8. ΔΗΛΩΣΕΙΣ ΕΛΕΓΧΟΥ – ΕΜΠΕΔΩΣΗ ΘΕΩΡΙΑΣ

8.0.1 Διευκρινήσεις θέματος 7

Στην Python, μπορούμε να ορίσουμε προαιρετικές παραμέτρους σε μια συνάρτηση παρέχοντας τιμές γι' αυτές τις παραμέτρους. Αυτό σημαίνει ότι αν μια τιμή για μια προαιρετική παράμετρο δεν παρέχεται όταν καλείται η συνάρτηση, θα χρησιμοποιηθεί η παρεχόμενη τιμή.

Ο αριθμός των προαιρετικών παραμέτρων στην Python δεν είναι συγκεκριμένος, είναι αόριστος, αλλά είναι καλή πρακτική να είναι μικρός, ώστε να μην υπάρχει σύγχυση στους χρήστες της συνάρτησης. Οι προαιρετικές παράμετροι μπορούν να οριστούν με οποιαδήποτε σειρά, αρκεί να βρίσκονται μετά από τις απαιτούμενες παραμέτρους.

Παράδειγμα:

```
def my_function(x, y=10, z=20):
    print("x =", x)
    print("y =", y)
    print("z =", z)
```

Όταν αυτή η συνάρτηση καλείται με μόνο μια παράμετρο, το x μπαίνει σ' αυτή την παράμετρο και τα y και z, παίρνουν τις προεπιλεγμένες τιμές τους.

Έξοδος:

```
x = 5
y = 10
```

z = 20

Όταν καλείται με δύο παραμέτρους, τυπώνονται αυτές και κατόπιν η προεπιλεγμένη τιμή του z:

my_function(5, 30)

Μπορούμε να ορίσουμε προαιρετικές παραμέτρους και χωρίς να τους δώσουμε τιμές. Αυτές οι παράμετροι καλούνται positional arguments και πρέπει να δηλώνονται με τη σειρά που έχουν δωθεί.

Παράδειγμα:

```
def my_function(x, y, z=10):
    print("x =", x)
    print("y =", y)
    print("z =", z)
```

Όταν αυτή η συνάρτηση καλείται με 2 ορίσματα, τα x και y παίρνουν αυτές τις θέσεις και το z παίρνει την προεπιλεγμένη τιμή του:

my_function(5, 30)

Έξοδος:

x = 5

y = 30

z = 10

Όταν καλείται με 3 ορίσματα, και οι τρεις παράμετροι παίρνουν τις τιμές των ορισμάτων:

my_function(5, 30, 40)

Έξοδος:

x = 5

y = 30

z = 40

8.0.2 Λύσεις προηγούμενων ασκήσεων

Βρίσκονται στο αρχείο 7_Λύσεις των ασκήσεων.docx

8.1 Δηλώσεις ελέγχου try, except

Η δήλωση ελέγχου try, except είναι μια γνωστή δήλωση στον προγραμματισμό. Αυτή η δήλωση ελέγχει πώς συνεχίζει ένα πρόγραμμα αφού παρουσιαστεί ένα σφάλμα. Η σύνταξη έχει ως εξής:

try:

κάνε κάτι

except:

κάνε κάτι άλλο όταν παρουσιαστεί ένα σφάλμα

Για παράδειγμα, ας δοκιμάσουμε να εκτελέσουμε το παρακάτω πρόγραμμα:

```
try:
   answer = 12/0
   print (answer)
except:
   print ("An error occurred")
```

Αν εκτελέσουμε το πρόγραμμα, θα λάβουμε το μήνυμα "Παρουσιάστηκε σφάλμα".

Αυτό συμβαίνει γιατί όταν το πρόγραμμα προσπαθεί να εκτελέσει την πρόταση answer = 12/0 στο μπλοκ try, παρουσιάζεται σφάλμα αφού δεν μπορούμε να διαιρέσουμε έναν αριθμό με το μηδέν.

