Instituto Politécnico da Guarda

Escola Superior de Tecnologia e Gestão

Orientação Tutorial
Algoritmos e Programação em Python

2 a 6 de Outubro

Python Online
Software para desenvolver programas em `Python`:
PyCharm Edu
Gerador de fluxogramas: Visustin flow chart generator
Entrada e validação de dados
Estruturas condicionais
Estruturas de repetição
Problem 3 -
Problem 4 -
Proposta de exercícios
Desafios

Curso: Engenharia Informática
Unidade Curricular: Algoritmos e Estruturas de Dados
Ano Letivo: 2017/2018
Docente: Paulo Jorge Costa Nunes
Coordenador da área disciplinar: Noel Lopes

Conteúdo

Capítulo 1

Recursos para Python

1.1 Python online

No Website `https://www.jdoodle.com/python3-programming-online` podemos escrever e executar online programas em Python e muitas outras linguagens. Portanto, para escrever e executar programas basta ter um Browser (e.g., Googgle Chrome) e uma ligação à Internet. A figura 1.1 ilustra um exemplo.

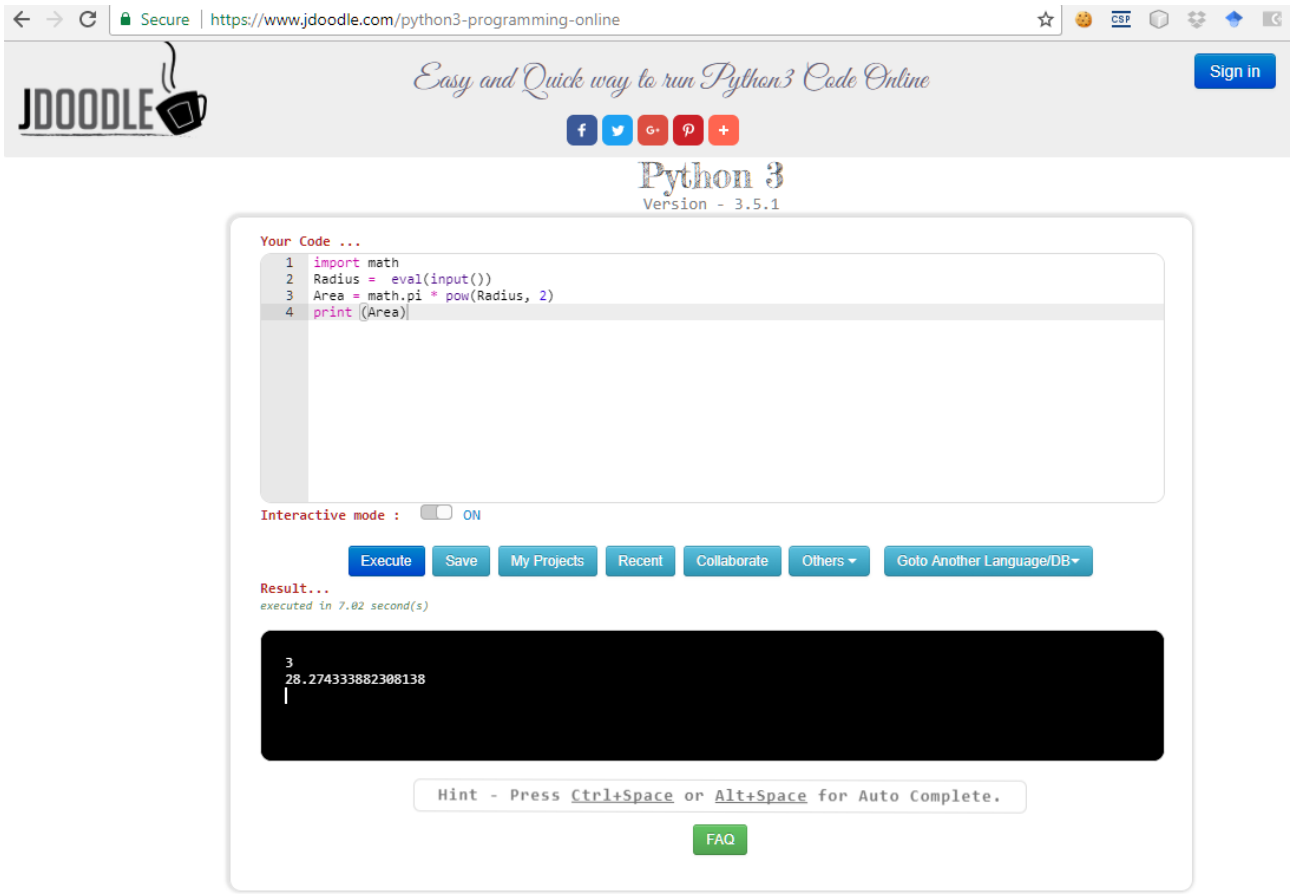

Figura 1.1: Python online: exemplo área de um círculo

1.2 Visustin Flow chart generator

Programa que permite desenhar fluxogramas a partir de programa.

