20/07/23

LE LAMBDA IN PYTHON

master.D

Campari Mirko

master.D

LE FUNZIONI LAMBDA, COMUNEMENTE CHIAMATE "LAMBDA", SONO UN CONCETTO CHIAVE DELLA PROGRAMMAZIONE FUNZIONALE.

IN GENERALE, UNA FUNZIONE LAMBDA È UNA FUNZIONE ANONIMA CHE PUÒ ESSERE DEFINITA SENZA UN NOME FORMALE. AL CONTRARIO DELLE FUNZIONI TRADIZIONALI DEFINITE CON LA PAROLA CHIAVE "DEF", LE FUNZIONI LAMBDA SONO DEFINITE UTILIZZANDO L'ESPRESSIONE "LAMBDA" SEGUITA DA UNA LISTA DI ARGOMENTI, DUE PUNTI E UN'ESPRESSIONE.

master.D

QUESTO TIPO DI FUNZIONE È SPESSO UTILIZZATO QUANDO SI DESIDERA CREARE UNA FUNZIONE SEMPLICE, CHE PUÒ ESSERE DEFINITA E UTILIZZATA NELLO STESSO PUNTO DEL CODICE, SENZA LA NECESSITÀ DI ASSEGNARLE UN NOME.

master.D

IN PYTHON, LE FUNZIONI LAMBDA SONO SUPPORTATE E OFFRONO UN MODO CONCISO PER DEFINIRE FUNZIONI BREVI E IMMEDIATE.

NELLO SPECIFICO, IN PYTHON, LE FUNZIONI LAMBDA VENGONO CREATE UTILIZZANDO LA STESSA SINTASSI GENERALE:

"LAMBDA ARGOMENTI: ESPRESSIONE".

master.D

QUESTE FUNZIONI POSSONO PRENDERE UN NUMERO QUALSIASI DI ARGOMENTI, MA POSSONO CONTENERE SOLO UN'ESPRESSIONE.

IL VALORE DI RITORNO DELLA FUNZIONE È DATO PROPRIO DALL'ULTIMA ESPRESSIONE VALUTATA. LE FUNZIONI LAMBDA SONO SPESSO UTILIZZATE INSIEME A FUNZIONI DI ORDINE SUPERIORE COME "MAP", "FILTER" E "REDUCE", RENDENDO IL CODICE PIÙ CONCISO E LEGGIBILE.

master.D

ORIGINE DELLE FUNZIONI LAMBDA:

IL CONCETTO DI FUNZIONI LAMBDA DERIVA DAL CALCOLO LAMBDA, UNA FORMA DI NOTAZIONE MATEMATICA INTRODOTTA DAL MATEMATICO ALONZO CHURCH NEGLI ANNI '30.

IL CALCOLO LAMBDA È UNA TEORIA MATEMATICA DI FUNZIONI COMPUTABILI E OFFRE UN MODELLO DI CALCOLO BASATO SULL'UTILIZZO DI FUNZIONI ANONIME, CHIAMATE LAMBDA ASTRAZIONI, CHE POSSONO ESSERE COMBINATE PER FORMARE CALCOLI PIÙ COMPLESSI.

master.D

ORIGINE DELLE FUNZIONI LAMBDA:

NEL CAMPO DELLA PROGRAMMAZIONE, IL TERMINE "LAMBDA" HA PRESO IL NOME DAL CALCOLO LAMBDA E SI RIFERISCE ALLE FUNZIONI ANONIME CHE POSSONO ESSERE DEFINITE SENZA ASSEGNARE LORO UN NOME FORMALE.

QUESTA CARATTERISTICA È STATA ADOTTATA IN DIVERSI LINGUAGGI DI PROGRAMMAZIONE, INCLUSI LISP, SCHEME E PYTHON, OFFRENDO UN MODO PIÙ CONCISO ED ELEGANTE PER DEFINIRE FUNZIONI SEMPLICI E IMMEDIATE.

master.D

UNA FUNZIONE LAMBDA È UNA PICCOLA FUNZIONE ANONIMA.

UNA FUNZIONE LAMBDA PUÒ ACCETTARE QUALSIASI NUMERO DI ARGOMENTI, MA PUÒ AVERE SOLO UN'ESPRESSIONE.

