



ΠΕΡΙΕΧΟΜΕΝΑ:

1. Εξαιρέσεις
 1. Χειρισμός Εξαίρεσης
 2. Τύποι Εξαιρέσεων
 3. Εξαιρέσεις Χρήστη
2. Data Structures: Δέντρο (Tree)
3. Data Project: Συσχετίσεις Part 1

Σπύρος Παπαγιάκουμος

Σμαραγδένιος Χορηγός Μαθήματος

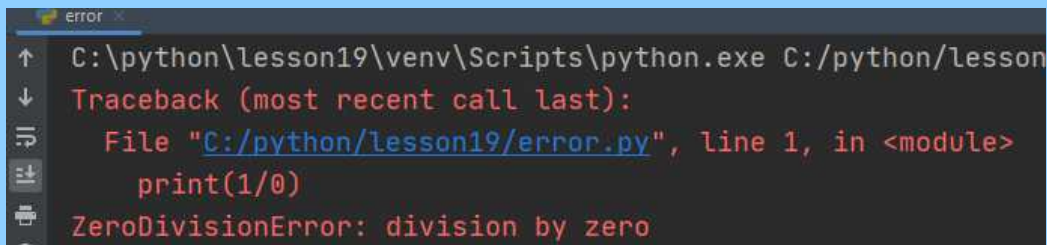
Σωτήρης Μ.

Σμαραγδένιος Χορηγός Μαθήματος

- Όταν προκαλείται ένα σφάλμα κατά την εκτέλεση του προγράμματος π.χ.:

```
# error.py  
print(1/0)
```

- η Python δημιουργεί ένα αντικείμενο-εξαίρεση (exception) (εδώ το ZeroDivisionError) και το “ρίχνει” (throw) σε εμάς:
 - Αν δεν χειριστούμε την εξαίρεση, το πρόγραμμα τερματίζει και μας επιστρέφεται η εξαίρεση που προέκυψε, καθώς και η σειρά εκτέλεσης των συνάρτησεων (traceback) που προκλήθηκε η εξαίρεση.



- Αν χειριστούμε (catch, handle) την εξαίρεση, τότε η εκτέλεση του προγράμματος συνεχίζει κανονικά.
- Ο χειρισμός της εξαίρεσης γίνεται με ένα block try-except:
 - try: Σε αυτό το block βάζουμε τον κώδικα στον οποίο ενδέχεται να προκύψει η εξαίρεση
 - except ErrorName: Εδώ βάζουμε τον κώδικα που θα τρέξει αν προκύψει η εξαίρεση ErrorName

Παράδειγμα 1: try.except.py

```
# try.except.py  
try:  
    nom = int(input("Give the nominator: "))  
    denom = int(input("Give the denominator: "))  
    res = nom/denom  
except ZeroDivisionError:  
    print("Error: Division by zero")
```

Παρατήρηση:

- Τα tracebacks (σημείο κλήσεων των συναρτήσεων) είναι μόνο για τον προγραμματιστή και δεν θα έπρεπε ΠΟΤΕ να φτάσουν στον τελικό χρήστη.
- Είναι δική μας ευθύνη να διαχειριζόμαστε bugs τα οποία μπορεί να προκύψουν και είτε να διακόψουμε την εκτέλεση του προγράμματος, βγάζοντας κάποιο ωραίο μήνυμα, είτε να συνεχίσουμε την εκτέλεση του προγράμματος.

Άσκηση 1:

Ανοίξτε για διάβασμα ένα αρχείο που δεν υπάρχει. Παρατηρήστε το αντικείμενο της εξαίρεσης και έπειτα γράψτε κώδικα το οποίο θα χειρίζεται την εξαίρεση και δεν θα προκαλεί την διακοπή του προγράμματος, αλλά θα ενημερώνει το χρήστη και θα του δίνει τη δυνατότητα στο χρήστη να δημιουργήσει το αρχείο

- Το ολοκληρωμένο συντακτικό είναι:

```
try:
    ...
except ErrorName:
    ...
else:
    ...
finally:
    ...
```

- **try:** Εδώ μπαίνει ο “επίφοβος” κώδικας
- **except:** “Πίανουμε” το σφάλμα. Μπορούμε να γράψουμε:

```
except:
```

(πιάνει οποιαδήποτε εξαίρεση)

```
except (ErrorName1, ErrorName2,...):
```

(για τη διαχείριση πολλών εξαιρέσεων ταυτόχρονα)

```
except ErrorName:
```

```
...
```

```
except ErrorName:
```

```
...
```

(για τη ξεχωριστή διαχείριση κάθε εξαίρεσης)

```
except ZeroDivisionError as ob:
```

(αποθηκεύει το αντικείμενο της εξαίρεσης και μεταξύ άλλων έχουμε πρόσβαση στο default μήνυμά της, τυπώνοντάς την.)