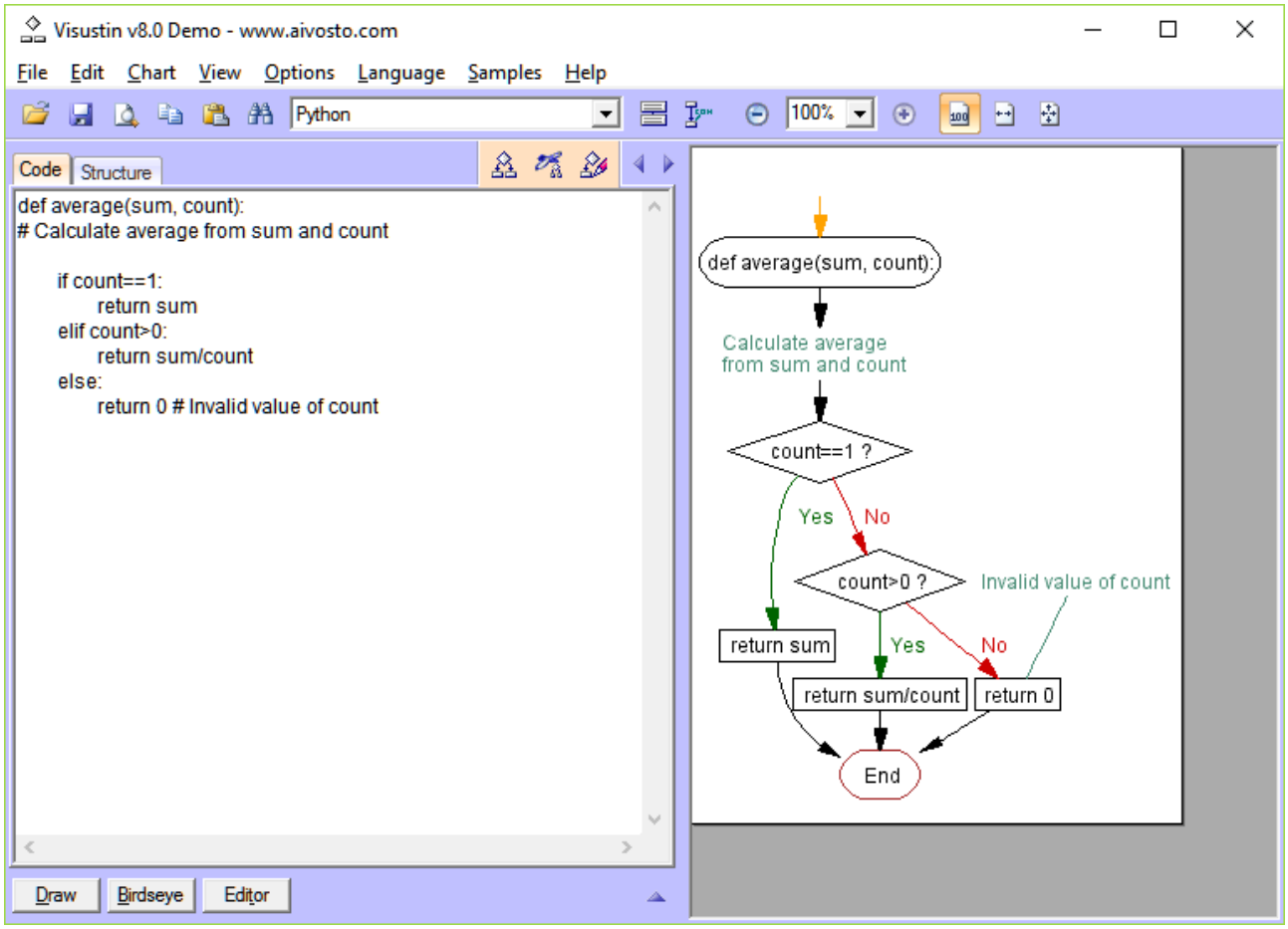

Figura 1.2: Visustin ? Flow chart generator`http://www.aivosto.com/`

Visustin ? Flow chart generator Convert source code to flow charts! Visustin is flowcharting software that documents programs automatically. Create Visio flow diagrams and PDF charts. Create UML diagrams.

Review algorithms. Verify program logic. Document complex functions. Restructure incomprehensible code.

Supports 49 programming languages: ABAP, ActionScript, Ada, ASP, assemblers, AutoIt, BASIC, .bat files, C/C++, C#, Clipper, COBOL, ColdFusion, Delphi, Fortran, GW-BASIC, HTML, Java, JavaScript, JCL, JSP, LotusScript, MATLAB, MXML, Pascal, Perl, PHP, PL/I, PL/SQL, PowerBASIC, PowerBuilder PowerScript, PureBasic, Python, QB, REALbasic, Rexx, RPG, Ruby, SAS, Tcl, T-SQL, shell script, VB, VBA, VBScript, VB.Net, Visual FoxPro, XML and XSLT.

Fonte: `http://www.aivosto.com/`

Examples: `http://www.aivosto.com/visustin-samples.html`

1.2.1 Get number of days in a month

1.3 PyCharm Edu!

Easy and Professional Tool to Learn & Teach Programming with Python.
O `PyCharm Edu!` é um ambiente de desenvolvimento integrado (*Integrated Development Environment* (IDE)), para desenvolvimento de software em `Python`. Pode ser descarregado do endereço `https://www.jetbrains.com/pycharm-edu/`. Requer uma versão do `Python` instalada. A figura 1.6 ilustra o ambiente de desenvolvimento do `PyCharm` com um programa para entrada de dados e o resultado de execução do programa.

