Εισαγωγή στη γλώσσα προγραμματισμού Python

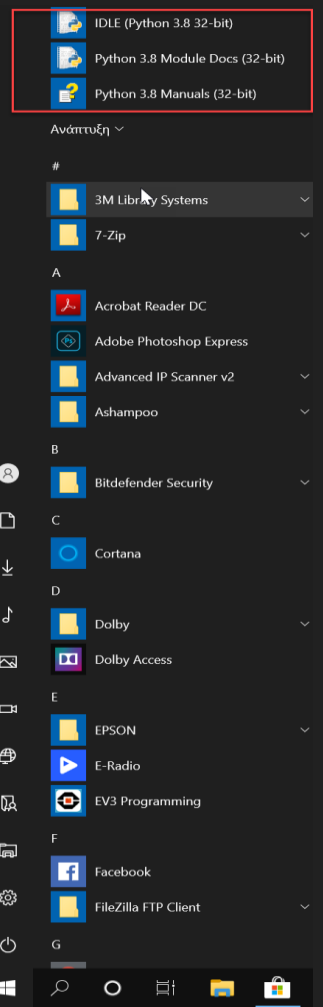
Τι είναι η Python;

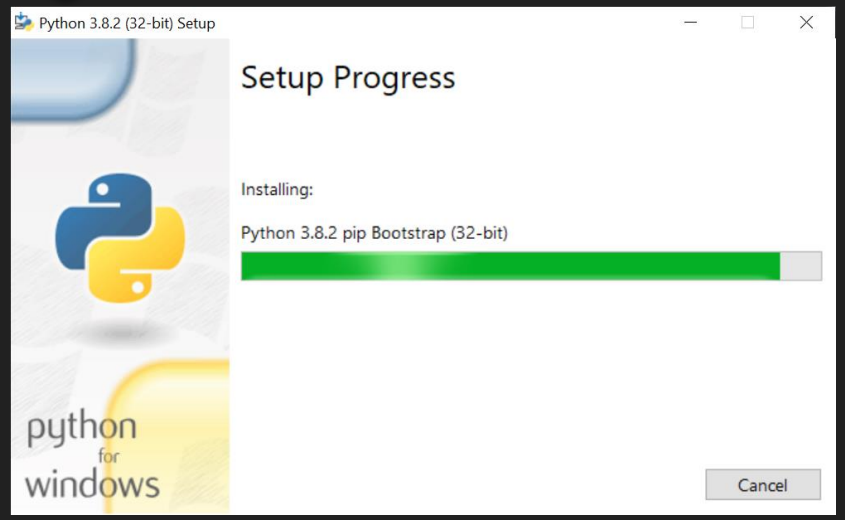
- Είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού
- Υψηλού επιπέδου 4ης γενιάς
- Υποστηρίζει δομές δεδομένων
- Ευανάγνωστη σε σχέση με τις περισσότερες γλώσσες προγραμματισμού

Που χρησιμοποιείται;

