B: variables and types

Programming with Python

- Simple and *minimal-chic* programming language
- Easy to learn and use: reading code is almost like reading English
- Simplicity as a result of design:
  - focus on the problem and not on the language/the data representation in memory
  - a way to represent an easy-to-read program
- Python is free, open source well-supported and well-maintaned:
  - VS Code, Jupyter, PIP, PyPi, Anaconda, Docker...
- Comes from "Monty Python"

Programming in Python

- Sequence of one or more instructions
- A fixed alphabet, strict syntactical rules and notations
- high-level computer language
- The Python interpreter maps Py. commands, one after the other, into low-level commands
- Computer hardware understands and executes the low-level commands
  - Even adding two numbers requires a certain visibility of RAM and CPU

```
load the number from memory location 2001 into the CPU

load the number from memory location 2002 into the CPU

add the two numbers in the CPU

Store the result into location 2003
```

Programming

- Python is easy to understand
- interpreters translate high-level Py. to machine language
- Two ways to do it, using:
  - Compiler or
  - Interpreter

```
grand_total = price + vat
```

```
print("Welcome to Python")
```

Simple Python commands

- Comments!
  - Use of hashtag **#**
- Print command **print**
  - Prints on screen!
    - Numbers and operations

```
# Command to print number 10!
print(10)

# More commands...
print(10+20)
print(10+2*20)
print((2*4)+6)
print(2**2)
```

Simple Python commands

- **print()**
  - Prints on screen!
  - Text is *containerized* by means of single (‘) or double (“) quotes

```
# prints on the screen:
print(‘Have a nice Autumn term!’)
```

Simple Python commands

- **print()**
  - Prints on screen!
  - Text is *containerized* by means of single (‘) or double (“) quotes

```
# Let’s print two names
print(“Stelios”)
print(“Sotiriadis”)
# Or
print(“Stelios”, “Sotiriadis”)

# Or
print(‘Stelios’, ‘Sotiriadis’)

# Or
print(‘Stelios ’+‘Sotiriadis’)
```

Variables

- Programs manipulate data that sits in the computer memory.
- Variables are generic names for the container of some value
- Computer variables are akin to both parameters and variables of mathematics
- A funny '=' symbol to assign value to a variable

  name = `Nik`

  age = 10

```
print(name)
```

```
x = age + 10
print(x)

print(10/x) # beware division by 0!

print(2*x + y) # What does it print?
```

```
NameError                                 Traceback (most recent call last)
<ipython-input-1-3e1c5a381366> in <module>()
      1 x=10
----> 2 print(2*x+y)

NameError: name 'y' is not defined
```

Variables can change their content

- Assignment statement
  - `fahrenheit = 9/5 * celsius + 32`

```
# This is an assignment expression!
myvar = 10
print(myvar)

myvar = 20 # Assign a new value
print(myvar)

myvar = myvar+1
print(myvar) # Prints 21
```

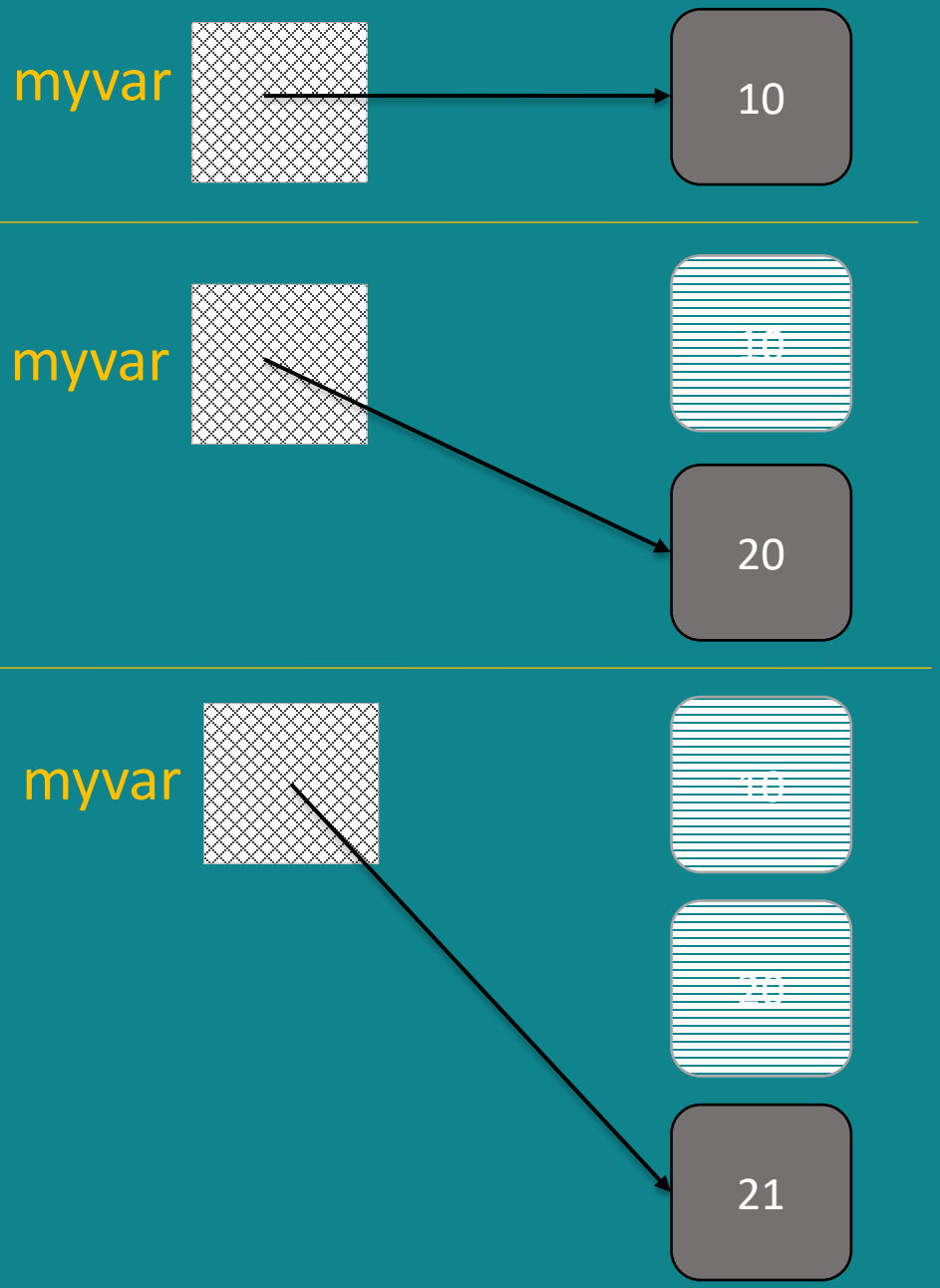
Variables can change!

```
# This is an assignment expression!
myvar = 10
print(myvar)

myvar = 20 # Assign a new value
print(myvar)

myvar = myvar+1
print(myvar) # Prints 21
```