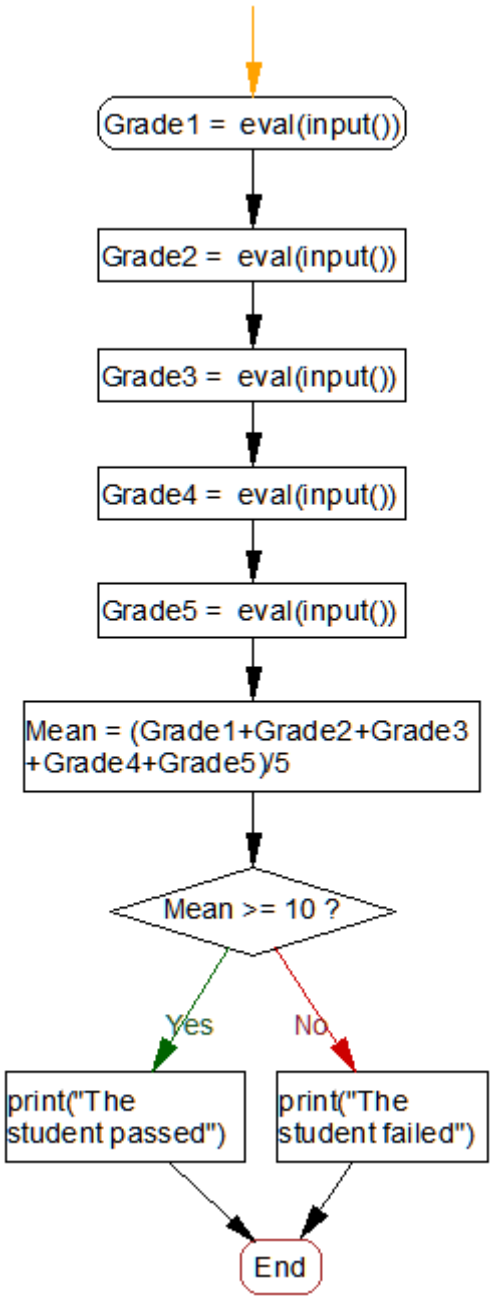

Figura 1.3: Flowchart: student assessment

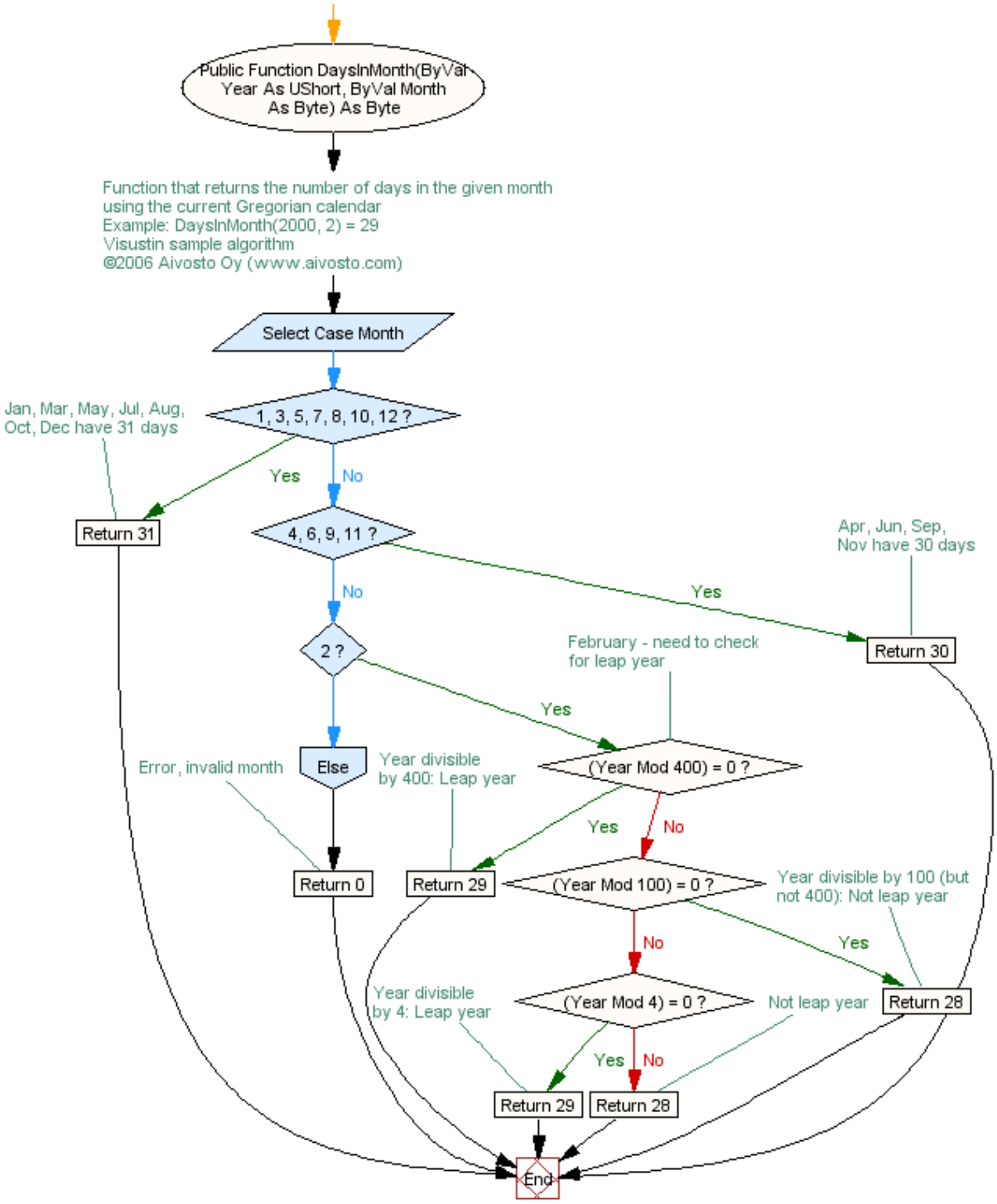

Figura 1.4: Flowchart: days in month.
Fonte: `http://www.aivosto.com/visustin/sample/daysinmonth.html`