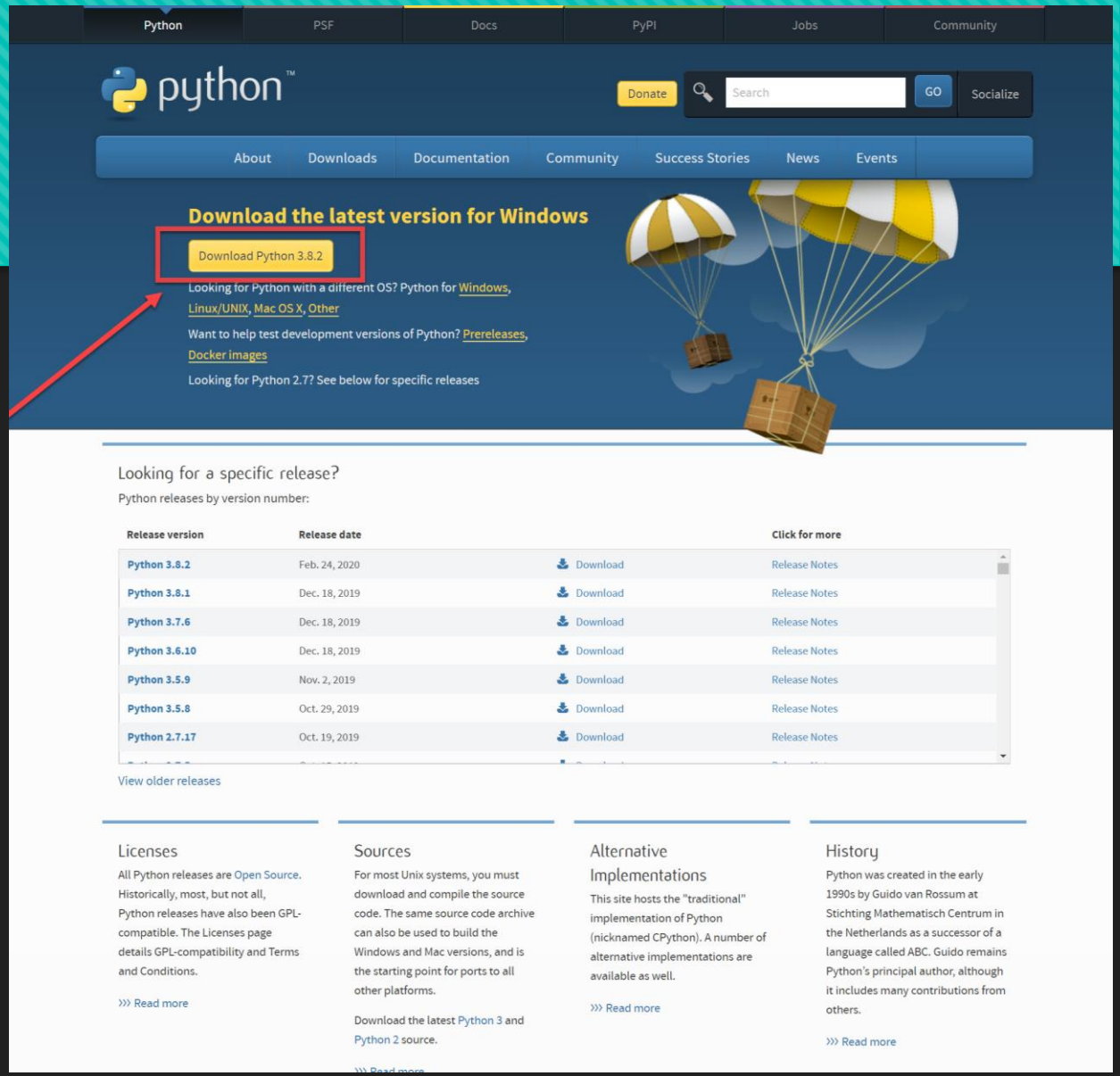
- Σε ιστοσελίδες και διαδικτυακές εφαρμογές
- Δημιουργία video παιχνιδιών
- Στην μηχανική εκμάθηση και στην τεχνητή νοημοσύνη
- Σε GUI: γραφικό περιβάλλον χρήστη
- Οπτικοακουστικές εφαρμογές