- **else (προαιρετικό):** Κώδικας που τρέχει αν δεν υπάρξει εξαίρεση.
- **finally (προαιρετικό):** Κώδικας που τρέχει είτε μετά το χειρισμό της εξαίρεσης, είτε μετά το “else”.

Παράδειγμα 2: else.finally.py

```
def safe_divide():
    try:
        nom = int(input("Give nominator: "))
        denom = int(input("Give denominator: "))
        res = nom/denom
    except ZeroDivisionError:
        print("Error! Denom is 0")
    except Exception as e:
        print("Something went really wrong: " + str(e))
    else:
        return res
    finally:
        return None

print(safe_divide())
```

Σημείωση: Το Exception είναι η κλάση την οποία κληρονομεί κάθε εξαίρεση. Συνεπώς η εκτύπωση, τυπώνει το μήνυμα του αντικειμένου – εξαίρεσης, παιδιού της Exception

- Το σύστημα εξαιρέσεων της Python περιέχει πολλούς τύπους εξαιρέσεων, π.χ.:
 - **IndexError**: Όταν κάνουμε π.χ. πρόσβαση σε λίστα και ο δείκτης είναι εκτός ορίων.
 - **SyntaxError**: Όταν υπάρχει συντακτικό λάθος στο πρόγραμμα.
 - **FileNotFoundError**: Όταν δεν υπάρχει αρχείο το οποίο προσπαθούμε να ανοίξουμε.
 - **ImportError**: Όταν δεν υπάρχει module το οποίο προσπαθούμε να ενσωματώσουμε.
 - **ZeroDivisionError**: Διαίρεση με το 0.
 - **MemoryError**: Όταν προκαλείται σφάλμα μνήμης.
 - **ValueError**: Όταν μία συνάρτηση δέχεται όρισμα το οποίο είναι σωστού τύπου, αλλά με λαθος τιμή.
- και πολλά ακόμη που έχουν οριστεί στο documentation της Python (<https://docs.python.org/3/library/exceptions.html>) (βλ. ειδικά το δέντρο των κλάσεων στο τέλος της σελίδας)

Πρόκληση εξαίρεσης:

- Μπορούμε να προκαλέσουμε και οι ίδιοι οποιαδήποτε built-in εξαίρεση με τη raise.
- Π.χ. εδώ προκαλούμε ValueError:

```
raise ValueError("NonNegative Value entered")
```
- Στο όρισμα παίρνει το λεκτικό που θέλουμε να περιέχει το αντικείμενο - εξαίρεση.

Παράδειγμα 3: raise.py

```
while True:
    try:
        x = int(input("Give nonnegative integer: "))
        if x < 0:
            raise ValueError("NonNegative Value entered")
    except ValueError as v:
        print(v)
    else:
        print(x)
        break
```

Άσκηση 2:

Κατασκευάστε τη συνάρτηση get_integer():

- Να προτρέπει το χρήστη να εισάγει έναν ακέραιο.
- Να προκαλεί ValueError αν εισαχθεί κενό string. Να εκτυπώνεται το μήνυμα "No digits entered"
- Να προκαλεί ValueError αν εισαχθεί κάτι που δεν είναι ψηφίο. Να εκτυπώνεται το μήνυμα "Wrong Input. Only digits please"
- Να διαχειρίζεται απρόβλεπτα λάθη εκτός των δύο παραπάνω.

Σε κάθε περίπτωση λάθους εισόδου, να επαναλαμβάνεται η διαδικασία. Αν όλα πάνε καλά, τότε να επιστρέφεται ο ακέραιος που διαβάστηκε.