Το υπόλοιπο του μπλοκ try αγνοείται και η δήλωση στο μπλοκ except εκτελείται αντ' αυτού.

Αν θέλουμε να εμφανίζονται πιο συγκεκριμένα και σχετικά με το σφάλμα μηνύματα σφάλματος στους χρήστες, μπορούμε να καθορίσουμε τον τύπο σφάλματος μετά τη λέξη-κλειδί except. Ας δούμε τον παρακάτω κώδικα:

```
try:
  userInput1 = int(input("Παρακαλούμε δώστε έναν αριθμό σαν διαιρετέο: "))
  userInput2 = int(input("Παρακαλούμε δώστε ακόμα έναν σαν διαιρέτη του: "))
  answer =userInput1/userInput2
  print ("Η απάντηση είναι ", answer)
  myFile = open("missing.txt", 'r')
except ValueError:
  print ("Σφάλμα: Δεν δώσατε αριθμό")
except ZeroDivisionError:
  print ("Σφάλμα: Δεν μπορεί να γίνει διαίρεση με το μηδέν")
except Exception as e:
  print ("Άγνωστο σφάλμα: ", e)
```

Η παρακάτω λίστα δείχνει τις διάφορες εξόδους για διαφορετικές εισόδους χρηστών.

Το “>>>” δηλώνει την είσοδο του χρήστη και το “=>” υποδηλώνει την έξοδο.

```
>>> Εισάγετε έναν αριθμό: m
```

=> Σφάλμα: Δεν δώσατε αριθμό

Αιτία: Ο χρήστης εισήγαγε μια συμβολοσειρά που δεν μπορεί να μετατραπεί σε έναν ακέραιο. Αυτό είναι ένα ValueError.

Έτσι, εμφανίζεται η δήλωση του μπλοκ except ValueError.

>>> Εισάγετε έναν αριθμό: 12

>>> Εισάγετε έναν άλλο αριθμό: 0

=> Σφάλμα: Δεν μπορεί να γίνει διαίρεση με το μηδέν

Αιτία: userInput2 = 0. Εφόσον δεν μπορούμε να διαιρέσουμε έναν αριθμό με το μηδέν, αυτό είναι ένα ZeroDivisionError.

Έτσι, εμφανίζεται η δήλωση του μπλοκ except ZeroDivisionError

>>> Εισάγετε έναν αριθμό: 12

>>> Εισάγετε έναν άλλο αριθμό: 3

=> Η απάντηση είναι 4.0

=> Άγνωστο σφάλμα: [Errno 2] Δεν υπάρχει τέτοιο αρχείο ή κατάλογος:

'missing.txt'

Αιτία: Ο χρήστης εισάγει αποδεκτές τιμές και η γραμμή

print ("Η απάντηση είναι ", answer) εκτελείται σωστά.

Ωστόσο, η επόμενη γραμμή εμφανίζει ένα σφάλμα ως το αρχείο

missing.txt δεν βρέθηκε.

Επειδή αυτό δεν είναι ***ValueError*** ή ***ZeroDivisionError***, εκτελείται το τελευταίο μπλοκ except.

Το ***ValueError*** και το ***ZeroDivisionError*** είναι δύο από τους πολλούς τύπους προκαθορισμένων σφαλμάτων της Python.

Το ***ValueError*** εμφανίζεται όταν μια ενσωματωμένη λειτουργία ή συνάρτηση λαμβάνει ένα όρισμα που έχει τον σωστό τύπο αλλά ακατάλληλη τιμή.

Το ***ZeroDivisionError*** εμφανίζεται όταν το πρόγραμμα προσπαθεί να διαιρέσει με το μηδέν.

Άλλα κοινά σφάλματα στην Python περιλαμβάνουν τα:

IOError:

Εμφανίζεται όταν μια λειτουργία I/O (όπως η ενσωματωμένη λειτουργία open()) αποτυγχάνει) για ένα λόγο που σχετίζεται με είσοδο/έξοδο, π.χ. "το αρχείο δεν βρέθηκε".