Περιβάλλον Python





https://www.python.org/downloads/



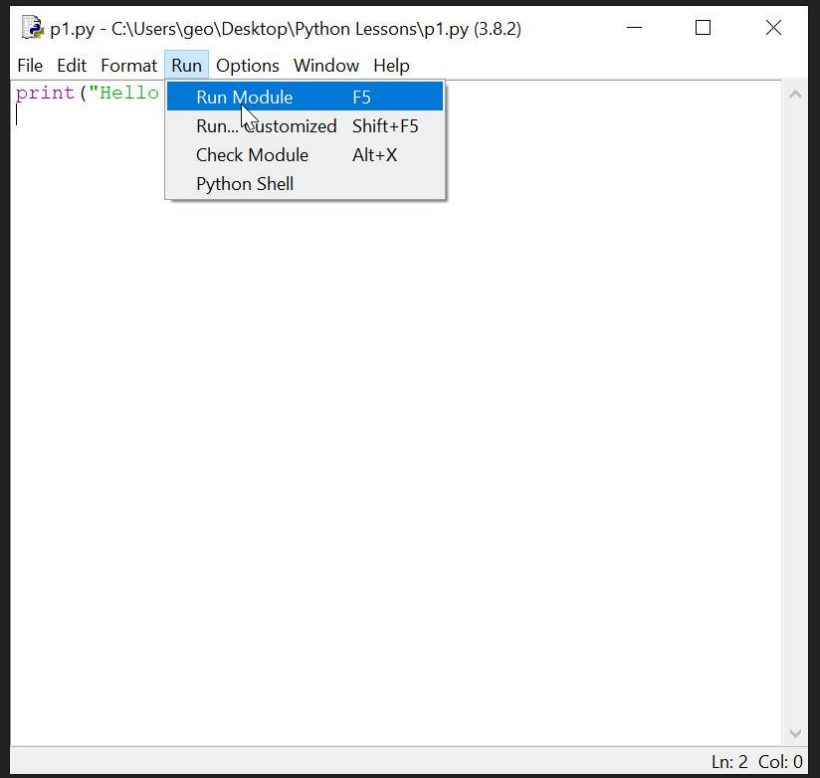
p1.py

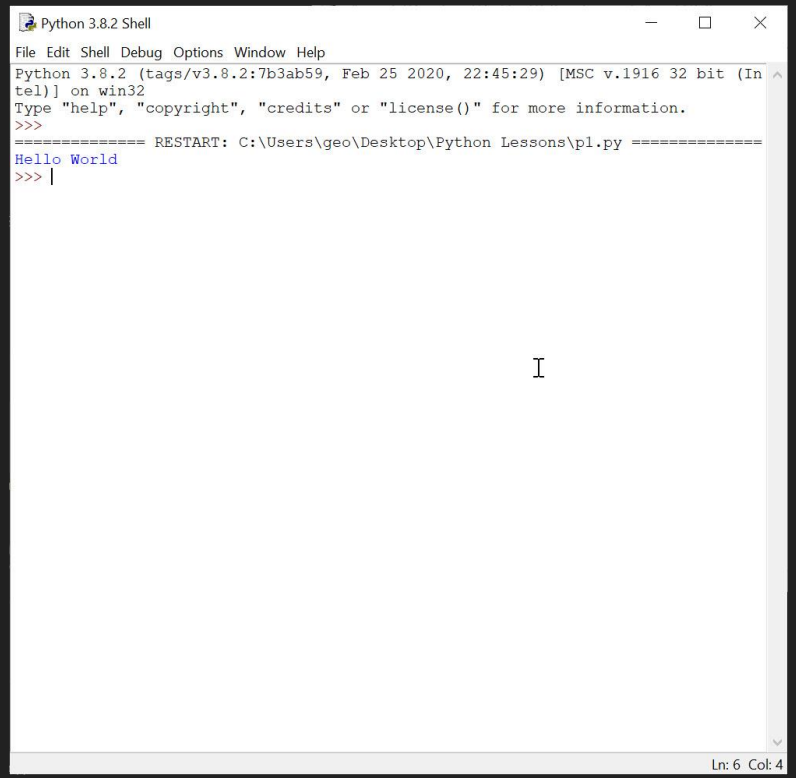
Hello World!

- Το πρώτο μας πρόγραμμα στην Python

```
print("Hello, World!")
```

```
C:\Users\My Name>python demo_default.py
Hello, World!
```





p2.py

Μεταβλητές

Στην Python οι μεταβλητές υλοποιούνται την στιγμή που τους αναθέτουμε μια τιμή

```
x = 4
x = "Sally"
print(x)
```

Στην παραπάνω περίπτωση το x πήρε τιμή ακέραιου (4) και στη συνέχεια πήρε τιμή αλφαριθμητικού (“Sally”) και στο τέλος εκτύπωσε Sally

```
C:\Users\My Name>python demo_variables2.py
Sally
```

p3.py

Παγκόσμιες Μεταβλητές + Μέθοδοι

Στο παρακάτω έχουμε μια παγκόσμια μεταβλητή την x μετά μια μέθοδο myfunc στην οποία δημιουργούμε μια τοπική μεταβλητή x και έπειτα καλούμε μια τοπική print.

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

Στη συνέχεια καλούμε την myfunc που εκτυπώνει "Python is fantastic" επειδή παίρνει την τοπική x και στην συνέχεια έχουμε μια print που παίρνει την παγκόσμια μεταβλητή και εκτυπώνει "Python is awesome".

```
C:\Users\My Name>python demo_variables_global2.py
Python is fantastic
Python is awesome
```

Τύποι δεδομένων

- Μπορείτε να πάρετε τον τύπο δεδομένων ενός αντικειμένου με τη χρήση την type()

```
x = 5
print(type(x))
```

```
C:\Users\My Name>python demo_type.py
<class 'int'>
```

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

p4.py

Εκτύπωση στην οθόνη

- Στη παρακάτω εικόνα χρησιμοποιούμε τα κελιά {} με έναν αριθμό μέσα τους για να προσδιορίσουμε με ποια σειρά θέλουμε να εμφανιστούν οι μεταβλητές μας
- Η format() παίρνει οποιονδήποτε αριθμό μεταβλητών και τους μετατρέπει σε string