SINTASSI

LAMBDA ARGUMENTS : EXPRESSION

master.D

L'ESPRESSIONE VIENE ESEGUITA E VIENE RESTITUITO IL RISULTATO:

AGGIUNGI 10 A ARGUMENT A E RESTITUISCI IL RISULTATO:

```
1. X = LAMBDA A : A + 10
2. PRINT(X(5))
3.
```

master.D

LE CARATTERISTICHE PRINCIPALI DELLE FUNZIONI LAMBDA IN PYTHON INSIEME AD ALCUNI ESEMPI DI CODICE:

FUNZIONI ANONIME: LE FUNZIONI LAMBDA SONO ANONIME, IL CHE SIGNIFICA CHE NON È NECESSARIO ASSEGNARLE UN NOME QUANDO VENGONO DEFINITE.

1. ADD = LAMBDA X, Y: X + Y
2. PRINT(ADD(2, 3)) # OUTPUT: 5

master.D

ESPRESSIONI SINGOLE:

LE FUNZIONI LAMBDA POSSONO CONTENERE SOLO UN'ESPRESSIONE, CHE VIENE VALUTATA E RESTITUITA COME RISULTATO.

```
1. SQUARE = LAMBDA X: X ** 2
2. PRINT(SQUARE(5))  # OUTPUT: 25
```

master.D

SUPPORTO PER ARGOMENTI MULTIPLI:

LE FUNZIONI LAMBDA POSSONO ACCETTARE PIÙ ARGOMENTI SEPARATI DA VIRGOLE.

1. `MULTIPLY = LAMBDA X, Y, Z: X * Y * Z`
2. `PRINT(MULTIPLY(2, 3, 4))  # OUTPUT: 24`

master.D

UTILIZZO CON FUNZIONI DI ORDINE SUPERIORE:

LE FUNZIONI LAMBDA SONO SPESSO UTILIZZATE INSIEME A FUNZIONI DI ORDINE SUPERIORE COME "MAP", "FILTER" E "REDUCE".

```
1. NUMBERS = [1, 2, 3, 4, 5]
2. SQUARED_NUMBERS = LIST(MAP(LAMBDA X: X ** 2, NUMBERS))
3. PRINT(SQUARED_NUMBERS)  # OUTPUT: [1, 4, 9, 16, 25]
```

master.D

RITORNO IMPLICITO:

IL RISULTATO DELL'ESPRESSIONE È IL VALORE DI RITORNO DELLA FUNZIONE LAMBDA.

1. `GET_FIRST_ITEM = LAMBDA ITEMS: ITEMS[0]`
2. `PRINT(GET_FIRST_ITEM([10, 20, 30]))  # OUTPUT: 10`

master.D

LE FUNZIONI LAMBDA OFFRONO UNA POTENTE FLESSIBILITÀ E CHIAREZZA NEL CODICE, CONSENTENDO DI CREARE FUNZIONI BREVI E SPECIFICHE IN MODO PIÙ DIRETTO RISPETTO ALLE FUNZIONI TRADIZIONALI DEFINITE CON "DEF".

master.D

ORDINAMENTO DI UNA LISTA DI TUPLE SECONDO UN ELEMENTO SPECIFICO

IMMAGINA DI AVERE UNA LISTA DI TUPLE, OGNUNA CONTENENTE IL NOME DI UNA PERSONA E LA SUA ETÀ. VOGLIAMO ORDINARE QUESTA LISTA IN BASE ALL'ETÀ DELLE PERSONE UTILIZZANDO LE FUNZIONI LAMBDA.

```
1. PEOPLE = [("ALICE", 25), ("BOB", 30), ("CHARLIE", 22), ("DAVID", 27)]
2.
3. # UTILIZZIAMO LA FUNZIONE LAMBDA PER SPECIFICARE CHE VOGLIAMO
   ORDINARE IN BASE ALL'ETÀ (SECONDO ELEMENTO DELLA TUPLA).
4. SORTED_PEOPLE = SORTED(PEOPLE, KEY=LAMBDA PERSON: PERSON[1])
5.
6. PRINT(SORTED_PEOPLE)
```

master.D

SPIEGAZIONE:

NELL'ESEMPIO SOPRA, ABBIAMO UTILIZZATO LA FUNZIONE SORTED() DI PYTHON PER ORDINARE LA LISTA PEOPLE. ABBIAMO FORNITO IL PARAMETRO KEY CHE DEFINISCE UNA FUNZIONE DI TRASFORMAZIONE PER GLI ELEMENTI DELLA LISTA PRIMA DI EFFETTUARE IL CONFRONTO DI ORDINAMENTO.