ImportError:

Εμφανίζεται όταν μια δήλωση import δεν μπορεί να βρει το ζητούμενο module.

IndexError:

Εμφανίζεται όταν ένας δείκτης ακολουθίας (π.χ. συμβολοσειρά, λίστα, πλειάδα) είναι εκτός εύρους.

KeyError::

Εμφανίζεται όταν δεν βρίσκεται ένα κλειδί λεξικού.

NameError:

Εμφανίζεται όταν δεν βρεθεί μια τοπική ή καθολική μεταβλητή.

TypeError:

Εμφανίζεται όταν μια λειτουργία ή συνάρτηση εφαρμόζεται σε ένα αντικείμενο ακατάλληλου τύπου.

Για μια πλήρη λίστα όλων των τύπων σφαλμάτων στην Python, μπορείτε να ανατρέξετε εδώ:

https://docs.python.org/3/library/exceptions.html .

Η Python έρχεται επίσης με προκαθορισμένα μηνύματα σφάλματος για κάθε ένα από τα διαφορετικά είδη σφαλμάτων. Εάν θέλουμε να εμφανίσουμε το μήνυμα, χρησιμοποιείτε τη λέξη-κλειδί “as”

μετά τον τύπο σφάλματος.

Για παράδειγμα, για να εμφανίσουμε το προεπιλεγμένο μήνυμα ValueError, γράφουμε:

except ValueError as e:

print (e)

Το e είναι το όνομα της μεταβλητής που έχει εκχωρηθεί στο σφάλμα. Μπορούμε να του δώσουμε κάποιο άλλο όνομα, αλλά είναι κοινή πρακτική η χρήση του e.

Η τελευταία δήλωση except στο πρόγραμμά μας:

except Exception as e:

print ("Unknown error: ", e)

είναι ένα παράδειγμα χρήσης και πάλι προκαθορισμένου μηνύματος σφάλματος. Χρησιμεύει ως ένα τελικό μήνυμα προσπαθώντας να εντοπίσει τυχόν απρόβλεπτα σφάλματα.

8.2 Ασκήσεις για εμπέδωση της θεωρίας

Άσκηση 1 (multiples_of5_8.1)

```
'''
Γράψτε ένα πρόγραμμα το οποίο να ζητάει από τον
χρήστη έναν αριθμό. Κατόπιν να τον πολλαπλασιάζει
με το 5 και να τυπώνει το αποτέλεσμα
```

'''

```
num = int(input("Εισάγετε έναν αριθμό: "))
result = num * 5
print(f"Το γινόμενό του με το 5 είναι: {result}")

# Μπορούμε να χρησιμοποιήσουμε και τους παρακάτω τρόπους
# για να πάρουμε την ίδια εκτύπωση (αφαιρέστε το # για
# να τους δοκιμάσετε

# print("Το γινόμενό του με το 5 είναι:",result)
# print("Το γινόμενό του με το 5 είναι: %s)" %(result))
# print("Το γινόμενό του με το 5 είναι: " + str(result))
```

Άσκηση 2 (name_welcome_8.2)

'''

Ζητήστε από το χρήστη να εισάγει το όνομά

του και τυπώστε ένα καλωσόρισμα.

'''

```
name = input("Ποιο είναι το όνομά σας; ")
print(f"Καλώς ήρθες, {name}!")
```

Άσκηση 3 (name_reversed_8.4)

'''

```
Ζητήστε από το χρήστη να εισάγει το όνομά του
και το επίθετό του και εμφανίστε τα σε αντίστροφη σειρά (όλα τα γράμματα αντίστροφα).
'''

first_name = input("Ποιο είναι το όνομά σας; ")
last_name = input("Ποιο είναι το επίθετό σας; ")
full_name = f"{first_name} {last_name}"
reversed_name = full_name[::-1]
print(f"Το όνομά σας ανάποδα είναι: {reversed_name}")
```

8.3 String slicing

Το slicing είναι μια μέθοδος διαχείρισης συμβολοσειρών της Python με πολλά και χρήσιμα χαρακτηριστικά:

Μπορούμε να πάρουμε μια σειρά χαρακτήρων χρησιμοποιώντας το slicing.