How memory looks like!



Programming is about input/output

- Input from the keyboard:
  - 'enter'/'return' to end the input.

    input()

  - Assign the input to a variable

    <variable> = input(<prompt>)

```
# This is an input example
name = input("Please enter a name: ")
print(name)


# This is another input example
my_fav_num = input("What is your favourite number? ")
print("Your favourite number is ", my_fav_num)
```

Programming is about input/output

- Input from the keyboard:
  - ‘enter’/’return’ to end the input.

    input()

  - Assign the input to a variable

    <variable> = input(<prompt>)

```
# This is another input example
fName = input(“Give first name: ”)
lName = input(“Give surname: ”)
print(fName, lName)
```

Quiz 2: Fill the gaps


```
a) _____________ # Write a statement to print your name

b) _____________ # Write a statement to print number 10

c) _____________ # Write a statement to print the sum of 1
and 2
```

Quiz 2 Solution

a) **print(“Stelios”)** # Write a statement to print your name

b) **print(10)** # Write a statement to print number 10

c) **print(1+2)** Write a statement to print the sum of 1 and 2

Quiz 3: Fill the gaps


```
# Provide the command to prompt the user to enter a name

my_input = d) _________________("Enter your name: ")



# Provide the command to print the variable my_input

e) _________________ (my_input)
```

Quiz 3 Solution

```
# Provide the command to prompt user to enter a name

my_input = d) __ input__(“Enter your name: “)



# Provide the command to print the variable my_input

e) __print__(my_input)
```

Data types

- Integers
  - x=10, y=100 …
- Floats
  - pi=3.14, a=10.10, rate=0.0992929
- Strings (text)
  - name="Stelios", age="32"
- Boolean (True or False, 1 or 0)
  - headphones=True

Working with numbers!

- Everything that the user inputs from the keyboard Python assumes that it is a **TEXT**!

- We need to transform text to numbers to make operations
    - Text to integer
    - Text to float

```
# The input is 10
x = input("Give a number")
print(x)
# x is a text, even if it looks like a number
# This will print 10!


# This is another input
x = input("Give a number")
print(x+20)
# Python does not know that x is a numbers...
# This will cause an error
```

What is the output?

```
# This is an input
x = input("Give a number")
print(x * 2)
# This is how you comment multiple lines...
"""
What does it print if the user enters 10?
   a. 20
   b. An error ...
   c. 1010
"""
```

The correct answer is 1010

Data types conversions

- String to Integer
  - use int(<value>)

```
# This is another input example
x = input("Give a number")
print(int(x) * 2)
""" What does it print if the user enters
10?
   a. 20
   b. An error ...
   c. 1010
"""
```

The correct answer is 20

Data types conversions

- String to Integer
  - use int(<value>)

```
# Hint
# If you except an integer, convert the input to integer directly
x = int(input("Give an integer number"))
print(x * 2) # Will print 20!
```

Data types conversions

- String to Float
  - use float(<value>)

```
# This is another input example
x = input(“Give a number”)
print(float(x) + 2)
# What does it print if the user enters 10.5?
```

The correct answer is 12.5

Data types conversions

- Int to String
  - use str(<value>)

```
x = 10
# What does it print?
print("ID" + x)
```

The correct answer is error

```
# What does it print?
print("a" + str(x))
```

The correct answer is a10

Recap

- Data types:
  - Integers: x=10, y=100 …
  - Floats: pi=3.14, a=10.10, rate=0.0992929
  - Strings (text): name="Stelios", age="32"
  - Boolean (True or False, 1 or 0): headphones=True

- Converting types:
  - int(<value>), float(<value>), str(<value>)

Quiz 4

- What is the output of the following script?

```
a = input("give a number")
# The user enters 10
b = input("give a second number")
# The user enters 20
print(a+b)
```

Quiz 4 Solution

- What is the output of the following script?

```
a = input("give a number")                    # The user enters 10
b = input("give a second number")             # The user enters 20
print(a+b)
```

If input is 10 and then 20 this will print 1020!

```
a = int(input("give a number"))
b = int(input("give a second number"))
print(a+b)
```

This will print 30!