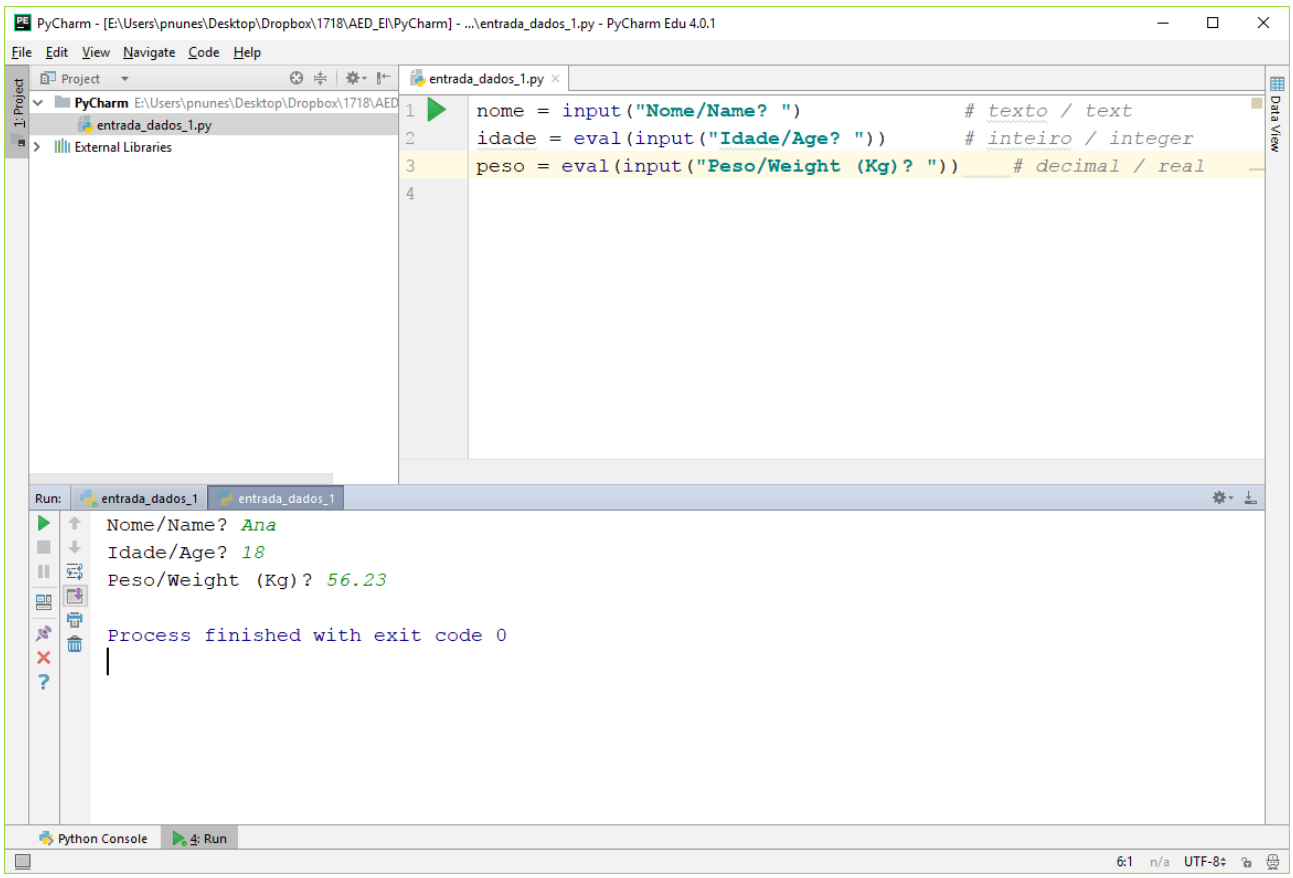

Figura 1.5: Exemplo de programa em PyCharm

1.3.1 Eating pizza

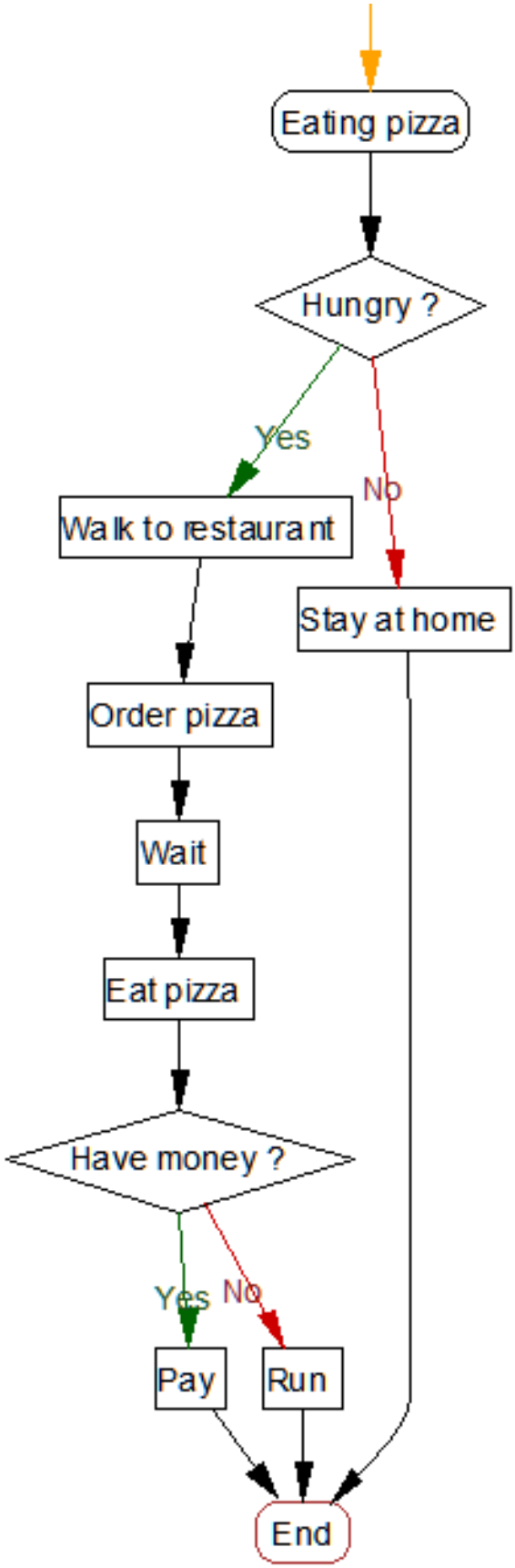

Figura 1.6: Flowchart

1.3.2 Making a cup of tea

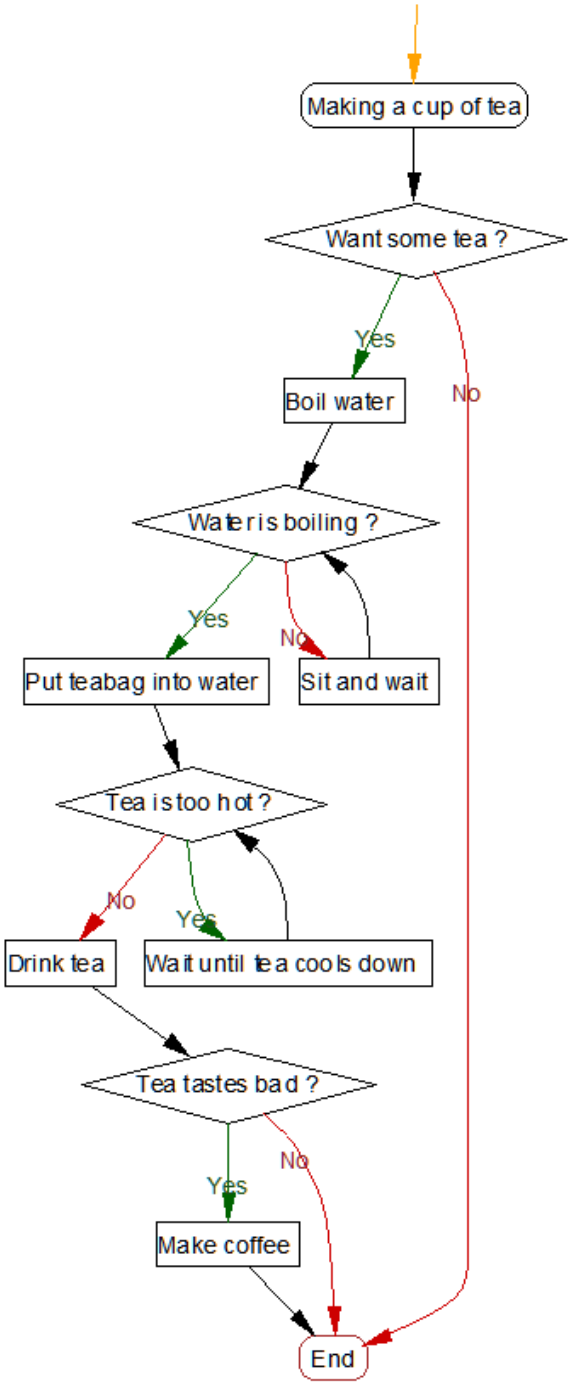

Figura 1.7: Flowchart Making a cup of tea

Capítulo 2

Tipos de dados

No Python não é necessário declarar as variáveis. O tipo da variável depende do valor que armazena. O tipo da variável pode mudar várias vezes durante a execução do programa. Os tipos de vaiáveis básicos em Python são número inteiros, números reais e strings. A listagem 2.2 apresenta vários exemplos.

```
1  vencimento_base  = 3437.34
2  pi = 3.14159
3  raiz_de2 = 1.4142124209
4  e = 2.71828
5  polegada = 2.54
6  grau_celcius = 32.0
7  velocidade_luz = 299792458
8  velocidade_som = 343.4
9  nome = "Carlos"
10 marca = "BMW"
11 marca = 1
```

