS A PROGRAMES REALS

- Conèixer els tipus de dades i les seves operacions és essencial per a la programació
- Però hi ha més elements que defineixen un programa que no pas les operacions bàsiques
- En aquesta classe aprendrem sobre: estructures de control, bucles i les funcions

ESTRUCTURES DE CONTROL CONDICIONS

THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER, Ph.D



ESTRUCTURES DE DECISIÓ

- En un programa es poden avaluar una o varies condicions per determinar els següents passos a seguir:
 - “Hi ha menjar a la nevera?” – “sí” – “cuinem” / “no” – “anar a comprar”
 - “El nostre mòbil té bateria?” – “sí” – “Instagram” / “no” – “carregar-lo”
- El programa, al trobar-se amb una d'aquestes preguntes, haurà de prendre una decisió. Normalment s'anomenen **estructures de decisió**

ESTRUCTURES DE DECISIÓ: CONDICIÓ, VALOR, ACCIÓ

- Cada estructura de decisió té una **condició**:
 - “S’ha esgotat la bateria del mòbil?”
- Quan avaluem la condició obtenim un **valor** boolean (True/False)
 - “Sí, s’ha esgotat” (TRUE); “No, encara té suficient bateria” (FALSE)
- Depenent del valor de la condició, l’estructura de decisió pren una decisió i executa l’**acció** corresponent:
 - IF True THEN “Posar el mòbil a carregar”

ESTRUCTURA DE DECISIÓN EN PYTHON

- La sintaxi usada en Python per a les estructures de decisió és:

if CONDICIÓN:

 ACCIÓN

else:

 UNA ALTRA ACCIÓN

ESTRUCTURA DE DECISIÓ EN PYTHON (2)

- Per iniciar una estructura de decisió en un programa introduïm “**if**”
- A continuació de l'expressió “if” s'especifica la condició a avaluar
- Aquesta condició ha de ser avaluada abans de prendre qualsevol acció
- El resultat de comprovar aquesta condició ha de ser **True** o **False**
- Després de la condició afegirem dos punts ‘:’ per fer-li saber al programa que ha finalitzat la condició a avaluar

ESTRUCTURA DE DECISIÓ EN PYTHON (3)

- Quan la condició seguida de l'expressió "if" sigui *True*, Python executarà les següents línies de codi (accions) que estiguin indentades. La indentació pot ser una tabulació en la majoria d'editors de codi. Aquest canvi de sagnia li permet a Python saber en quin context es troba.
- El codi de l'ACCIÓ només s'executarà si la condició és TRUE
- L'expressió "else" és opcional per si volem afegir una acció per quan la condició és FALSE
- Si no especifiquem cap "else" i la condició no es compleix, el programa no executarà cap de les accions i continuarà el codi, saltant aquesta part

ESTRUCTURA DE DECISIÓ NESTING

- Les estructures de decisió en Python poden ser niades (*nesting*). És a dir, podem afegir una o més estructures de decisió dins d'una altra ja existent:

```
if edat < 25:  
    if genere = "dona":  
        print("dona jove")  
    else:  
        print("home jove")
```

OPERACIONS AMB DADES NUMÈRIQUES

- Fins ara sabem que el “resultat” d’avaluar una condició ha de ser True o False
- Però, com es programen aquestes condicions?
 - Booleana: if True: (o) if False: (no tenen sentit, però funcionen)
 - Igual a (==): if 5 == 4: (o) if age == 25:
 - Diferent a (!=): if 5 != 4: (o) if age != 25:
 - Major que (>): if 5 > 4: (o) if age > 25:
 - Major o igual que (>=): if 5 >= 4: (o) if age >= 25:
 - Menor que (<): if 5 < 4: (o) if age < 25:
 - Menor o igual que (<=): if 5 <= 4: (o) if age <= 25:

QUÈ PODEM TROBAR DINS D'UNA CONDICIÓN?

- Comparacions numèriques com les anteriors
- Comparacions d'*strings* o llistes
- Funcions o mètodes que retornin un valor boolean (més endavant veurem què són les funcions)

BUCLES:
EL BUCLE 'WHILE'

BUCLES

- A vegades, en un programa és necessari executar la mateixa acció diverses vegades seguides
- Per exemple, quan mengem un entrepà, anem mossegada a mossegada fins que [CONDICIÓ] ja s'ha acabat O no tenim més gana
- Podríem escriure de dues maneres aquest algorisme:
 - Repetir per escrit X vegades les línies de codi: fer una mossegada, comprovar... Fins que es compleixi una de les condicions mencionades
 - Utilitzar una estructura de control "loop" (bucle)
- Diverses maneres de realitzar "loop", però començarem amb WHILE

EL BUCLE WHILE

- El bucle WHILE és similar a l'estructura de decisió, ja que també avalua condicions. La sintaxi del bucle WHILE en Python és la següent:

sentencia_1

while CONDICIÓN:

sentencia_2

sentencia_3

(En general, la sentència 2 farà que el valor de la condició canviï passades X repeticions)

- De nou, fem servir els dos punts i la indentació per indicar el final de la condició i el canvi de context

EL BUCLE WHILE (2)

- El funcionament del bucle WHILE és molt similar al de l'estructura de decisió
- La diferència principal és que l'estructura de decisió executa les accions una única vegada, mentre que el bucle les executarà un cop rere altre mentre la condició segueixi sent TRUE
- Normalment, les accions dins el bucle modificaran part de la condició del bucle
 - Per exemple: l'exemple de l'entrepà, la instrucció serà “fer una mossegada” i la condició serà si encara queda entrepà. Si encara queda entrepà (TRUE) es tornarà a realitzar l'acció i es comprovarà de nou