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

```
C:\Users\My Name>python demo_string_format3.py
I want to pay 49.95 dollars for 3 pieces of item 567
```

Operators 1/2

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Operators 2/2

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Λίστες και πως να τις προσπελάσουμε

- Μια λίστα είναι μια συλλογή που είναι ταξινομημένη και μεταβλητή και επιτρέπει διπλότυπα μέλη
- Στον παρακατω κωδικα προσπελάζουμε τα στοιχεια της thislist από το 3 έως το 5

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

```
C:\Users\My Name>python demo_list_range.py
['cherry', 'orange', 'kiwi']
```

- Ενώ εδώ προσπελάζουμε όλα τα στοιχεία της λίστας

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

```
C:\Users\My Name>python demo_list_loop.py
apple
banana
cherry
```

Λίστες πρόσθεση αφαίρεση και αντιγραφή

- Για αν **προσθέσουμε** ένα αντικείμενο σε μια συγκεκριμένη θέση, χρησιμοποιούμε τη insert():

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
C:\Users\My Name>python demo_list_insert.py
['apple', 'orange', 'banana', 'cherry']
```

- Η remove() μέθοδος **αφαιρεί** ένα επιλεγμένο αντικείμενο της λίστας:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
C:\Users\My Name>python demo_list_remove.py
['apple', 'cherry']
```

- Κάνε εάν **αντίγραφο** μιας λίστας με την copy():

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

```
C:\Users\My Name>python demo_list_copy.py
['apple', 'banana', 'cherry']
```

Λίστες

Python has a set of built-in methods that you can use on lists.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

p6.py

Έλεγχος If, elif, και else

- Στον παρακάτω κώδικα έχουμε μια if μια elif και μια else. Όταν πάει στην if θα κοιτάξει αν το b είναι μεγαλύτερο του a.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

- Στην περίπτωση που το αποτέλεσμα δεν είναι αληθές στην if θα κοιτάξει την συνθήκη στην elif. Και τέλος αν και η elif είναι ψευδής τότε θα εκτελέσει τον κώδικα της else.

```
C:\Users\My Name>python demo_if_else.py
a is greater than b
```

Βρόγχοι while και for 1/2

- Στον παρακάτω βρόγχο κάνουμε έχουμε πέντε επαναλήψεις και εκτυπώνουμε τον μετρητή στην οθόνη.

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
C:\Users\My Name>python demo_while.py
1
2
3
4
5
```

- Στον παρακάτω κώδικα χρησιμοποιούμε την break για να διακόψουμε τον βρόγχο όταν το I τα πάρει την τιμή τρία.

```
i = 1
while i < 6:
  print(i)
  if (i == 3):
    break
  i += 1
```

```
C:\Users\My Name>python demo_while_break.py
1
2
3
```

Βρόγχοι while και for 2/2

- Στον παρακάτω βρόγχο χρησιμοποιεί την continue για να διακόψουμε τον υπόλοιπο κώδικα και να συνεχίσουμε με τις επανάληψής στον βρόγχο. Όπως θα παρατηρήσατε στην γραμμή αποτελεσμάτων δεν υπάρχει το τρία.

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
C:\Users\My Name>python demo_while_continue.py
1
2
4
5
6
```

- Στην for χρησιμοποιούμε την range() για να προσδιορίσουμε τον αριθμό που θέλουμε να έχουμε επανάληψη. Στην προκειμένου περίπτωση την κάνουμε να ξεκινάει με το δυο και να φτάνει μέχρι το πέντε. Αν θα θέλαμε να αυξήσουμε τον μετρητή τότε θα είχαμε και μια τρίτη μεταβλητή π.χ. range(2, 6, 3).

```
for x in range(2, 6):
  print(x)
```

```
C:\Users\My Name>python demo_for_range2.py
2
3
4
5
```

p9.py

Μέθοδοι 1/2

- Στις μεθόδους έχουμε την λέξη κλειδί def για να δηλώσουμε μια μέθοδο.
- def *όνομα συνάρτησης*(*παράμετροι*)
- Στο παρακάτω πρόγραμμα έχουμε την my_function με προκαθορισμένη παράμετρο το “Norway”. Στο κύριο πρόγραμμα καλούμε την μέθοδο με διαφορετικές μεταβλητές εκτός από μια που παίρνει την προκαθορισμένη τιμή