LA FUNZIONE LAMBDA LAMBDA PERSON: PERSON[1] PRENDE OGNI TUPLA (PERSON) E RESTITUISCE IL SUO SECONDO ELEMENTO (L'ETÀ).

IN QUESTO MODO, LA FUNZIONE SORTED() ORDINA LA LISTA PEOPLE IN BASE ALL'ETÀ DELLE PERSONE, OTTENENDO COME RISULTATO LA LISTA SORTED_PEOPLE ORDINATA IN MODO CRESCENTE DI ETÀ.

master.D

FILTRAGGIO DI NUMERI PARI DA UNA LISTA

SUPPONIAMO DI AVERE UNA LISTA DI NUMERI INTERI E VOGLIAMO FILTRARE SOLO I NUMERI PARI UTILIZZANDO LA FUNZIONE FILTER() CON UNA FUNZIONE LAMBDA.

```
1. NUMBERS = [1, 5, 8, 10, 15, 20, 25]
2.
3. # UTILIZZIAMO LA FUNZIONE LAMBDA PER FILTRARE SOLO I NUMERI
   PARI.
4. EVEN_NUMBERS = LIST(FILTER(LAMBDA X: X % 2 == 0, NUMBERS))
5.
6. PRINT(EVEN_NUMBERS)
```

master.D

SPIEGAZIONE:

NELL'ESEMPIO SOPRA, ABBIAMO UTILIZZATO LA FUNZIONE FILTER() DI PYTHON INSIEME A UNA FUNZIONE LAMBDA PER FILTRARE SOLO I NUMERI PARI DALLA LISTA NUMBERS.

LA FUNZIONE LAMBDA LAMBDA X: X % 2 == 0 PRENDE CIASCUN NUMERO (X) NELLA LISTA E RESTITUISCE TRUE SE IL NUMERO È PARI E FALSE ALTRIMENTI.

LA FUNZIONE FILTER() QUINDI MANTIENE SOLO GLI ELEMENTI DELLA LISTA NUMBERS PER I QUALI LA FUNZIONE LAMBDA RESTITUISCE TRUE, OTTENENDO COSÌ COME RISULTATO LA LISTA EVEN_NUMBERS CONTENENTE SOLO I NUMERI PARI.

master.D

IN TUTTE QUESTE SITUAZIONI, L'USO DELLE FUNZIONI LAMBDA RENDE IL CODICE PIÙ COMPATTO E LEGGIBILE, EVITANDO LA NECESSITÀ DI DEFINIRE FUNZIONI CON NOMI SEPARATI SOLO PER SCOPI TEMPORANEI O SEMPLICI OPERAZIONI DI TRASFORMAZIONE.

TUTTAVIA, È FONDAMENTALE UTILIZZARE LE FUNZIONI LAMBDA CON PARSIMONIA E NEL RISPETTO DELLA CHIAREZZA DEL CODICE, POICHÉ FUNZIONI PIÙ COMPLESSE POSSONO RENDERE IL CODICE MENO COMPRENSIBILE E MANCARE DI ESPRESSIVITÀ A CAUSA DELL'ASSENZA DI NOMI DESCRITTIVI PER LE FUNZIONI.

master.D

LE FUNZIONI LAMBDA, PUR ESSENDO STRUMENTI POTENTI E FLESSIBILI, NON HANNO TUTTE LE FUNZIONALITÀ E LA COMPLESSITÀ DELLE FUNZIONI DEFINITE CON DEF. TUTTAVIA, POSSONO ESSERE UTILIZZATE IN MODO AVANZATO IN DIVERSE SITUAZIONI.

ECCO DUE PARAGRAFI CHE DESCRIVONO ALCUNE DELLE FUNZIONI PIÙ AVANZATE DELLE LAMBDA IN PYTHON:

master.D

UTILIZZO DI FUNZIONI LAMBDA CON IL MODULO FUNCTOOLS

UNA DELLE APPLICAZIONI AVANZATE DELLE FUNZIONI LAMBDA È IL LORO UTILIZZO IN COMBINAZIONE CON IL MODULO FUNCTOOLS. QUESTO MODULO FORNISCE FUNZIONALITÀ AGGIUNTIVE PER LA MANIPOLAZIONE E L'ELABORAZIONE DI FUNZIONI, RENDENDO LE LAMBDA ANCORA PIÙ POTENTI.

UN ESEMPIO COMUNE DI UTILIZZO AVANZATO È L'USO DELLA FUNZIONE FUNCTOOLS.PARTIAL() PER CREARE FUNZIONI PARZIALMENTE APPLICATE.