- Μπορούμε να ορίσουμε και δικές μας εξαιρέσεις, απλά κληρονομώντας την κλάση Exception.
- Σημεία που πρέπει να προσέξουμε όταν ορίζουμε μια εξαίρεση χρήστη:
 - Όνομα Εξαίρεσης: Δίνουμε ένα περιγραφικό όνομα στην κλάση.
 - Δημιουργούμε κλάση, άρα μπορούμε να την επεκτείνουμε ορίζοντας επιπλέον μέλη, κάνοντας υπολογισμούς κ.λπ.
 - Αρχικοποιούμε την υπερκλάση Exception με το περιγραφικό μήνυμα που θέλουμε να εμφανίζεται τελικά.
 - `__str__`: Εδώ εμφανίζεται το μήνυμα από την υπερκλάση, αλλά μπορούμε να το επαναορίσουμε κατά βούληση.

Παράδειγμα 4: user.exception.py

Ορίζουμε μια δική μας κατηγορία εξαίρεσης κληρονομώντας την Exception:

```
class MyException(Exception):
    def __init__(self, val, message):
        self.val = val
        self.message = message

    def __str__(self):
        return f"{self.message}: {str(self.val)} is not valid"
```

και έπειτα μπορούμε να τη χρησιμοποιήσουμε σαν οποιαδήποτε άλλη εξαίρεση:

```
try:
    val = int(input("Give an integer: "))
    if val < 0:
        raise MyException(val, "Invalid Value")
except MyException as e:
    print(e)
```

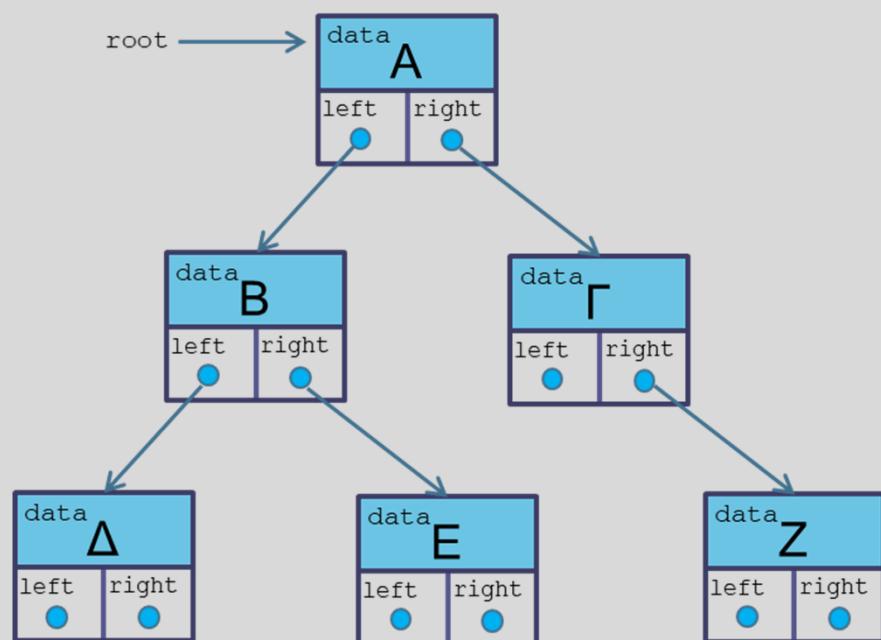
Άσκηση 3:

Τροποποιήστε τη συνάρτηση `get_integer()` στην `get_integer_con()` ώστε να δέχεται έναν ακέραιο που να είναι από 100 έως 200 και πολλαπλάσιο του 5:

- Να προτρέπει το χρήστη να εισάγει έναν ακέραιο.
- Να προκαλεί τα ίδια λάθη με την προηγούμενη έκδοση, και επιπλέον:
 - `ValueTooSmallError`: Αν δοθεί τιμή μικρότερη του 100
 - `ValueTooLargeError`: Αν δοθεί τιμή μεγαλύτερη του 200
 - `NotMultipleOfFiveError`: Αν δοθεί τιμή που δεν είναι πολλαπλάσιο του 5. Ειδικά γι' αυτό το λάθος η εκτύπωση να παρουσιάζει τον αριθμό που πληκτρολογήθηκε, ακολουθούμενο από το μήνυμα ότι δεν είναι πολλαπλάσιο του 5.