Listing 2.1: Exemplos de atribuição de valores a variáveis.

O tipo de dados de uma variável determina quais os valores que ele pode ter e quais as operações que podem ser executadas nela. Em Python existe uma função especial para que permite obter o tipo de qualquer variável.

```
1 >>> type(3)
2 <class 'int'>
3 >>> type(3.1)
4 <class 'float'>
5 >>> nome = "Ana"
6 >>> type(nome)
7 <class 'str'>
8 >>>
```

Listing 2.2: Python Shell. Determinação to tipo de diversas variáveis.

Capítulo 3

Entrada e validação de dados

A função de entrada `input()` permite fazer a entrada de dados do utilizador (teclado). Para entrada de strings a sintaxe é a seguinte:

```
<Variável> = input (<prompt>)
```

A sintaxe da função para leitura de números é a seguinte:

```
<Variável> = eval (input (<prompt>))
```

A função `eval()` avalia a cadeia de caracteres digitados pelo utilizador e transforma-o num valor (número em Python). A expressão `prompt` é uma cadeira de caracteres que representa uma mensagens a apresentar ao utilizador.

Na listagem 3.5 são apresentados vários exemplo de entrada e atribuição de valores a variáveis de diversos tipos (números inteiros, número e reais cadeia de texto).

```
1 nome = input("Nome/Name?")         # texto / text
2 idade = eval(input("Idade/Age?"))   # inteiro / integer
3 peso = eval(input("Peso/Weight (Kg)?")) # decimal / real
4
5 >>>
6 Nome/Name? Ana
7 Idade/Age? 18
8 Peso/Weight (Kg)? 56.23
```

Listing 3.1: Entrada e atribuição de valores a variáveis.

3.0.1 Entrada de número inteiros

```
age = int(input("Please enter your age: "))
if age >= 18:
    print("You are able to vote in the United States!")
else:
    print("You are not able to vote in the United States.")

>>>
Please enter your age: 15
You are not able to vote in the United States.

>>>
Please enter your age: Ana
Traceback (most recent call last):
  File "jdoodle.py", line 1, in <module>
    age = int(input("Please enter your age: "))
ValueError: invalid literal for int() with base 10: 'Ana'
```

Listing 3.2: Entrada de números inteiros.

```
while True:
    try:
        # Note: Python 2.x users should use raw_input, the equivalent of 3.
    x's input
        age = int(input("Please enter your age: "))
    except ValueError:
        print("Sorry, I didn't understand that.")
        #better try again... Return to the start of the loop
        continue
    else:
        #age was successfully parsed!
        #we're ready to exit the loop.
        break
if age >= 18:
    print("You are able to vote in the United States!")
else:
    print("You are not able to vote in the United States.")

>>>
Please enter your age: Ana
Sorry, I didn't understand that.
Please enter your age: 17
You are not able to vote in the United States.
```

Listing 3.3: Entrada de números inteiros.

3.0.2 Função para ler números inteiros

```
def get_non_negative_int(prompt):
    while True:
        try:
            value = int(input(prompt))
        except ValueError:
            print("Sorry, I didn't understand that.")
            continue

        if value < 0:
            print("Sorry, your response must not be negative.")
            continue
        else:
            break
    return value

age = get_non_negative_int("Please enter your age: ")
kids = get_non_negative_int("Please enter the number of children you have: ")
salary = get_non_negative_int("Please enter your yearly earnings, in dollars: ")
```

Listing 3.4: Entrada de números inteiros.