CON PARTIAL(), POSSIAMO FISSARE ALCUNI DEGLI ARGOMENTI DI UNA FUNZIONE LAMBDA, OTTENENDO UNA NUOVA FUNZIONE CON UN NUMERO INFERIORE DI PARAMETRI.

master.D

```
1. from functools import partial
2.
3. # Definizione di una funzione lambda con due argomenti.
4. add = lambda x, y: x + y
5.
6. # Creazione di una nuova funzione "add_5" con il primo
   argomento fissato a 5.
7. add_5 = partial(add, 5)
8.
9. print(add_5(3))  # Output: 8 (5 + 3)
```

master.D

IN QUESTO ESEMPIO, ADD_5 È UNA NUOVA FUNZIONE OTTENUTA DALLA FUNZIONE LAMBDA ADD APPLICANDO PARTIAL() CON IL PRIMO ARGOMENTO FISSATO A 5.

QUESTO APPROCCIO È UTILE QUANDO SI DESIDERA CREARE FUNZIONI SPECIALIZZATE DA FUNZIONI PIÙ GENERICHE SENZA DOVER RIPETERE LA DEFINIZIONE DI NUOVE FUNZIONI.

master.D

UTILIZZO DI FUNZIONI LAMBDA ALL'INTERNO DI STRUTTURE DATI COMPLESSE

LE FUNZIONI LAMBDA POSSONO ESSERE UTILIZZATE PER DEFINIRE FUNZIONI PERSONALIZZATE ALL'INTERNO DI STRUTTURE DATI COMPLESSE COME DIZIONARI E LISTE.

QUESTO CONSENTE DI OTTENERE STRUTTURE DATI ALTAMENTE DINAMICHE ED ESPANDIBILI.

```
# Dizionario con funzioni lambda come valori.
operation_dict = {
    "add": lambda x, y: x + y,
    "subtract": lambda x, y: x - y,
    "multiply": lambda x, y: x * y
}

# Utilizzo delle funzioni lambda definite nel dizionario.
result_add = operation_dict["add"](5, 3)
 # Output: 8 (5 + 3)

result_multiply = operation_dict["multiply"](4, 6)  # Output:
24 (4 * 6)
```

master.D

IN QUESTO ESEMPIO, ABBIAMO CREATO UN DIZIONARIO OPERATION_DICT CON CHIAVI CORRISPONDENTI ALLE OPERAZIONI MATEMATICHE ("ADD", "SUBTRACT" E "MULTIPLY") E VALORI CORRISPONDENTI A FUNZ

IONI LAMBDA CHE IMPLEMENTANO QUESTE OPERAZIONI. QUESTO APPROCCIO PUÒ ESSERE UTILE QUANDO SI DESIDERA DEFINIRE FACILMENTE UN INSIEME DI FUNZIONI PERSONALIZZATE ALL'INTERNO DI UNA STRUTTURA DATI E RICHIAMARLE DINAMICAMENTE.

master.D

SINTESI:

LE FUNZIONI LAMBDA IN PYTHON SONO UTILIZZATE PER CREARE FUNZIONI ANONIME CON UNA SINTASSI COMPATTA.

SONO PARTICOLARMENTE UTILI QUANDO SI LAVORA CON FUNZIONI DI ORDINE SUPERIORE, COME MAP(), FILTER(), SORTED() E ALTRE, IN CUI È NECESSARIO PASSARE UNA FUNZIONE COME ARGOMENTO.

LE FUNZIONI LAMBDA SEMPLIFICANO IL CODICE QUANDO SI DESIDERA DEFINIRE RAPIDAMENTE UNA FUNZIONE SEMPLICE SENZA LA NECESSITÀ DI ASSEGNARLE UN NOME.

master.D

SONO POTENTI E VERSATILI E POSSONO RENDERE IL CODICE PIÙ CONCISO E LEGGIBILE, SOPRATTUTTO QUANDO SI AFFRONTANO OPERAZIONI SEMPLICI E IMMEDIATE.

TUTTAVIA, È IMPORTANTE UTILIZZARLE CON CAUTELA, POICHÉ L'ECCESSIVO UTILIZZO DI FUNZIONI LAMBDA COMPLESSE PUÒ RENDERE IL CODICE MENO COMPRENSIBILE A CAUSA DELL'ASSENZA DI NOMI DESCRITTIVI PER LE FUNZIONI.