Καθορίζουμε τον δείκτη (index) έναρξης και τον δείκτη (index) τέλους, διαχωρισμένα με άνω και κάτω τελεία, για να επιστρέψουμε ένα μέρος της συμβολοσειράς.

Παίρνουμε τους χαρακτήρες από τη θέση 2 μέχρι τη θέση 5 (μη συμπεριλαμβανόμενη):

```
b = "Hello, World!"
print(b[2:5])
```

Αποτέλεσμα:

```
llo
```

Slicing από την αρχή

Μη συμπεριλαμβάνοντας το index έναρξης, το εύρος θα ξεκινά από τον πρώτο χαρακτήρα:

Παράδειγμα

Πως παίρνουμε τους χαρακτήρες από την αρχή μέχρι τη θέση 5 (δεν συμπεριλαμβάνεται):

```
b = "Hello, World!"
print(b[:5])
```

Slicing μέχρι το τέλος

Αφήνοντας έξω τον τελικό δείκτη (index), το εύρος θα πάει στο τέλος:

Παράδειγμα

Πως παίρνουμε τους χαρακτήρες από τη θέση 2 μέχρι το τέλος:

```
b = "Hello, World!"
print(b[2:])
```

Slicing με Αρνητική ευρετηρίαση (Negative indexing)

Χρησιμοποιήστε αρνητικούς δείκτες για να ξεκινήσετε το slicing από το τέλος της συμβολοσειράς:

Παράδειγμα

Παίρνουμε τους χαρακτήρες:

Από: το "o" στο "World!" (θέση -5)

Μέχρι το "d" (χωρίς να περιλαμβάνεται) στο "World!" (θέση -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Αποτέλεσμα:

```
orl
```

Άσκηση 4 (palindrome_8.2)

```
'''
Γράψτε ένα πρόγραμμα το οποίο να ζητάει
από τον χρήστη μια λέξη και μετά να εξετάζει
αν είναι παλίνδρομο (αν γράφεται και αντίστροφα).
Κατόπιν, να τυπώνει αν είναι ή όχι.
'''
def is_palindrome(word):
    return word == word[::-1]
user_input = input("Εισάγετε μια λέξη ή φράση: ")
if is_palindrome(user_input):
    print("Είναι παλίνδρομο!")
else:
    print("Δεν είναι παλίνδρομο.")
```

~~**Άσκηση 5** (name_reversed_8.4)~~

```
'''
Ζητήστε από το χρήστη να εισάγει το όνομά του
και το επίθετό του και να εμφανίστε τα σε
αντίστροφη σειρά (όλα τα γράμματα).
'''
first_name = input("Ποιο είναι το όνομά σας; ")
last_name = input("Ποιο είναι το επίθετό σας; ")
full_name = f"{first_name} {last_name}"
reversed_name = full_name[::-1]
print(f"Το όνομά σας ανάποδα είναι: {reversed_name}")
```

Άσκηση 6 (max_num_8.5)

```
'''
Ζητήστε από το χρήστη να εισάγει μια λίστα από αριθμούς και
εμφανίστε το μεγαλύτερο στοιχείο της λίστας.
'''
nums = input("Εισάγετε μια λίστα από αριθμούς, χωρισμένους με κόμματα: ")
nums_list = nums.split(",")
max_num = max(nums_list)
print(f"Το μεγαλύτερο στοιχείο της λίστας είναι το: {max_num}")
```

Άσκηση 7 (odd_or_even_8.6)

```
'''
Ζητήστε από το χρήστη να εισάγει έναν
ακέραιο αριθμό και εμφανίστε
αν ο αριθμός είναι άρτιος ή περιττός.
```

```
'''
num = int(input("Εισάγετε έναν ακέραιο αριθμό: "))
if num % 2 == 0:
   print("Ο αριθμός είναι άρτιος.")
else:
   print("Ο αριθμός είναι περιττός.")
```