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```
C:\Users\My Name>python demo_function_param2.py
I am from Sweden
I am from India
I am from Norway
I am from Brazil
```

Μέθοδοι 2/2

- Στην παρακάτω μέθοδο κάνουμε μια απλή επανάληψη όλων των στοιχείων της παραμέτρου

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

```
C:\Users\My Name>python demo_function_param3.py
apple
banana
cherry
```

- Στο επόμενο πρόγραμμα επιστρέφουμε την μεταβλητή μας πολλαπλασιασμένη με το πέντε. Και χρησιμοποιούμε την print για να το εκτυπώσουμε στην οθόνη.

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
C:\Users\My Name>python demo_function_return.py

15
25
45
```

p11.py

Κλάσεις και αντικείμενα 1/2

- Έχουμε μια κλάση την Person και στη συνέχεια υλοποιούμε τον constructor της με την _init_(). Η πρώτη μεταβλητή της _init_() είναι το στιγμιότυπο αντικειμένου της κλάσης, ότι ακολουθεί έπειτα είναι μεταβλητές της κλάσεις.
- Στον constructor αρχικοποιούμε τα μέλη της κλάσης με τις τιμές που θα έχουν τα name και age.

```
class Person:
  def __init__(mysillyobject, name, age):
    mysillyobject.name = name
    mysillyobject.age = age

  def myfunc(abc):
    print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

```
C:\Users\My Name>python demo_class5.py
Hello my name is John
```

Κλάσεις και αντικείμενα 2/2

- Αν θέλουμε να **αλλάξουμε** ένα από τα μέλη του αντικειμένου μας βάζουμε μια τελεία μπροστά από το αντικείμενο στο κύριος πρόγραμμα μας και του αναθέτουμε μια ανάλογη τιμή

```
p1.age = 40
```

- Αν θέλουμε να **διαγράψουμε** την τιμή ενός από τα μέλη του αντικειμένου μας χρησιμοποιούμε την λέξη κλειδί del. Και επιλέγουμε πιο από τα μέλη θα διαγραφεί

```
del p1.age
```

- Στην περίπτωση που θέλουμε να διαγράψουμε ολόκληρο το αντικείμενο βάζουμε μπροστά από την del το στιγμιότυπο του αντικειμένου που θέλουμε να διαγραφεί

```
del p1
```

p12.py

Κληρονομικότητα 1/2

- Η κληρονομικότητα μας επιτρέπει να προσδιορίσουμε μια κλάση που κληρονομεί όλες τις μεθόδους και στοιχεία μια άλλης κλάσης. Η **κλάση γονέα** είναι η κλάση που κληρονομεί στην **κλάση παιδί.**

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)
```

```
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

```
C:\Users\My Name>python demo_inheritance_add_method.py
Welcome Mike Olsen to the class of 2019
```

Κληρονομικότητα 2/2

- Για να δείξουμε ότι η κλάση Student κληρονομεί από την Person βάζουμε μέσα σε παρενθέσεις class Student(Person):
- Η super() αυτομάτως κληρονομεί τα στοιχεία και τις μεθόδους της κλάσης γονέα επιτρέποντας μας να έχουμε άμεση πρόσβαση.

Εξαιρέσεις 1/2

- Η λέξη κλειδί **try** σου επιτρέπει να ελέγξεις ένα κομμάτι κώδικα για πιθανά λάθη
- Η λέξη κλειδί **except** σε βοηθάει πως να χειριστείς την εξαίρεση
- Η **finally** σου επιτρέπει να εκτελέσεις κώδικα, ανεξάρτητος του αποτελέσματος της **try** και της **except**.

```
try:
  print(x)
except:
  print("An exception occurred")
```

```
C:\Users\My Name>python demo_try_except.py
An exception occurred
```

- Ο κώδικας μας δημιουργεί εξαίρεση επειδή η "x" δεν έχει οριστεί ακόμα.

Εξαιρέσεις 2/2

- Όπως και πριν δεν έχουμε ορίσει την "x" οπότε έχουμε εξαίρεση και μετά πηγαίνουμε στο κομμάτι της **finally.**

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

```
C:\Users\My Name>python demo_try_except4.py
Something went wrong
The 'try except' is finished
```

- Με την **raise** μπορούμε να δημιουργήσουμε δίκες μας εξαιρέσεις

```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

```
C:\Users\My Name>python demo_keyword_raise.py
Traceback (most recent call last):
  File "demo_ref_keyword_raise.py", line 4, in <module>
    raise Exception("Sorry, no numbers below zero")
Exception: Sorry, no numbers below zero
```