Tecnologia em Sistemas de Comp

Glamber de Souza Faria

1721130501 60

Angra dos Reis

AP2 X - Prog. Com Interfaces Gráficas

①

```
DEF F(A, B):
    IF A == []: RETURN []
    IF A[0] IN B: RETURN F(A[1:], B)
    RETURN F(A[1:], B) + A[:1]
PRINT(F(LIST(RANGE(10)), [1, 9]))
```

②

```
DEF g(A, B):
    IF B == 1: RETURN [[X] FOR X IN A]
    IF LEN(A) == B: RETURN [A]
    RET = []
    FOR C IN g(A[1:], b-1):
        RET.APPEND([A[0]] + C)
    RETURN RET + g(A[1:], B)
PRINT(g(LIST("ABC"), 2))
PRINT(g([1, 2, 3, 4], 3))
```

③

```
FROM OPERATOR IMPORT ITEMGETTER
CLASS PROVA (OBJECT):
    DEF __INIT__(SELF, NOME):
        SELF.NOME = NOME
        SELF.PROVA = []
    DEF RESP (SELF, Q, L):
        IF (SELF.VERIFICAQ(Q)):
            IF (SELF.VERIFICAL(L)):
                IF (SELF.VERIFICAEXISTEQ(Q)):
                    SELF.SUBSTITUI(Q, L)
                ELSE:
                    RESPOSTA = {"Q": Q, "L": L}
                    SELF.PROVA.APPEND(RESPOSTA)
            ELSE:
                PRINT("'" + L + "' NÃO É UM VALOR VALIDO PARA L")
        ELSE:
            PRINT("VALUE ERROR, Q NÃO É NATURAL DE 1 A 25")
    DEF SUBSTITUI (SELF, Q, L):
        FOR I IN RANGE (0, LEN(SELF.PROVA)):
            IF Q == SELF.PROVA[I]["Q"]:
                SELF.PROVA[I]["Q"] = Q
                SELF.PROVA[I]["L"] = L
                BREAK
```

```
DEF VERIFICAEXISTEQ(SELF,Q):
    ISVALID=FALSE
    FOR I IN RANGE(0,LEN(SELF.PROVA)):
        IF Q==SELF.PROVA[I]["Q"]:
            ISVALID=TRUE
            BREAK
    IF ISVALID:
        RETURN TRUE

    ELSE:
        RETURN FALSE
DEF VERIFICAQ(SELF,Q):
    ISVALID=FALSE
    FOR I IN RANGE(1,26):
        IF Q==I:
            ISVALID=TRUE
            BREAK
    IF ISVALID:
        RETURN TRUE
    ELSE:
        RETURN FALSE

DEF VERIFICAL(SELF,L):
    ALTERNATIVA="ABCD"
    ISVALID=FALSE
    FOR I IN ALTERNATIVAS:
        IF L==I:
            ISVALID=TRUE
            BREAK
    IF ISVALID:
        RETURN TRUE
    ELSE:
        RETURN FALSE
```

```
DEF --REPR-- (SELF):
    RETURN STR(SELF.NOME)+":"+STR(SELF.ARRUMAPROVA(
                                  SELF.ORDENAPROVA()))

DEF ORDENAPROVA(SELF):
    RETURN SORTED(SELF.PROVA, KEY=ITEMGETTER('Q'))

DEF ARRUMAPROVA(SELF, PROVAORDENADA):
    QUESTOES = ""
    FOR I IN RANGE(0, LEN(PROVAORDENADA)):
        IF(I != LEN(PROVAORDENADA)-1):
            QUESTOES += STR(PROVAORDENADA[i]["Q"])+'-'+
                        STR(PROVAORDENADA[i]["L"])+' '
        ELSE:
            QUESTOES+=STR(PROVAORDENADA[i]["Q"])+'-'+
                      STR(PROVAORDENADA[i]['L'])+""
    RETURN QUESTOES
```

```
CLASS GABARITO (PROVA):
    DEF __INIT__ (SELF):
        SELF.PROVA = []
    DEF RESP (self, Q, L):
        IF (SELF.VERIFICAQ(Q)):
            IF (SELF.VERIFICAL(L)):
                IF (SELF.VERIFICAEXISTEQ(Q)):
                    SELF.VARIASRESPOSTAS(Q, L)
                ELSE:
                    RESPOSTA = {"Q": Q, "L": L}
                    SELF.PROVA.APPEND(RESPOSTA)
            ELSE:
                PRINT("L" + L + "NÃO É UM VALOR VÁLIDO PARA L")
        ELSE:
            PRINT("VALUE ERROR, Q NÃO É NATURAL DE 1 A 25")
    DEF VARIASRESPOSTAS (SELF, Q, L):
        FOR I IN RANGE (0, LEN(SELF.PROVA)):
            IF Q == SELF.PROVA[i]['Q']:
                SELF.PROVA[i]['L'] = [SELF.PROVA[i]["L"], L]
                BREAK

    DEF VERIFICAQ (SELF, Q):
        Isvalid = False
        FOR I IN RANGE (1, 26):
            IF Q == I:
                Isvalid = TRUE
                BREAK
        IF Isvalid:
            RETURN TRUE
        else
            RETURN FALSE
```

```
def verificaL(self, L):
    alternativas = 'ABCDX'
    isValid = False
    for i in alternativas:
        if L == i:
            isValid = True
            break
    if isValid:
        return True
    else:
        return False

def verificaExisteQ(self, Q):
    isValid = False
    for i in range(0, len(self.prova)):
        if Q == self.prova[i]["Q"]:
            isValid = True
            break
    if isValid:
        return True
    else:
        return False

def ordenaGabarito(self)
    return sorted(self.prova, key = itemgetter('Q'))
```

```
DEF ARRUMA GABARITO (SELF, gabaritoOrdenado):
    QUESTOES = ""
    FOR I IN RANGE(0, LEN(gabaritoOrdenado)):
        IF(LEN(gabaritoOrdenado[i]["L"]) > 1):
            QUESTOES += STR(gabaritoOrdenado[i]['Q']) + '-'
            FOR j IN RANGE(LEN(gabaritoOrdenado[i]['L'])):
                IF(j != LEN(gabaritoOrdenado[i]["L"]) - 1):
                    QUESTOES += STR(gabaritoOrdenado[i]['L'][j]) + '/'
                ELSE:
                    QUESTOES += STR(gabaritoOrdenado[i]['L'][j])
            QUESTOES += ' '
        ELSE:
            IF(I != LEN(gabaritoOrdenado) - 1):
                QUESTOES += STR(gabaritoOrdenado[i]['Q']) + '-' +
                                STR(gabaritoOrdenado[i]['L']) + ' '
            ELSE:
                QUESTOES += STR(gabaritoOrdenado[i]['Q']) + '-' +
                                STR(gabaritoOrdenado[i]['L'])
    RETURN QUESTOES
```

```
DEF NOTA(SELF, P):
    VALORQUESTAO = SELF.VALORQUESTAO()
    ERROS, ACERTOS = SELF.VERIFICAACERTOS(P)
    NOTA = (ACERTOS * VALORQUESTAO) - (ERROS * (VALORQUESTAO/2))
    RETURN NOTA

DEF VERIFICAACERTOS(SELF, P):
    ACERTOS = 0
    ERROS = 0
    ANULADAS = 0
    FOR I IN RANGE(LEN(SELF.PROVA)):
      FOR J IN RANGE(LEN(P.PROVA)):
        IF SELF.PROVA[I]['Q'] == P.PROVA[J]['Q']:
            FOR K IN RANGE(LEN(SELF.PROVA[I]['L'])):
                IF(SELF.PROVA[I]['L'][K] == P.PROVA[J]['L']):
                    ACERTOS += 1
      ELIF SELF.PROVA[I]['L'] == P.PROVA[J]['L']:
        ACERTOS += 1
      ELSE:
        IF SELF.PROVA[I]['L'] != 'X':
          ERROS += 1
        ELSE:
          ANULADAS += 1
    RETURN (ERROS, ACERTOS)
```

```
DEF VALORQUESTAO(SELF):
    Anuladas = 0
    FOR I IN Range(Len(self.PROVA)):
        IF SELF.PROVA[i]['L'] == 'X':
            Anuladas += 1
    RETURN 10/(LEN(self.PROVA)-Anuladas)

DEF __REPR__(SELF):
    RETURN "GABARITO" + STR(SELF.MARCAGABARITO(
                            SELF.ORDENAGABARITO()))

A = PROVA('JOAO')
A.RESP(1,'A'); A.RESP(2,'B'); A.RESP(4,'C'); A.RESP(5,'D')
B = PROVA('MARIA')
B.RESP(1,'B'); B.RESP(2,'B'); B.RESP(3,'D'); B.RESP(5,'D')
g = GABARITO()
g.RESP(1,'B'); g.RESP(2,'B'); g.RESP(3,'C'); g.RESP(3,'D')
g.RESP(4,'A'); g.RESP(5,'X')
TRY:
    g.RESP(2,'E')
EXCEPT VALUEERROR AS E:
    PRINT(E)
```

```
PRINT(g, "-->", g.NOTA(g))
PRINT(A, "-->", g.NOTA(A))
PRINT(B, "-->", g.NOTA(B))
```