FUNCIONS I MÈTODES

FUNCIONS

- Les variables ens permeten donar un nom a les dades que manipulem per poder-les reutilitzar
- Les funcions ens permeten posar nom a un algorisme per poder-lo cridar i reutilitzar
- Una funció és un bloc de codi, escrit dins d'un programa, el qual podem cridar/executar escrivint el nom/etiqueta que li hem assignat
- Fins ara coneixem funcions natives de Python: `print()` i `len()`

FUNCIONS: US I PARÀMETRES

- Per executar una funció, hem d'escriure el nom de la funció seguit dels arguments d'entrada dins els parèntesi:
 - `print("Hello World!")`
- La funció `print()` mostra per pantalla el text que li passem com entrada
- "Hello World" rep el nom de paràmetre o argument d'entrada. Li estem dient a la funció QUÈ és el que volem que mostri. Canviar el paràmetre d'entrada canviarà el resultat, però el codi executat internament serà sempre el mateix

FUNCIONS NATIVES I PERSONALITZADES

- Hi ha dos tipus de funcions: les funcions natives són aquelles que venen incloses de base en el llenguatge Python i estan disponibles en qualsevol moment sense cap instal·lació extra
 - `print()` o `len()`
- Podem també crear les nostres pròpies funcions – funcions específiques que ens serveixen per crear un programa. Nosaltres triem el nom, els arguments d'entrada, les instruccions i la sortida que ha retornar

FUNCIONS. RETURN VALUES

- Les funcions poden, opcionalment, retornar un valor o diversos (OUTPUT)
- Per exemple, si executem `len("Hello World")`, li demanem a la funció `len()` que calculi la longitud de l'*string* "Hello World", que en aquest cas serà el valor enter 11. Aquest valor de retorn de la funció la podem passar directament a una variable (`str_len = len("Hello World")`)
- En el cas de la funció `print()` senzillament mostra el que li passem com argument d'entrada per la pantalla.

FUNCIONS. PARÀMETRES D'ENTRADA I SORTIDA

- Podem pensar en la funció com un petit algorisme:
 - Entrada → algorisme → sortida
 - Arguments d'entrada → codi de la funció → valors de sortida
- Les funcions poden requerir de 0 arguments d'entrada: *printRandomPhrase()*
- També poden retornar 0 o més valors: *first, last = getFirstAndLastLetter(word)*

MÈTODES

- Els mètodes són funcions específiques d'un determinat tipus de dada "objecte". Un objecte és una instància d'una classe.
- Parlar de classes, objectes i programació orientada a objectes no està inclòs en aquest curs. Per tant, considerarem els mètodes com funcions que pertanyen a certs tipus de dades.
- Exemple de mètode específic d'una instància de classe *string*:
 - `text = "Això és un TEXT"`
 - `text.lower()` ← retorna "això és un text"
 - `text.endswith("T")` ← retorna True

MÈTODES (2)


- Els mètodes “pertanyen” a una certa variable. Fan “quelcom” amb el contingut de la variable. Per exemple, `.lower()` transforma tots els caràcters en minúscules.
- Accedim als mètodes d'una variable utilitzant l'operador `'.'` just després del nom de la variable
- De manera similar a les funcions, els mètodes poden tenir 0 o més arguments d'entrada
- De manera similar a les funcions, els mètodes poden retornar 0 o més valors

FUNCIONS I MÈTODES: MÒDULS I IMPORTACIONS

- Les funcions natives de Python poden ser utilitzades directament en el nostre codi. Els tipus de dades natives (int, string, bool, float) tenen els seus mètodes accessibles també. Qualsevol altra funció que no sigui nativa de Python ha de ser importada abans de ser utilitzada en el nostre codi. Exemple: paquet **math**

```
import math
```

```
math.sqrt(36) ← retorna 6
```



Per accedir a una funció dins d'un objecte o mòdul/libreria, fem servir l'operador '.' seguit del nom de la funció

DEFINIR LA NOSTRA PRÒPIA FUNCIO

- Observem el codi de la següent funció. Què rep, què fa i què retorna?

```
def personal_info(edat, genere):  
    if edat < 25:  
        if genere == "femení":  
            print("Dona jove")  
        else:  
            print("Home jove")  
    else:  
        print("Persona sàvia!")
```

DEFINIR LA NOSTRA PRÒPIA FUNCIO

- **def** indica el començament de la definició d'una funció
 - El nom de la funció ha d'aparèixer just després
 - Tots els paràmetres d'entrada de la funció han de definir-se entre els parèntesi.
Cada paràmetre és una variable
 - El doble punt marca el final de la definició de la funció i els seus arguments d'entrada
 - Les instruccions dins la funció s'escriuen com sempre. Vigilar la indentació!
- **return** indica el valor o valors a retornar, si existeixen (per defecte és None)

RESUM

- En aquestes dues primeres sessions hem cobert els conceptes bàsics de la programació
- És important entendre bé aquests conceptes, ja que els utilitzarem durant les properes sessions
- Qualsevol dubte, pregunteu i investigueu
- La sessió pràctica hauria d'ajudar per familiaritzar-vos amb el que hem vist fins ara

GRÀCIES!
PREGUNTES?