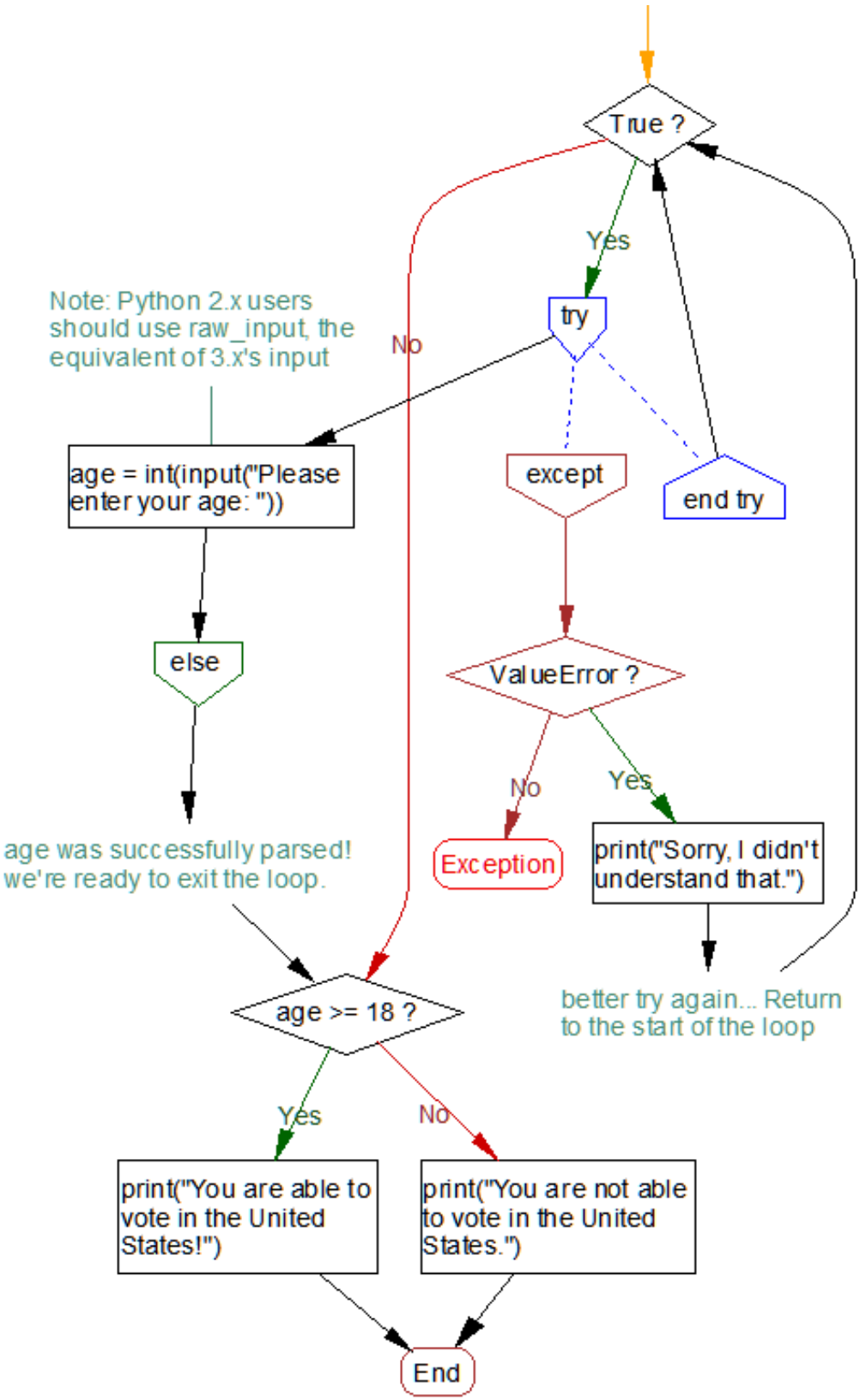
True ?
Yes
No
try
Note: Python 2.x users should use raw_input, the equivalent of 3.x's input
age = int(input("Please enter your age: "))
except
end try
else
ValueError ?
No
Yes
Exception
print("Sorry, I didn't understand that.")
age was successfully parsed! we're ready to exit the loop.
better try again... Return to the start of the loop
age >= 18 ?
Yes
No
print("You are able to vote in the United States!")
print("You are not able to vote in the United States.")
End