Άσκηση 4.1: Κλάση Κόμβος Δένδρου

Το δένδρο είναι μια δομή δεδομένων με μη γραμμική διάταξη, στην οποία κάθε κόμβος είναι: τα δεδομένα και αναφορές στο αριστερό και το δεξί παιδί (None, αν δεν υπάρχουν)



Ορίστε την Κλάση Node (σε αρχείο tree.py):

- Μέλη: data, left, right
- Μέθοδος: init (με default στα left, right: None)

Άσκηση 4.2: Κλάση Δένδρο

Ορίστε την Κλάση Tree (στο ίδιο αρχείο):

- Μέλος: root (αρχικοποιείται σε None)
- Μέθοδος insert_root(data): Χρησιμοποιείται μόνο όταν το δένδρο είναι άδειο, για να εισάγουμε τον πρώτο κόμβο (ρίζα)
- Μέθοδος insert_left(node, data): Εισάγει έναν νέο κόμβο με περιεχόμενο data ως το αριστερό παιδί του κόμβου node.
- Μέθοδος insert_right(node, data): Εισάγει έναν νέο κόμβο με περιεχόμενο data ως το δεξί παιδί του κόμβου node.
- Μέθοδος __str__(): Τυπώνει τα περιεχόμενα του δένδρου. Π.χ. το δένδρο της διαφάνειας θα τυπώνεται ως εξής (σκεφθείτε αναδρομικά)

```
A(B(Δ(_),E(_)),Γ(_Z(_)))
```

Άσκηση 4.3: Κλάση Δένδρο Αναζήτησης (προχωρ. άσκηση => βλ. βιντεο)

Ορίστε την υποκλάση BinarySearchTree του Tree (στο ίδιο αρχείο):

- Μέθοδος insert(): Εισάγει ένα νέο δεδομένο στο δένδρο ως εξής: Ξεκίνα από τη ρίζα. Αν η τιμή του δεδομένου είναι μικρότερη από τον κόμβο εισήγαγε το στο αριστερό υποδένδρο. Αν η τιμή του δεδομένου είναι μεγαλύτερη από τον κόμβο, εισήγαγε το στο δεξί υποδένδρο. Αν είναι ίδια, τότε μην το εισάγεις.
- Μέθοδος inorder(): Επιστρέφει μία λίστα με τις τιμές των κόμβων με τον εξής αναδρομικό κανόνα: Η λίστα να έχει πρώτα το αριστερό υποδένδρο, μετά τη ρίζα και τελικά το δεξιό υποδένδρο.

Άσκηση 5.1: Οντότητα “Μάθημα”

Κατασκευάζουμε μία νέα κλάση με το όνομα “Μάθημα”

- που θα αποθηκεύει το όνομα του μαθήματος και θα έχει τη γνωστή λειτουργικότητα CRUD
- Κατασκευάστε αντίστοιχα μία κλάση Lesson, μία κλάση Lessons και το αρχείο lessons.json

Άσκηση 5.2: Συσχέτιση M:N

Ένα μάθημα σχετίζεται με τους μαθητές που έχουν γραφτεί σε αυτό. Λέμε ότι είναι μία συσχέτιση M:N διότι σε ένα μάθημα γράφονται πολλοί μαθητές (και ένας μαθητής γράφεται σε πολλά μαθήματα). Μια συσχέτιση M:N μπορεί να υλοποιηθεί (με τις μέχρι τώρα γνώσεις μας):

- διατηρώντας σε κάθε μαθητή μία λίστα με τα id των μαθημάτων στα οποία έχει εγγραφεί.
- είτε διατηρώντας μία λίστα με τα id των μαθητών σε κάθε μάθημα.

Επιλέγοντας το 2ο τρόπο:

- Ορίστε στο μάθημα μία λίστα με τα ids των μαθητών που έχουν γραφτεί σε αυτό
- Αντίστοιχα ορίστε μία λίστα με τα ids των καθηγητών που διδάσκουν το συγκεκριμένο μάθημα.

Υλοποιήστε όλες τις CRUD πράξεις για το μάθημα, με ειδική μέριμνα για την ενημέρωση, ώστε να προστίθεται ή να αφαιρείται καθηγητής ή μαθητής από το μάθημα.

Ενημερώστε το κυρίως πρόγραμμα ώστε να γίνεται αυτή η διαχείριση.