Figura 3.1

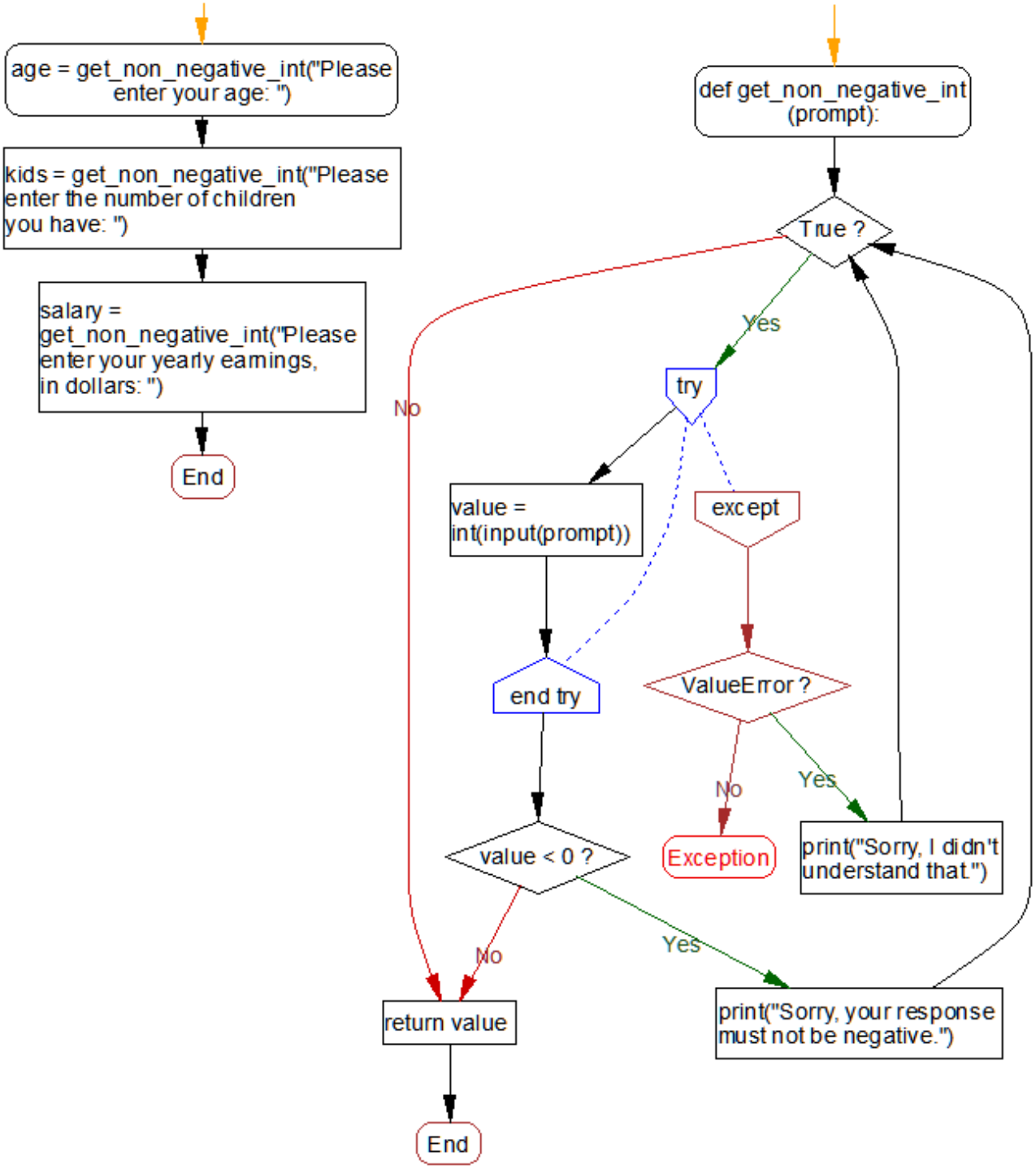
age = get_non_negative_int("Please enter your age: ")
kids = get_non_negative_int("Please enter the number of children you have: ")
salary = get_non_negative_int("Please enter your yearly earnings, in dollars: ")
End
def get_non_negative_int (prompt):
True ?
Yes
No
try
value = int(input(prompt))
except
end try
ValueError ?
No
Yes
Exception
print("Sorry, I didn't understand that.")
value < 0 ?
No
Yes
return value
print("Sorry, your response must not be negative.")
End

Figura 3.2

3.0.3 Função genérica para ler dados

You can extend this idea to make a very generic input function:

```
def sanitised_input(prompt, type_=None, min_=None, max_=None, range_=None):
    if min_ is not None and max_ is not None and max_ < min_:
        raise ValueError("min_ must be less than or equal to max_.")
    while True:
        ui = input(prompt)
        if type_ is not None:
            try:
                ui = type_(ui)
            except ValueError:
                print("Input type must be {0}.".format(type_.__name__))
                continue
        if max_ is not None and ui > max_:
            print("Input must be less than or equal to {0}.".format(max_))
        elif min_ is not None and ui < min_:
            print("Input must be greater than or equal to {0}.".format(min_))
        elif range_ is not None and ui not in range_:
            if isinstance(range_, range):
                template = "Input must be between {0.start} and {0.stop}."
                print(template.format(range_))
            else:
                template = "Input must be {0}."
                if len(range_) == 1:
                    print(template.format(*range_))
                else:
                    print(template.format(" or ".join((", ".join(map(str,
                                                                     range_[:-1])),
                                                       str(range_[-1])))))
        else:
            return ui
With usage such as:

age = sanitised_input("Enter your age: ", int, 1, 101)
answer = sanitised_input("Enter your answer", str.lower, range_=('a', 'b', 'c', 'd'))
```

Listing 3.5: Entrada de números inteiros.

3.1 Desafio / Challenge

3.1.1 Soma dos N primeiros número inteiros

Entrada:

$\{n \in \mathbb{N}^*\}$

Saída: soma

$$\{soma : soma = 1 + 2 + 3 + ...n\} \tag{3.1}$$

3.1.2 Soma de termos em x

Entrada:

$\{x \in \mathbb{R}^+\}$

Saída: soma

$$\{soma : soma = 1 + x + x^2 + x^3 + ...x^n\} \tag{3.2}$$

3.1.3 Factorial de um número

Entrada:

$\{n \in \mathbb{N}^*\}$

Saída: produto

$$\{produto : produto = n \times (n-1) \times (n-2) \times ... \times 2 \times 1\} \tag{3.3}$$

3.1.4 Série de Taylor da Função Exponencial

Calcular o valor de uma Soma Parcial do desenvolvimento em Série de Taylor da Função Exponencial para um dado $x \in \Re$, de modo a que o valor absoluto do último termo somado seja inferior a um dado $\epsilon \in \Re^+$.

Entrada:

$\{x \in \Re, \epsilon \in \Re^+\}$

Saída: soma

$$\{soma : soma = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...\frac{x^n}{n!} \wedge |\frac{x^n}{n!}| \leq \epsilon\} \tag{3.4}$$

Bibliografia